Adaptive Multi-Population Differential Evolution for Dynamic Environments

Mathys Cornelius du Plessis

Adaptive Multi-Population Differential Evolution for Dynamic Environments

Mathys Cornelius du Plessis

Submitted in fulfilment of the requirements for the degree of

Philosophiae Doctor

in the Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

April 2012

Supervisor : Prof. A. P. Engelbrecht

Abstract

Dynamic optimisation problems are problems where the search space does not remain constant over time. Evolutionary algorithms aimed at static optimisation problems often fail to effectively optimise dynamic problems. The main reason for this is that the algorithms converge to a single optimum in the search space, and then lack the necessary diversity to locate new optima once the environment changes.

Many approaches to adapting traditional evolutionary algorithms to dynamic environments are available in the literature, but differential evolution (DE) has been investigated as a base algorithm by only a few researchers. This thesis reports on adaptations of existing DE-based optimisation algorithms for dynamic environments.

A novel approach, which evolves DE sub-populations based on performance in order to discover optima in an dynamic environment earlier, is proposed. It is shown that this approach reduces the average error in a wide range of benchmark instances. A second approach, which is shown to improve the location of individual optima in the search space, is combined with the first approach to form a new DE-based algorithm for dynamic optimisation problems.

The algorithm is further adapted to dynamically spawn and remove sub-populations, which is shown to be an effective strategy on benchmark problems where the number of optima is unknown or fluctuates over time.

Finally, approaches to self-adapting DE control parameters are incorporated into the newly created algorithms. Experimental evidence is presented to show that, apart from reducing the number of parameters to fine-tune, a benefit in terms of lower error values is found when employing self-adaptive control parameters.

Keywords: Differential evolution, dynamic environments, competing populations, self-adaptive control parameters, dynamic number of populations, moving peaks.

Supervisor: Prof A. P. Engelbrecht

Department: Department of Computer Science

Degree: Philosophiae Doctor

Acknowledgements

I wish to thank my supervisor, Professor A.P. Engelbrecht for his insight, mentorship and guidance during the course of this research.

Thank you to Professor A.P. Calitz for his encouragement, motivation and valuable discussions regarding the structuring of this thesis.

The experimental work presented in this thesis would not have been possible without the technical assistance of Mr C. van der Merwe and Mr J. Rademakers, to whom I am very grateful.

To Mr J. Cullen, Mr K. Naudé and Dr T. Gibbon, I would like to express my gratitude for their valuable assistance in proofreading this thesis.

A special word of thanks to my colleagues, family and friends for their constant encouragement and support throughout the duration of this research.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Evolutionary algorithms (EAs) are optimisation algorithms inspired by facets of biological evolution. Populations of potential solutions (individuals) are iteratively evolved by encouraging the reproduction of more fit individuals. Mutations and crossovers of parent individuals are used to increase the genetic diversity of subsequent generations. EAs have been shown to be effective on optimisation problems that pose a challenge to more traditional optimisation methods [Bäck *et al.*, 1997].

Differential evolution (DE) is an EA variant which uses the positional difference between randomly selected individuals to create a mutant vector. A trial vector is created through a crossover between a mutant and a parent individual. Selection is implemented as competition between a parent individual and a trial vector. DE has been shown to be an effective optimisation algorithm in static environments.

A dynamic optimisation problem is a problem in which the fitness landscape varies over time. Dynamic optimisation problems are found in many real world domains, for example in air-traffic control, polarisation mode dispersion compensation in optical fibre, and target tracking in military applications [Gibbon *et al.*, 2008] [Li *et al.*, 2006]. A considerable amount of research has been conducted on applying EA to dynamic optimisation problems, but very few of these algorithms are based on DE. The focus of this thesis is on the application of DE variants to dynamic optimisation problems.

This chapter gives a motivation for this research in Section 1.1. The research objectives are listed in Section 1.2. Section 1.3 gives the methodology followed in this thesis. The

main contributions of this study are highlighted in Section 1.4. The scope of this research study is delineated in Section 1.5. A break-down of the thesis structure is presented in Section 1.6.

1.1 Motivation

Despite the fact that evolutionary algorithms often solve static problems successfully, dynamic optimisation problems tend to pose a challenge to evolutionary algorithms [Morrison, 2004]. Lack of diversity is the main disadvantage of most standard evolutionary algorithms when applied to dynamic problems, since the algorithms tend to converge to a single optimum in the fitness landscape and then lack the diversity to locate new global optima when they appear. Differential evolution is one of the evolutionary algorithms that does not scale well to dynamic environments due to lack of diversity [Zaharie and Zamfirache, 2006].

A significant body of work exists on algorithms for optimising dynamic problems (refer to Section 3.3). However, very few of these algorithms are based on DE. DE has been successfully applied to a large number of static optimisation problems (refer to Section 2.4.5), and is generally considered to be a reliable, accurate, robust, and fast optimisation technique [Salman *et al.*, 2007]. DE provides several advantages as an optimisation algorithm:

- The function to be optimised by DE does not have to be differentiable [Ilonen *et al.*, 2003].
- The DE selection and mutation operators are simpler than those of other EA algorithms [Zaharie, 2002b].
- DE has relatively few control parameters [Storn and Price, 1997].
- The DE algorithm exhibits fast convergence [Karaboga and Okdem, 2004].
- The mutation step size of DE is automatically controlled so that the exploration is favoured initially, and exploitation is favoured later in the optimisation process [Xue *et al.*, 2003].

- The space complexity of DE is comparatively low [Das and Suganthan, 2011].

The application of DE to dynamic optimisation problems has only recently begun to receive attention (refer to Section 3.3.3.3), despite the previously mentioned advantages of DE. The relatively small amount of research into DE-based algorithms for dynamic environments means that the performance and behaviour of DE on dynamic optimisation problems have not been thoroughly investigated, and that improvements to current DE-based algorithms can potentially be made. The purpose of this thesis is to investigate, create, and refine DE-based algorithms aimed at dynamic optimisation problems.

1.2 Objectives

The primary objective of this thesis is to create improved DE-based algorithms for dynamic environments. The sub-objectives are as follows:

- Investigate existing algorithms aimed at solving dynamic optimisation problems, specifically focusing on algorithms based on differential evolution.
- Identify specific approaches, tailored to dynamic optimisation problems, that are commonly used in the above algorithms.
- Extend and hybridise existing DE algorithms to create more effective DE-based optimisation algorithms for dynamic environments.
- Identify and investigate dynamic optimisation problem types that have not been thoroughly investigated by other researchers.
- Investigate the use of adaptive control parameters to reduce the number of parameters that must be manually tuned.
- Perform scalability studies on existing and newly created algorithms.
- Compare the performance of the newly created algorithms to the reported results of other algorithms.

1.3 Methodology

The primary objective of this thesis is to create improved DE-based algorithms for dynamic environments. Eight algorithms are created sequentially in this study. Figure 1.1 illustrates how each algorithm relies on one or more base algorithms. Names of new algorithms, developed in this thesis, are printed in bold italics. DynDE [Mendes and Mohais, 2005], a DE-based multi-population algorithm for dynamic environments, is used as the root algorithm.

Two novel algorithms are directly based on DynDE. The first is competitive population evaluation (CPE), which improves the performance of DE by evolving only one sub-population at a time. The selection of sub-populations for evolution is based on performance. Reinitialisation midpoint check (RMC) is a technique aimed at improving the detection process that DynDE uses to determine if more than one sub-population converges to the same optimum. CPE and RMC are combined to form competing differential evolution (CDE). The performance and scalability of DynDE, CPE, RMC and CDE are evaluated on a standard benchmark for dynamic environments. The results of these experiments are used to draw conclusions regarding the effectiveness of the new algorithms. The general applicability of the CPE and RMC approaches is demonstrated by incorporating their respective algorithmic components into *jDE*, a DE-based optimisation algorithm that was developed by Brest *et al.* [2009]. Comparisons are drawn with the reported results of other algorithms in the literature which focus on dynamic optimisation problems.

Dynamic population differential evolution (DynPopDE) is an extension of CDE which dynamically adjusts the number of sub-populations. DynPopDE is aimed at dynamic optimisation problems in which the number of optima is unknown or fluctuates over time. Fluctuating numbers of optima have not been investigated by previous researchers (although an environment of this type has been included in the benchmark set of the IEEE WCCI-2012 competition on evolutionary computation for dynamic optimisation problems [Li *et al.*, 2011]). An adaptation is made to a standard benchmark to investigate the behaviour of DynPopDE on environments in which the number of optima fluctuates over time. A scalability study identifies problem instances where DynPopDE performs better than CDE and DynDE.

CDE is used as base algorithm to investigate the incorporation of self-adaptive control parameters. Three algorithms that self-adapt the DE scale and crossover factors, which were developed for static environments, are identified as potential approaches to work effectively in conjunction with CDE. Adaptations, aimed at improving their effectiveness in dynamic environments, are made to the three approaches. A total of 13 algorithms are consequently evaluated and compared to find the most effective, i.e. jSA2Ran.

Four strategies for self-adapting the Brownian radius (refer to Section 3.4.1.3) are investigated. SABrNorRes is shown to be the most effective of the four strategies. The final algorithms developed are SACDE and SADynPopDE, which are formed by incorporating the self-adaptive components from jSA2Ran and SABrNorRes into CDE and DynPopDE, respectively. The performances of SACDE and SADynPopDE are evaluated on a benchmark function and are compared to the performances of their base algorithms.

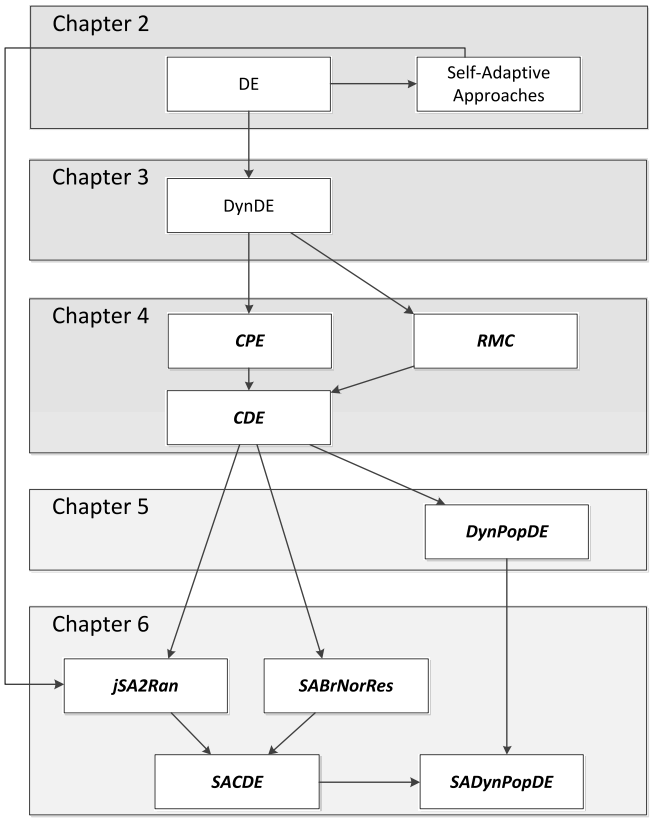

Figure 1.1: Thesis outline

1.4 Contributions

The main contributions of this thesis can be classified under the headings practical and theoretical. These contributions are briefly highlighted below.

Practical Contributions

- A novel approach to improving optimisation in dynamic environments based on sub-populations competing for function evaluations.
- An improvement of the approach used to prevent sub-populations from converging to the same optima by checking for valleys in the fitness landscape.
- A novel algorithm that dynamically spawns and removes sub-populations. This algorithm is aimed at dynamic environments where the number of optima is unknown or fluctuating.
- A novel algorithm to self-adapt the Brownian radius.
- An extension of a standard benchmark that allows the study of optimisation algorithms on environments in which the number of optima fluctuates over time.

Theoretical Contributions

- A unified taxonomy for classifying dynamic environments.
- Algorithmic approaches which are commonly used in dynamic optimisation problems are identified.
- Appropriate values for the sub-population size and number of Brownian individuals are determined.
- The general applicability of the competitive population evaluation and reinitialisation midpoint check approaches is illustrated.
- Evidence is presented to prove that fewer sub-populations should be used than the number of optima that are present in the environment, especially when the number of optima is large.

- An investigation into the most effective self-adaptive scale and crossover approach to use in conjunction with algorithms for dynamic environments is provided.
- An investigation into the most effective approach to self-adapting the Brownian radius is presented.
- Extensive scalability studies of existing and novel algorithms.
- The convergence profiles of the algorithms are investigated.

1.5 Scope

The scope of this research is delineated as follows:

- Dynamic environments in which the number of dimensions vary over time are excluded.
- The effect of different change detecting strategies is not investigated. The algorithms considered are independent of the change detection mechanism. The default detection strategy used is a change detection "oracle" which uses no function evaluations and detects changes perfectly. However, the optimisation algorithms employ actual change detection strategies when comparing these results with the published results of other researchers that used detection strategies.
- Only dynamic environments in which changes are periodic are considered. This constraint applies to the environments that are investigated, but is not exploited by the algorithms presented in this thesis. Specifically, at no point is it known to the algorithms how many function evaluations remain before the next change in the environment.

1.6 Thesis structure

This thesis is structured as follows:

Chapter 2 contains a brief overview of optimisation and evolutionary algorithms. Differential evolution is discussed in detail with reference to DE schemes and control pa-

rameters. A literature review of approaches to self-adapt or eliminate control parameters is included. Dynamic environments are described and the main factors which influence optimisation algorithm effectiveness are identified. The benchmarks and performance measures that are used in this study are discussed.

Chapter 3 introduces related work solving dynamic optimisation problems. The chapter commences with a discussion on how normal DE performs on dynamic optimisation problems. Algorithms presented by other researchers in the literature are reviewed. Three broad categories of strategies, which are followed to adapt optimisation algorithms to dynamic environments, are given. Seven specific strategies that fall within these categories are identified. It is argued that most current algorithms aimed at dynamic environments utilise at least one of the seven strategies. Two DE-based algorithms, DynDE and *jDE*, are reviewed in detail. The chapter concludes with a brief description of techniques used to detect changes in the environment.

Chapter 4 presents novel extensions to DynDE. An alternative method of calculating the DynDE exclusion threshold in the absence of knowledge regarding the number of optima present in the environment is given. Competitive population evaluation (CPE) is described and motivated. Reinitialisation midpoint check (RMC) is presented and potential pitfalls with the use of this approach are described. CPE and RMC are combined to form competing differential evolution (CDE). The scalability and performances of DynDE, CPE, RMC and CDE are compared. The general applicability of the CPE and RMC approaches is illustrated by their incorporation into another DE-based algorithm aimed at dynamic optimisation problems. CDE is compared to other algorithms in the literature.

Chapter 5 presents DynPopDE, an extension of CDE aimed at dynamic optimisation problems where the number of optima is unknown, or unknown and fluctuating over time. The technique used by DynPopDE to spawn or remove sub-populations is discussed and motivated. The performance of DynPopDE on problems where the number of optima is unknown is compared to that of DynDE and CDE. In addition, DynPopDE results are compared to results of other algorithms reported in the literature. DynPopDE is tested on an extension of one of the benchmarks that allows for fluctuating numbers of optima. Comparative results with DynDE and CDE are presented.

Chapter 6 incorporates self-adaptive control parameter approaches into CDE. Several

approaches and variations of the approaches are investigated to self-adapt the DE scale and crossover factors. A new approach to self-adapting the Brownian radius is presented. The self-adaptive algorithms are combined to form SACDE. The performance and scalability of SACDE are compared to that of its predecessor algorithms and results of other algorithms reported in the literature. The self-adaptive approach is incorporated into DynPopDE to form SADynPopDE. The performance of SADynPopDE on problems where the number of optima is unknown or fluctuates is compared to the performance of DynPopDE and SACDE.

Chapter 7 summarises the conclusions found in this study and describes avenues of future research.

Several appendices are included in this thesis. Appendix A empirically shows the parameter dependence of *jDE*, a DE-based algorithm aimed at dynamic optimization problems (refer to Section 3.4.2).

Appendices B, C and D, provide additional results relevant to Chapters 4, 5, and 6, respectively. Appendix E lists the symbols used in this thesis, while Appendix F gives a list of acronyms that are used. Appendix G lists the publications that were derived from this research study.

Chapter 2

BACKGROUND

This chapter provides a contextual background for the rest of the thesis. A brief overview of optimisation techniques and general evolutionary algorithms is followed by a discussion of the differential evolution algorithm, common variations and control parameters. Related research on eliminating and adapting differential evolution control parameters is briefly outlined. Dynamic environments with specific focus on the benchmarks used in this research are described. A review of performance measures available in the literature is presented.

2.1 Introduction

Optimisation is an important branch of mathematics, engineering, and computer science. Many real-world problems can be reduced to optimisation problems, for which solutions can be found iteratively using optimisation algorithms (refer to Section 2.2 where optimisation algorithms are described). Differential evolution (DE) is an optimisation algorithm which is based on ideas from biological evolution. Section 2.4 describes the basic DE algorithm, its common variations, and DE control parameters. Best results are found if DE control parameters are fine-tuned for each specific problem. This process can be time consuming and should ideally be avoided. A literature survey on adapting and eliminating DE control parameters is included in Section 2.4.

The focus of this thesis is on the application of DE to dynamic optimisation problems (dynamic environments). A dynamic environment is a problem space that does not remain

static over time, but which changes periodically. Section 2.5 formally defines the concept of dynamic environments and describes the different types of dynamic environments.

The development of a new algorithm requires a test-bed of problems on which to evaluate the effectiveness of the new algorithm, explore the scalability of the algorithm, and compare the performance of the algorithm to existing algorithms. Researchers have developed benchmark problems to provide a common platform for the evaluation and comparison of algorithms aimed at optimisation in dynamic environments. Two benchmarks are used in this study, namely the moving peaks benchmark (MPB) and the generalised dynamic benchmark generator (GDBG). These benchmarks are discussed in Section 2.5, which also contains a survey of approaches to measure the performance of optimisation algorithms in dynamic environments. Conclusions are drawn in Section 2.6.

2.2 Optimisation

Optimisation refers to the process of iteratively refining the proposed solution to a problem until a satisfactory result is reached. In the n_d dimensional real-valued number space, a minimisation optimisation problem is defined as the process of finding $\vec{x}^*$ with $x_j^* \in \mathbb{R}$ for $j = 1, \ldots, n_d$ such that

$$F(\vec{x}^*) \leq F(\vec{x}) \; \forall \; \vec{x} \in \mathcal{S}^{n_d} \tag{2.1}$$

where F is a function that gives the value of the potential solution vector $\vec{x}$ and $\mathcal{S}^{n_d} \subseteq \mathbb{R}^{n_d}$ denotes the search space. The vector $\vec{x}^*$ is referred to as a global optimum of F. Note that multiple global optima may exist. A local optimum is a point in the fitness landscape which gives the lowest value for F within a subregion of $\mathcal{S}^{n_d}$, but does not represent the lowest value of F in the entire fitness landscape. Equation (2.1) is an example of a function minimisation problem (i.e. the goal is to find the lowest value of function F). Problems where it is necessary to find the maximum value of a function is referred to as function maximisation problems. A global optimum position, $\vec{x}^*$, must thus be found such that

$$F(\vec{x}^*) \geq F(\vec{x}) \; \forall \; \vec{x} \in \mathcal{S}^{n_d} \tag{2.2}$$

The value of the solution vector, $\vec{x}^*$, can typically not be found analytically and large branches of applied mathematics and computer science have been dedicated to optimisa-

tion. Historically, many optimisation techniques are derivative-based [Burden and Faires, 2001]. Essentially, derivative-based approaches make use of the gradient of the function to perform stepwise updates to the solution vector. A basic derivative-based function minimisation optimisation approach is given in Algorithm 1.

Algorithm 1: Simple Derivative-Based Optimisation

Randomly select vector $\vec{x}$ in the n_d dimensional search space;
$t = 0$;
while $\vec{x}(t)$ *is not a satisfactory solution* **do**
  foreach $j \in \{1, \ldots, n_d\}$ **do**

$$x_j(t+1) = x_j(t) - \eta \cdot \frac{\partial F(\vec{x}(t))}{\partial x_j(t)} \tag{2.3}$$

  $t = t + 1$;
  end
end

The learning rate, η, is a constant that controls the size of the steps taken up (for maximisation) or down (for minimisation) the gradient of the fitness landscape. A large step size increases the rate of convergence of Algorithm 1, but increases the risk of stepping over a good optimum. A small step size typically increases the convergence time and the risk of becoming stuck in a local optimum, but ensures a finer grained search. An adaptive function, $\eta(t)$, is often used to provide a large step size initially and an increasingly smaller step size to refine the solution during the later stages of optimisation. It is important to note that Algorithm 1 is not guaranteed to converge to a global optimum.

A disadvantage of derivative-based optimisation is that two requirements must be met. Firstly, a continuous function must exist that fully describes the problem space. Secondly, the function must be differentiable. These requirements are not met for all optimisation problems, which means that derivative-based approaches cannot be applied to all optimisation problems. A further disadvantage of derivative-based approaches is that, in the process of descending down the gradient, the algorithm could become trapped in a local optimum. Since the algorithm cannot recover from such situations, suboptimal

results may be found.

Research conducted into the simulation of biological evolution by scientists like Barricelli [1957] and Fraser [1957] on early electronic computers marked the advent of the field of evolutionary computation. Evolutionary algorithms (EAs) draw on ideas from genetics and Darwinian evolution [Darwin, 1859] to stochastically evolve solutions to optimisation problems. An elementary EA to minimise a function F is given in Algorithm 2.

Algorithm 2: Elementary Evolutionary Algorithm

Generate a population, P_x, of I individuals by creating vectors of random candidate solutions, $\vec{x}_i$, $i = 1, \cdots, n_I$;

Evaluate the fitness, $F(\vec{x}_i)$, of all individuals;

while *termination criteria are not met* **do**

  Select parent individuals from P_x to reproduce;

  Create offspring from parent individuals through reproduction;

  Produce the next population from current parents and offspring;

  Evaluate the fitness, $F(\vec{x}_i)$, of all individuals;

end

Candidate solutions (individuals) are referred to as chromosomes, in an analogy to biological evolution. Individuals are given a representation based on the problem and the specific evolutionary algorithm used. Binary vectors were initially used in genetic algorithms (GAs), popularised by Holland [1975], while more recently real-valued numbers are commonly used as the representation scheme. In genetic programming, individuals are normally represented as trees [Koza, 1992]. Evolutionary programming, which is an example of phenotypic evolution rather than genotypic, use individuals to represent behavioural traits.

The initial population of individuals is randomly generated based on the representation used. Ideally, the initial population will uniformly represent the entire search space.

A fitness function is created to quantify the quality of each potential solution. This can, for example, take the form of a function that gives the error of any individual within the search space, or merely the function value at the point represented by an individual.

Termination criteria depend on the problem, but commonly, a set number of genera-

tions or a threshold for fitness is used.

The selection process in Algorithm 2 is generally done based on fitness. More fit individuals are assigned a higher likelihood of reproducing than less fit individuals. Components from good solutions are consequently spread through the population. A strong focus on the most fit individual increases the selection pressure, which may lead to the population being dominated by a single solution [Engelbrecht, 2007].

In GAs, reproduction usually involves crossover and mutation. Crossover is the process of combining genetic building blocks from two or more parent vectors to form one or more new offspring individuals. Mutation is the process of injecting random noise into offspring vectors to form a slightly different offspring individual, thereby increasing the genetic diversity of the population.

Broad and often overlapping subfields of evolutionary computation include genetic algorithms, genetic programming [Koza, 1992], evolutionary programming [Fogel, 1962], evolution strategies [Rechenberg, 1973], cultural evolution [Reynolds and Sverdlik, 1994] and differential evolution [Storn, 1996].

Particle swarm optimisation (PSO) [Kennedy and Eberhart, 1995] is often included in the above list, although it is strictly not an evolutionary algorithm [Engelbrecht, 2006]. PSO mimics the swarming behaviour of bird flocks to explore optimisation problem spaces.

Evolutionary algorithms and swarm-based algorithms solve the challenges faced by derivative-based approaches which were mentioned earlier. EAs are global optimisation techniques and thus more, global information about the fitness landscape to determine new candidate solutions [Engelbrecht, 2007], making EAs less likely to converge to a local optimum. Furthermore, the function to be optimised need not be differentiable or continuous. Technically, all that is required is a function that, given two potential solutions to the problem, can indicate which one is the better solution.

This thesis focuses on differential evolution, which is discussed in Section 2.4.

2.3 Genotypic Diversity

Central to the behaviour of an EA in a search space is the concept of genotypic diversity. Genotypic diversity (subsequently referred to simply as diversity) refers to the dispersion

of individuals in the search space.

A large portion of the fitness landscape is searched when individuals are spread over a large area. This corresponds to a highly diverse population and such a wide search of the fitness landscape is referred to as exploration. A fine-grained search of the fitness landscape results from low diversity (i.e. individuals are clustered closely together). This is referred to as exploitation.

Typically, EAs commence with a high diversity population which favours exploration. As time passes the individuals converge which lowers the diversity and leads to exploitation. Premature convergence, which refers to the population's converging to a local optimum, can be the result of transitioning from exploration to exploitation too early [Riget and Vesterstrøm, 2002]. A balance must thus be reached between maintaining diversity and rapid convergence [Ursem, 2002].

Several diversity measures are available in the literature [Olorunda and Engelbrecht, 2008]. Diversity measures may be algorithm specific, for example tree similarity measures used for genetic programming [Burke *et al.*, 2004]. However, a basic diversity measure, D_{basic}, for an EA with a population of size n_I in n_d dimensions can be formulated as:

$$D_{basic} = \sum_{i=1}^{n_I} \|\vec{d} - \vec{x}_i\|_2 \tag{2.4}$$

where $\vec{d}$ is the average location of all individuals, calculated as:

$$d_j = \frac{\sum_{i=1}^{n_I} x_{i,j}}{n_I}, \ \forall \ j \in n_d \tag{2.5}$$

The diversity is thus calculated as the sum of the Euclidian distances of all the individuals from their average location. A low value of D_{basic} implies that the individuals are grouped spatially close together, while a large value implies that individuals are dispersed throughout the fitness landscape.

The diversity measure as calculated in equation (2.4) does not take into account that a large population size may artificially increase the diversity value (since the sum over the Euclidian distances would be taken over a large number of individuals). This problem can be remedied by normalising with respect to the population size, by changing the diversity

calculation to give the normalised diversity, D_{norm} [Krink *et al.*, 2002]:

$$D_{norm} = \frac{\sum_{i=1}^{n_I} \|\vec{d} - \vec{x}_i\|_2}{n_I} \tag{2.6}$$

The magnitude of the range of the search space has an influence on how the calculated diversity value is interpreted. For example, $D = 0.8$ would be considered a high diversity in a two dimensional search space with range $[0, 1]$ for each dimension, but low in a 100 dimensional search with range $[0, 100000]$ for each dimension. Therefore, to simplify the interpretation of the diversity value, it is common to normalise with respect to the length of the longest diagonal in the search space, L [Ursem, 2002][Riget and Vesterstrøm, 2002]. Equation (2.6) is thus further adapted to calculate diversity, D, as

$$D = \frac{\sum_{i=1}^{n_I} \|\vec{d} - \vec{x}_i\|_2}{n_I L} \tag{2.7}$$

Equation (2.7) will be used as diversity measure throughout this thesis.

2.4 Differential Evolution

The purpose of this section is to describe differential evolution (DE) and to discuss variations of the basic DE algorithm. The original DE algorithm is discussed in Section 2.4.1, followed by a discussion of DE schemes in Section 2.4.2 and DE control parameters in Section 2.4.3. A review of approaches to eliminate or adapt DE control parameters is given in Section 2.4.4. DE applications are listed in Section 2.4.5.

2.4.1 Basic Differential Evolution

Differential evolution (DE) is a population-based, stochastic optimisation algorithm, created by Storn and Price [Price *et al.*, 2005] [Storn and Price, 1997]. DE differs from other EAs in the following aspects:

- Mutational step-sizes are not sampled from a prior specified distribution. Rather, mutations are made based on the spatial difference between two or more individuals added to a base vector.
- Mutation is applied first and thereafter the parent is combined with the mutated individual.

- Selection is applied as competition between the parent and the offspring.

Several variants of the DE algorithm have been suggested, but the original algorithm is given in Algorithm 3.

The following definitions are used for symbols in Algorithm 3: Each $\vec{v}_i$ is known as a mutant vector, $\mathcal{F}$ is known as the scale factor, $\vec{x}_{i_1}$ is referred to as the base vector, Cr is referred to as the crossover probability, $\vec{u}_i$ is referred to as a trial or offspring vector, and each $\vec{x}_i$ that is tested for replacement by $\vec{u}_i$ is known as a target or parent vector. The subscripts of $\vec{x}_{i_1}$, $\vec{x}_{i_2}$ and $\vec{x}_{i_3}$ are simplified to $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$, when no confusion is possible.

A thorough comparison of DE with 16 other optimisation algorithms showed that, while DE is not always the fastest method, it frequently produced the best results on a large number of benchmark problems [Price *et al.*, 2005]. In the first international contest on evolutionary optimisation, held in 1996, the performance of DE on benchmark functions was ranked third amongst several competing algorithms [Bersini *et al.*, 1996]. Vesterstrøm and Thomsen [2004] showed that DE performed better than PSO variants and a simple evolutionary algorithm on several benchmark functions. DE is now generally considered to be a reliable, accurate, robust and fast optimisation technique [Salman *et al.*, 2007].

2.4.2 Differential Evolution Schemes

Most variations of DE (called schemes) are based on different approaches to create the mutant vectors, $\vec{v}_i$ [Storn, 1996] (see equation (2.8)), and different approaches to create trial vectors. One of two crossover schemes are typically used to create trial vectors. The first, binomial crossover, is used in equation (2.9). The second approach is called exponential crossover, given in Algorithm 4.

By convention, schemes are labelled in the form DE/α/β/γ, where α is the method used to select the base vector, β is the number of difference vectors, and γ is the method used to create offspring. The scheme used in Algorithm 3 is referred to as DE/rand/1/bin.

Several methods of selecting the base vector have been developed and can be used with either of the crossover methods. Popular base vector selection methods include [Storn, 1996][Storn and Price, 1996] [Mezura-Montes *et al.*, 2006] (in each case the selected vectors' indexes are assumed to be unique):

Algorithm 3: Basic Differential Evolution

Generate a population, P_x, of n_I individuals by creating vectors of random candidate solutions, $\vec{x}_i$, $i = 1, \cdots, n_I$ and a dimensionality of n_d;

Evaluate the fitness, $F(\vec{x}_i)$, of all individuals.;

while *termination criterion are not met* **do**

  foreach $i = 1, \ldots, n_I$ **do**

    Select three individuals, $\vec{x}_{i_1}$, $\vec{x}_{i_2}$ and $\vec{x}_{i_3}$, at random from the current population such that $i_1 \neq i_2 \neq i_3 \neq i$;

    Create a new mutant vector $\vec{v}_i$ using:

$$\vec{v}_i = \vec{x}_{i_1} + \mathcal{F} \cdot (\vec{x}_{i_2} - \vec{x}_{i_3}) \tag{2.8}$$

    where $\mathcal{F} \in (0, \infty)$;

    Create trial vector $\vec{u}_i$ as follows:

$$u_{i,j} = \begin{cases} v_{i,j} \text{ if } (U(0,1) \leq Cr \text{ or } j == r) \\ x_{i,j} \text{ otherwise} \end{cases} \tag{2.9}$$

    where $Cr \in (0, 1)$ is the crossover probability and r is a randomly selected index, i.e. $r \sim U(1, n_d)$;

    Evaluate the fitness of $\vec{u}_i$;

  end

  foreach *target vector, $\vec{x}_i$, in the current population* **do**

    Select corresponding $\vec{u}_i$ from trial population;

    If $\vec{u}_i$ has a better fitness value than $\vec{x}_i$ then replace $\vec{x}_i$ with $\vec{u}_i$;

  end

end

Algorithm 4: Exponential Crossover

$r \sim U(1, n_d)$;

$j = r$;

repeat

  $u_{i,j} = v_{i,j}$;

  $j = ((j-1) \bmod (n_d - 1)) + 1$;

until $U(0,1) \geq Cr$ or $j == r$;

while $j \neq r$ **do**

  $u_{i,j} = x_{i,j}$;

  $j = ((j-1) \bmod (n_d - 1)) + 1$;

end

DE/rand/2: Two pairs of difference vectors are used:

$$\vec{v}_i = \vec{x}_1 + \mathcal{F} \cdot (\vec{x}_2 + \vec{x}_3 - \vec{x}_4 - \vec{x}_5) \tag{2.10}$$

DE/best/1: The best individual in the population is selected as the base vector:

$$\vec{v}_i = \vec{x}_{best} + \mathcal{F} \cdot (\vec{x}_1 - \vec{x}_2) \tag{2.11}$$

Using the best individual as the base vector encourages fast convergence (a consequence of which is exploitation) since offspring individuals will always be a mutation of the best individual.

DE/best/2: The best individual in the population is used as base vector in conjunction with two pairs of difference vectors:

$$\vec{v}_i = \vec{x}_{best} + \mathcal{F} \cdot (\vec{x}_1 + \vec{x}_2 - \vec{x}_3 - \vec{x}_4) \tag{2.12}$$

DE/rand-to-best/1: A point between a randomly selected individual and the current best individual is used as the base vector:

$$\vec{v}_i = \vec{x}_1 + \mathcal{G} \cdot (\vec{x}_{best} - \vec{x}_1) + \mathcal{F} \cdot (\vec{x}_2 - \vec{x}_3) \tag{2.13}$$

where $\mathcal{G}$ is a parameter that scales the contribution of the position of the current best individual to the resultant base vector. Should $\mathcal{G}$ be assigned a value of one,

DE/rand-to-best/1 is equivalent to DE/best/1 (which encourages exploitation). A value of zero for $\mathcal{G}$ would result in the position of the best individual being ignored, thus favouring exploration, as the mutant vector would not be biased towards the best individual. $\mathcal{G}$ thus controls the balance between exploitation and exploration of the fitness landscape, and is referred to as the greediness of the scheme.

DE/current-to-best/1: A point between the target vector and the current best individual is used as the base vector:

$$\vec{v}_i = \vec{x}_i + \mathcal{G} \cdot (\vec{x}_{best} - \vec{x}_i) + \mathcal{F} \cdot (\vec{x}_1 - \vec{x}_2) \tag{2.14}$$

where $\mathcal{G}$ is a parameter that scales the contribution made by the target and the best individual to the resultant base vector. A $\mathcal{G}$ value of zero results in only the target vector contributing, while a value of one results in only the best individual contributing. Exploration versus exploitation around the target vector and best individual is thus controlled by $\mathcal{G}$.

DE/current-to-rand/1: A point between a randomly selected individual and the target vector is used as the base vector:

$$\vec{v}_i = \vec{x}_i + \mathcal{G} \cdot (\vec{x}_1 - \vec{x}_i) + \mathcal{F} \cdot (\vec{x}_2 - \vec{x}_3) \tag{2.15}$$

where $\mathcal{G}$ is a parameter that scales the contribution made by the target and the randomly selected individual to the resultant base vector. A $\mathcal{G}$ value of zero results in only the target vector contributing, while a value of one results in only the randomly selected individual contributing. Exploration versus exploitation around the target vector is thus controlled by $\mathcal{G}$.

2.4.3 Differential Evolution Control Parameters

Differential evolution algorithms have several control parameters that have to be set. Ignoring extra parameters introduced by some DE schemes, the main DE control parameters are the population size (n_I), scale factor ($\mathcal{F}$) and crossover factor (Cr).

The scale factor controls the magnitude of the difference vector and consequently the amount by which the base vector is perturbed. Large values of $\mathcal{F}$ encourage large scale

exploration of the fitness landscape but could lead to premature convergence, while small values result in a more detailed exploration of the local fitness landscape (exploitation) while increasing convergence time.

The crossover factor controls the diversity of the population, since a large value of Cr will result in a higher probability that new genetic material will be incorporated into the population. Large values of Cr result in broad exploration of the fitness landscape, which in turn may result in slow convergence. Conversely, very small values result in very little genetic material being introduced into the population which may lead to premature convergence. A value for Cr must thus be used that is large enough to ensure sufficient exploration, but small enough to allow exploitation and consequently acceptably fast convergence times.

General guidelines for the values of parameters that work reasonably well on a wide range of problems are known. For example, Zaharie [2002a] showed on three test functions that the smallest reliable value for $\mathcal{F}$ is 0.3 when $Cr = 0.2$ and $n_I = 50$. Storn and Price [1997] showed that $0 \leq Cr \leq 0.2$ worked well on decomposable functions, while $0.9 \leq Cr \leq 1$ worked well on functions that are not decomposable. Other recommendations are available in the literature [Storn, 1996][Rönkkönen *et al.*, 2005][Pedersen, 2010], however, best results in terms of accuracy and convergence time are found if the parameters are tuned for each problem individually [Engelbrecht, 2007].

The population size influences the diversity of the population. A large population size, rather than a small population size, makes a more diverse sampling of the search space likely. The population size also controls the number of function evaluations that are performed per iteration. A small population size would result in a larger number of iteration than would result from a large population size, when a constant number of function evaluations is available. A value for n_I must thus be selected to ensure that an adequate number of iterations can be performed, but also that the fitness landscape is sufficiently explored.

2.4.4 Adapting and Eliminating Control Parameters

A disadvantage of control parameters is that their fine-tuning is a time consuming manual task. Furthermore, ideal values for the control parameters may vary during the evolution

process. For example, values that encourage exploration may be desirable initially, while values that encourage exploitation may be effective at a later time during the algorithm's execution.

Three broad strategies have emerged to address the disadvantages associated with DE control parameters. The first strategy uses adaptive control parameters and is discussed in Section 2.4.4.1. The values of the control parameters are adapted using information gathered during the optimisation process or to predetermined values. The second strategy is to use self-adapting control parameters. This is discussed in Section 2.4.4.2. Parameters are incorporated into the evolution process, which results in the optimisation of the control parameters in parallel with optimising the fitness landscape [Eiben *et al.*, 2000]. The third strategy is to explicitly eliminate the need to tune control parameters from the algorithm and is discussed in Section 2.4.4.3.

The computational intelligence community have not reached consensus on when an algorithm should be classified as "adaptive" and when it should be classified as "self-adaptive". For example, Brest *et al.* [2006] and Qin and Suganthan [2005] describe their algorithms as "self-adaptive", while Zhang and Sanderson [2009] argue that these algorithms should be described as "adaptive". The following convention is used in this thesis: Algorithms that explicitly control the values of control parameters during the optimisation process are classified as "adaptive". For example, linearly decreasing the scale factor as a function of time (initially to encourage exploration and later to encourage exploitation) are classified as "adaptive". Algorithms that select control parameters based on the success rate of previous values during the optimisation process are classified as "self-adaptive".

Ideally, algorithms that adapt control parameters should reduce the number of parameters, or preferably eliminate all parameters. However, the literature review presented in this section illustrates that, in practice, adaptive and self-adaptive control parameter algorithms do not in all cases reduce the number of parameters. Several approaches to adapting or eliminating DE parameters are available in the literature and are discussed in this section. Algorithms that are relevant to Chapter 6, which presents the incorporation of self-adaptive control parameters into the algorithms that are developed in this thesis, are described in more detail.

2.4.4.1 Adaptive Control Parameters

A simple method of adapting a control parameter is to linearly vary it between two values as a function of the number of iterations. Das *et al.* [2005] linearly decreased the scale factor from 1.2 to 0.4 during the course of the optimisation process. A disadvantage of this approach is that the number of iterations that will be performed must be known in advance.

Ali and Törn [2004] proposed an algorithm that adapts the scale factor at the end of each generation using:

$$\mathcal{F}_{new} = \begin{cases} \max\left\{0.4, 1 - \left|\dfrac{\max\limits_{\vec{x}_a \in P_x}\{F(\vec{x}_a)\}}{\min\limits_{\vec{x}_b \in P_x}\{F(\vec{x}_b)\}}\right|\right\} & \text{if } \left|\dfrac{\max\limits_{\vec{x}_a \in P_x}\{F(\vec{x}_a)\}}{\min\limits_{\vec{x}_b \in P_x}\{F(\vec{x}_b)\}}\right| < 1 \\ \max\left\{0.4, 1 - \left|\dfrac{\min\limits_{\vec{x}_a \in P_x}\{F(\vec{x}_a)\}}{\max\limits_{\vec{x}_b \in P_x}\{F(\vec{x}_b)\}}\right|\right\} & \text{otherwise} \end{cases} \tag{2.16}$$

The effect of this approach is that small scale factors are used when the difference in function value between the best and the worst individuals in the population is large (this should occur during the initial stages of the evolution). The approach of Ali and Törn [2004] thus initially encourages exploitation. Larger scale factors are produced when the difference in function value between the best and the worst individuals in the population is small (which occurs once the population has converged). Exploration is thus favoured during the later stages of the optimisation process. A disadvantage of this algorithm is that adding a constant to F would result in different scale factors being produced.

Liu and Lampinen [2005] proposed an approach that uses fuzzy logic controllers to adapt the scale factor, $\mathcal{F}$, and crossover factor, Cr. This algorithm is called fuzzy adaptive differential evolution (FADE). FADE employs knowledge of a human expert (in the form a fuzzy rules) in order to adjust values of $\mathcal{F}$ and Cr based on differences in position and fitness of individuals in successive generations. It was shown that this approach outperformed normal DE on high dimensional problems. A disadvantage of FADE is that the fuzzy rules have to be manually created for a specific type of problem and consequently depends on the subjectivity of the human expert [Liu and Lampinen, 2005]. Rules can only be as effective as the specialised knowledge of the expert.

Zaharie [2002a] concluded that premature convergence of the DE algorithm can be avoided by selecting appropriate values for the scale factor, $\mathcal{F}$, and the crossover factor, Cr, during the optimisation process. These parameters should be selected such that higher levels of divesity are maintained for longer to facilitate better exploration. Consequently, an algorithm that adapts $\mathcal{F}$ and Cr to avoid loss of diversity was developed [Zaharie, 2002b]. This approach introduces a new parameter, ϖ, into the algorithm which determines the convergence rate of the DE algorithm. Values of $\varpi > 1$ increases the diversity of the population, thus reducing the probability of premature convergence. Values of $\varpi < 1$ increases the convergence rate. The problem of tuning the two interrelated parameters $\mathcal{F}$ and Cr is thus reduced to tuning only one parameter, ϖ.

2.4.4.2 Self-Adaptive Control Parameters

Abbass [2002] introduced a self-adaptive crossover operator into the Pareto differential evolution algorithm [Abbass and Sarker, 2002], which uses the DE/rand/1/bin scheme. Each individual, $\vec{x}_i$, in the population stores its own crossover value Cr_i. When individuals, $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$, are selected to create a mutant vector, the crossover value, Cr_{new}, to be used in equation (2.9) is calculated as follows:

$$Cr_{new_i} = Cr_1 + N(0,1) \cdot (Cr_2 - Cr_3) \tag{2.17}$$

where Cr_1, Cr_2 and Cr_3 are the crossover values associated with individuals $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$, respectively. The value of Cr_{new_i} is forced to be within the bounds $[0, 1]$ by making use of a repair rule. The repair rule essentially keeps the decimal part of Cr_{new_i} (i.e. $Cr_{new_i} - \lfloor Cr_{new_i} \rfloor$). A value of 3.2 would thus be changed to 0.2 by the repair rule. Cr_{new_i} is then associated with the newly created offspring individual. The scale factor $\mathcal{F}_{new_i}$, used in equation (2.8), is selected randomly from a Gaussian distribution, $N(0, 1)$, clipped to the range $[0, 1]$.

Self-adaptive differential evolution (SaDE), an approach that adapts the values of $\mathcal{F}$, Cr and the DE scheme, was proposed by Qin and Suganthan [2005]. SaDE adapts the DE scheme by utilising DE/rand/1/bin and DE/best/2/bin with probabilities ρ_1 and ρ_2, respectively. The probabilities, ρ_1 and ρ_2, are assigned initial probabilities of 0.5 each, and subsequently receive values based on the rate at which each has been used to

generate trial vectors that successfully replaced the target vector. DE was found to be less sensitive to the value of the scale factor, and $\mathcal{F}$ is consequently allowed to assume random numbers sampled from $N(0.5, 0.3)$, clipped to the range $(0, 2]$. Crossover probabilities, Cr_i, are associated with each individual, and is originally created from a normal distribution, $N(0.5, 0.1)$. Each Cr_i is randomly perturbed at regular time intervals ($\rho_3 = 5$ generations) from the distribution $N(\rho_4, 0.1)$, where ρ_4 is originally set to 0.5. The algorithm stores all Cr_i values which result in offspring replacing parent individuals. These successful values are averaged and assigned to ρ_4 every $\rho_5 = 25$ generations, whereafter the list of successful values is cleared. Qin and Suganthan [2005] do not explicitly explain how the algorithm treats randomly generated values of Cr_i that are larger than 1.0 or smaller than zero. However, since the background section of [Qin and Suganthan, 2005] states that Cr is a number in the range $[0, 1)$, it is presumed that Cr_i is clipped to the range $[0, 1)$. This algorithm is classified as self-adaptive since only successful values of Cr_i are averaged to replace ρ_4.

A weakness of SaDE is that five potentially problem dependent parameters are introduced: ρ_3, ρ_5, and the original values of ρ_1, ρ_2 and ρ_4. Furthermore, ρ_3 and ρ_5 clearly depend on the population size (since the population size affects the number of function evaluations before Cr is adapted).

The self-adaptive differential evolution (SDE) algorithm [Omran *et al.*, 2005a] uses a technique similar to that of Abbass and Sarker [2002], but in this case the scale factor is adapted rather than the crossover factor. Each individual, $\vec{x}_i$, stores its own scale factor $\mathcal{F}_i$. When individuals, $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$, are selected to create mutant vectors using the DE/rand/1/bin scheme, three extra vectors $\vec{x}_4$, $\vec{x}_5$ and $\vec{x}_6$ are selected from the population in order to create a new scale factor. The scale factor to be used in equation (2.8) is then calculated as follows:

$$\mathcal{F}_{new_i} = \mathcal{F}_4 + N(0, 0.5) \cdot (\mathcal{F}_5 - \mathcal{F}_6) \tag{2.18}$$

where $\mathcal{F}_4$, $\mathcal{F}_5$ and $\mathcal{F}_6$ are the scale factors associated with $\vec{x}_4$, $\vec{x}_5$ and $\vec{x}_6$, respectively. The crossover factor, Cr_{new}, is sampled from a Gaussian distribution $N(0.5, 0.15)$ to create each new offspring individual. It is not clear from [Omran *et al.*, 2005a] how random values of Cr_{new} outside the range $(0, 1)$ are treated. However, since 99.7% of the numbers

sampled from the Gaussian distribution $N(0.5, 0.15)$ falls within the range $(0.05, 0.95)$, it can be assumed that the values are clipped to the range $(0, 1)$. Values outside the range $(0, 1)$ occur extremely infrequently and the assumption should not have a significant impact on the performance of the algorithm. It was shown that SDE produced favourable results when compared to other versions of DE [Salman *et al.*, 2007].

Brest *et al.* [2006] presented an algorithm that self-adapts the values of both the crossover factor and the scale factor. This algorithm is referred to here as jSADE. Each individual, $\vec{x}_i$, stores its own value for the crossover factor, Cr_i, and scale factor, $\mathcal{F}_i$. Before equation (2.8) is used to create a new mutated individual, the scale factor and crossover factor of the target individual ($\vec{x}_i$ in equation (2.9)) are used to create new values for the scale factor and crossover factor to be used in equations (2.8) and (2.9). The new scale and crossover factors are calculated as follows:

$$\mathcal{F}_{new_i} = \begin{cases} \mathcal{F}_l + U(0,1) \cdot \mathcal{F}_u \text{ if } (U(0,1) < \tau_1) \\ \mathcal{F}_i \text{ otherwise} \end{cases} \tag{2.19}$$

$$Cr_{new_i} = \begin{cases} U(0,1) \text{ if } (U(0,1) < \tau_2) \\ Cr_i \text{ otherwise} \end{cases} \tag{2.20}$$

where τ_1 and τ_2 are the probabilities that the factors will be adjusted. Brest *et al.* [2006] used 0.1 for both τ_1 and τ_2. Other values used were $\mathcal{F}_l = 0.1$ and $\mathcal{F}_u = 0.9$. $\mathcal{F}_l$ is a constant introduced to avoid premature convergence by ensuring that the scale factor is never too small (see Section 2.4.3), while $\mathcal{F}_u$ determines the range of scale factors that can be explored by the algorithm.

Brest *et al.* [2006] showed that this algorithm was better or at least comparable to DE and other evolutionary algorithms, including FADE [Liu and Lampinen, 2005], using various benchmark functions. However, a disadvantage of the algorithm is that, while $\mathcal{F}$ and Cr need not be tuned, four new parameters are introduced (τ_1, τ_2, $\mathcal{F}_l$ and $\mathcal{F}_u$). Fine tuning of these parameters may be required for best results on specific problems.

jSADE was extended by Brest *et al.* [2007] to form jSADE2. This algorithm utilises the approach from jSADE to self-adapt the scale and crossover factors for two DE schemes, DE/rand/1/bin and DE/best/1/bin. A pair of scale and crossover values are stored by each individual for each of the schemes. Schemes are selected using the same approach as

SaDE. jSADE2 further differs from jSADE in that the τ_3 worst performing individuals are randomly reinitialised every τ_4 generations. A drawback of jSADE2 is that it introduces two additional parameters (τ_3 and τ_4) to be tuned.

Zhang and Sanderson [2009] proposed JADE, a DE algorithm that self-adapts the scale and crossover factors. JADE employs a new scheme, which is similar to DE/current-to-best/1/bin. The new scheme is called DE/current-to-κbest/1/bin by Zhang and Sanderson [2009] and is given by

$$\vec{v}_i = \vec{x}_i + \mathcal{F}_i \cdot (\vec{x}_{best}^{\kappa_1} - \vec{x}_i) + \mathcal{F}_i \cdot (\vec{x}_1 - \vec{x}_2) \tag{2.21}$$

where $\kappa_1 \in (0, 100]$ and $\vec{x}_{best}^{\kappa_1}$ is randomly selected from the best $\kappa_1\%$ individuals in the the population. The crossover factor for each individual at each generation is calculated as

$$Cr_{new_i} = N(\mu_{Cr}, 0.1) \tag{2.22}$$

where μ_{Cr} (which is originally set to $\kappa_2 = 0.5$) is updated at the end of each generation using

$$\mu_{Cr,new} = (1 - \kappa_3)\mu_{Cr} + \kappa_3 \frac{\sum_{Cr_O \in \mathcal{O}_{Cr}} Cr_O}{|\mathcal{O}_{Cr}|} \tag{2.23}$$

where $\mathcal{O}_{Cr}$ is the set of all successful crossover factors (i.e. values of Cr that resulted in trial vectors that replaced parent vectors), and $\kappa_3 \in [0, 1]$ is a parameter to the algorithm. The value of $\mu_{Cr,new}$ is thus calculated as the weighted average of the current value and the mean of all previous successful crossover factors. Cr_{new_i} is truncated to the range $[0, 1]$.

JADE calculates the scale factor for each individual at each generation as

$$\mathcal{F}_{new_i} = C(\mu_{\mathcal{F}}, 0.1) \tag{2.24}$$

where $\mu_{\mathcal{F}}$ (which is originally set to $\kappa_4 = 0.5$) is updated at the end of each generation using

$$\mu_{\mathcal{F},new} = (1 - \kappa_3)\mu_{\mathcal{F}} + \kappa_3 \frac{\sum_{\mathcal{F}_O \in \mathcal{O}_{\mathcal{F}}} \mathcal{F}_O^2}{\sum_{\mathcal{F}_O \in \mathcal{O}_{\mathcal{F}}} \mathcal{F}_O} \tag{2.25}$$

where $\mathcal{O}_{\mathcal{F}}$ is the set of all successful scale factors. Equation (2.25) uses the Lehmer mean to place more emphasis on larger scale factors, while equation (2.24) uses a Cauchy distribution to encourage more diverse values of $\mathcal{F}_{new_i}$. $\mathcal{F}_{new_i}$ is truncated to the range $[0, 1]$.

JADE also contains an optional archive component which was found to be beneficial for high dimensional problems. The archive population, which is initially empty, is grown by adding trial vectors that do not replace parent vectors in the selection step of the DE algorithm. Individuals are randomly removed from the archive when the size of the archive exceeds the size of the normal DE population. The archive is utilised by the DE algorithm by randomly selecting individuals for mutation from the union of the archive and normal DE populations.

Zhang and Sanderson [2009] report better results for JADE in comparison with SaDE and jSADE on several benchmark problems. A disadvantage of the JADE algorithm is that four new parameters are introduced (κ_1, κ_2, κ_3 and κ_4). The authors do, however, argue that JADE is relatively insensitive to these parameters, and that JADE works effectively on a large range of values of the parameters.

The preceding approaches that were discussed all focused on adapting the scale and crossover factors. An algorithm that self-adapts the population size was presented by Teo [2006]. In this algorithm, each individual encodes a value for population size which is evolved along with the individual's position. The average of all the individual population size values is then used to calculate the actual population size for the next generation. When the population size is increased, copies of the best individual in the current population are inserted (in addition to all the individuals in the current population) into the new population until the new population has grown to the appropriate size. In cases where the population size is reduced, the weakest individuals in the current populations are not copied to the population of the next generation.

Brest *et al.* [2008] also adapted the population size with jSADE3, an extension to jSADE. jSADE3 halves the population size every τ_5 function evaluations. The fitness of each individual in the first half of the population is compared to the fitness of the corresponding individual in the second half of the population (i.e. the first individual in the first half is compared to the first individual in the second half, etc.). After each

comparison, the fittest individual is placed in the new population. The old population is discarded after the comparison process and evolution continues using the new, smaller, population. jSADE3 does not provide a mechanism to increase the population size.

jSADE3 further differs from jSADE in that the sign of the scale factor is changed, with a probability of $\tau_6 = 0.75$, if $F(\vec{x}_2) > F(\vec{x}_3)$ (assuming a function minimisation problem), where $\vec{x}_2$ and $\vec{x}_3$ are the randomly selected difference vectors used in the DE/rand/1/scheme. The difference vector thus points towards the individual with the lowest error value after changing the sign. Note that neither of the changes made to jSADE to form jSADE3 can be classified as "self-adaptive". jSADE3 was, however, included in this section because of the self-adaptive components inherited from jSADE.

A significant disadvantage of the jSADE3 algorithm is that the number of function evaluations that are used during the optimisation process must be known in advance in order to set parameter τ_5. jSADE3 could potentially reduce the population size to the point where it is impossible to execute the DE algorithm (i.e. the population could become too small to select unique individuals for the trial-vector creation step) if τ_5 is set too low.

2.4.4.3 Eliminating Control Parameters

Three of the the self-adaptive approaches described in the previous section eliminated the need to fine-tune control parameters by sampling their values from a distribution. The approaches of Abbass and Sarker [2002] and Qin and Suganthan [2005] used random values for the scale factor, while Omran *et al.* [2005a] used random values for the crossover factor. These parameters were thus not self-adapted. The randomisation of control parameters to eliminate the need for fine-tuning is a common approach, and has been used by several researchers [Price *et al.*, 2005] [Das *et al.*, 2005].

Several researchers have investigated DE and PSO hybrids [Hendtlass, 2001], [Zhang and Xie, 2003]. Omran *et al.* [2009] presented a self-adaptive hybrid PSO and DE algorithm, called the Barebones DE, as a means of eliminating DE control parameters. Three significant changes were made to the standard DE algorithm. Firstly, each individual is given a memory that stores the best location that the individual has occupied during the execution of the algorithm. This is called the *personal best* value and is labelled $\vec{y}_i$ for individual $\vec{x}_i$. The fittest *personal best* value within the entire population is called the

global best and is labelled $\vec{\hat{y}}$. In the second place, the competition between parents and offspring of DE is eliminated and offspring are placed directly into the main population, even if they are less fit than the original individual. Good solutions are not lost, however, since these solutions will be stored in the *personal best* value. The third change is in how offspring are created. The trial vector is calculated as:

$$v_{i,j} = r_{j_1} \cdot y_{i,j} + (1 - r_{j_1}) \cdot \hat{y}_j + r_{j_2} \cdot (x_{1,j} - x_{2,j}) \tag{2.26}$$

where r_{j_1},$r_{j_2} \sim U(0,1)$. The trial vector is thus computed as the weighted average of the current individual's personal best and the global best. The choice for trial vector is motivated by studies that showed that PSO particles converge to the weighted average of their personal and neighbourhood best positions [van den Bergh and Engelbrecht, 2006][Trelea, 2003][Clerc and Kennedy, 2002]. Note that the scale factor has been eliminated.

The crossover step (see equation (2.9)) is replaced by:

$$x_{i,j} = \begin{cases} v_{i,j} \text{ if } (U(0,1) \leq Cr) \\ y_{3,j} \text{ otherwise} \end{cases} \tag{2.27}$$

Crossover is thus performed with the *personal best* of one of the randomly selected parent individuals. Note that the crossover factor has not been eliminated.

Omran *et al.* [2009] compared Barebones DE to DE, PSO and Barebones PSO. It was concluded that the Barebones DE is more noise tolerant than the other approaches and that it gave superior performance on high-dimensional problems.

2.4.4.4 Summary

The majority of approaches discussed in this section are aimed at adapting, or eliminating the need to tune the scale and crossover factors. The purely adaptive approaches removed the need to tune both the scale and crossover factors, but in the case of the approach of Zaharie [2002b], a new parameter was introduced which must be fine-tuned. In the case of FADE, the approach of Liu and Lampinen [2005], the fuzzy rules must be manually created, which is arguably as onerous as fine-tuning control parameters and subjective.

An advantage of the self-adaptive approaches described in this section is that they are significantly simpler than the adaptive approaches that were discussed. However, only two

of the self-adaptive approaches, [Zhang and Sanderson, 2009][Brest *et al.*, 2006], applies self-adaption to both the scale and crossover factors, but at the expense of introducing four new parameters into the algorithm. The other self-adaptive approaches only self-adapt one of the two factors, while the need to tune the other factor is eliminated by repeatedly selecting it from a normal distribution.

The hybrid DE and PSO approach of Omran *et al.* [2009] completely removes the scale factor from the algorithm, but retains the crossover factor.

2.4.5 Differential Evolution Applications

Differential evolution has been applied to a wide range of problems, including image analysis [Li *et al.*, 2003][Xu and Dony, 2004][Omran *et al.*, 2005b], neural network training [Chen *et al.*, 2002][Magoulas *et al.*, 2004][Moalla *et al.*, 2002], scheduling [Lin *et al.*, 2000][Rae and Parameswaran, 1998][Rzadca and Seredynski, 2005], and controllers [Chang and Du, 2000][Chiou and Wang, 1998][Cruz *et al.*, 2003]. This thesis investigates the use of DE in dynamic environments.

2.5 Dynamic Environments

This section commences with a formal definition of dynamic environments in Section 2.5.1. Section 2.5.2 provides a discussion of the different types of dynamic environments and the factors that influence an algorithm's ability to effectively locate optima in a dynamic environment. The moving peaks benchmark and the generalised dynamic benchmark generator, two benchmarks used to simulate dynamic environments, are discussed in Sections 2.5.3 and 2.5.4 respectively. Various performance measures have been suggested for use in dynamic optimisation problems, which are described in Section 2.5.5.

2.5.1 Formal Definition

The solution to real-world optimisation problems often vary over time. Consider, for example, the optimal air-fuel mixture of a aircraft during flight. The optimal mixture at any point in time depends on the prevailing winds, the altitude and the speed of the aircraft, and also varies due to the change in mass caused by burning fuel.

A dynamic environment, in the context of optimisation problems, is a fitness landscape that varies over time. Formally, $\exists\ t, t' \in \mathcal{T}$, where $\mathcal{T}$ is the set of all time steps, and $\exists\ \vec{x} \in \mathcal{S}^{n_d} \subseteq \mathbb{R}^{n_d}$ such that

$$F(\vec{x}, t) \neq F(\vec{x}, t') \tag{2.28}$$

where F is a dynamic function. Consequently, the optima in the fitness landscape may vary in number, location, and function value over time. The objective of an algorithm applied to a dynamic optimisation problem (DOP) is to find the best solutions at all time steps during the optimisation process [Weicker, 2002], i.e. for all $t \in \mathcal{T}$, assuming a minimisation problem, find

$$\vec{x}^*(t) : F(\vec{x}^*(t), t) \leq F(\vec{x}, t)\ \forall\ \vec{x} \in \mathcal{S}^{n_d} \tag{2.29}$$

where $\vec{x}^*(t) \in \mathcal{S}^{n_d}$ is the location of a global optimum at time step t.

The above discussion gives a broad definition of dynamic environments. The following section explores various types of dynamic environments and the characteristics of each.

2.5.2 Types of Dynamic Environments

Several classifications of dynamic environments based on different characteristics are available in the literature [Angeline, 1997][Trojanowski and Michalewicz, 1999a][Branke, 2002][Li *et al.*, 2008][Hu and Eberhart, 2001][De Jong, 1999]. This section endeavors to synthesise disparate classification approaches into a single unified classification scheme by which different types of dynamic environments can be identified.

The three mayor considerations to be taken into account when classifying dynamic environments are the fitness landscape composition, the types of changes and the pervasiveness of changes. An individual discussion of these considerations follows:

1. **Fitness landscape composition**: Changes in a dynamic environment depend in a large extend on the underlying composition of the fitness landscape. The composition of a search space can be classified as either homogeneous or heterogeneous:

   (a) A homogeneous fitness landscape is made up of a single underlying function. Such a fitness landscape was suggested by Angeline [1997]. The underlying function is moved with respect to the coordinate system of the search space to

create a change in the environment. The number of optima and the function value of each optimum thus depends purely on the underlying function that is used and does not change over time.

(b) A heterogeneous fitness landscape contain multiple underlying functions. Two ways of how underlying functions are combined have been suggested:

i. At each point in the fitness landscape the sum of the function values of all underlying functions is taken [Li *et al.*, 2008], i.e. $F(\vec{x}, t) = \sum_{p=1,\ldots,n_p} f_p(\vec{x})$, where f_p is one of the n_p underlying functions. As the underlying functions are moved with respect to the coordinate system of the search space, optima in the fitness landscape are formed at locations where the optima of the underlying functions correspond, analogous to constructive interference in wave theory [Feynman *et al.*, 1963]. The number of optima in the fitness landscape and their respective function values thus not only depend on the number of optima and function values of the underlying functions, but also on the relative position and orientation of the underlying functions with respect to the coordinate system of the search space.

ii. At each point in the fitness landscape the maximum function value of all underlying functions is taken for function maximisation problems (i.e. $F(\vec{x}, t) = \max_{p=1,\ldots,n_p}\{f_p(\vec{x})\}$, where f_p is one of the n_p underlying functions) and the minimum is taken for function minimisation problems (i.e. $F(\vec{x}, t) = \min_{p=1,\ldots,n_p}\{f_p(\vec{x})\}$) [Branke, 2002]. The effect of this approach is that some optima may be obscured by other optima, thus varying the number of optima in the fitness landscape over time. The function value of the global optimum, however, depends purely on the function value of the most optimal underlying function.

2. **Change types**: The nature of changes in a dynamic environment influence approaches that can be followed by algorithms to recover from changes. The three broad change types that have been identified:

(a) Random changes imply that a particular change in the environment does not depend on a previous change [Trojanowski and Michalewicz, 1999a], i.e. no

pattern governing changes exist.

(b) Chaotic changes refer to changes where a dependency, or pattern, between successive changes exists. This is in contrast to random changes where each change is completely independent of the previous. According to Trojanowski and Michalewicz [1999a], the dependency between changes may be so complex that changes appear random and may thus not be predictable. On the other hand, if the relationship between successive changes is simple enough, changes may be considered predictable, and could be exploited by an optimisation algorithm. Angeline [1997] identified two types of predictable changes:

i. Linear, where successive changes are linearly correlated.

ii. Cyclic (or recurrent), where the dynamic environment periodically returns to the same state.

(c) Combinations of chaotic and random changes are changes where some relationship between successive changes exist, but which also contain a random component [Li *et al.*, 2008][Branke, 2002].

3. **Change Pervasiveness**: This metric classifies dynamic environments on the attributes of the environment that are modified. Hu and Eberhart [2001] identified three classifications based on properties of a dynamic environment that can change, to which a fourth category can be added:

(a) The locations of optima in the environment change (referred to by Hu and Eberhart [2001] as Type I environments).

(b) The locations of the optima remain the same, but the values of the optima change (referred to by Hu and Eberhart [2001] as Type II environments). A consequence of this is that different optima can become the global optimum over time.

(c) The locations and the values of the optima change (referred to by Hu and Eberhart [2001] as Type III environments).

(d) A classification not included in the categorisation of Hu and Eberhart [2001] is represented by environments in which new optima are explicitly introduced or

old optima are removed, i.e. the number of optima does not remain constant.

The above taxonomy (i.e. fitness landscape composition, change type and change pervasiveness) can be used to classify a dynamic environment, but does not encompass all the physical characteristics that describes a dynamic environment. The type of dynamic environment also depends on factors that influence an optimisation algorithm's ability to successfully locate optima.

It is argued here that the three main factors that combine to influence an optimisation algorithm's ability to locate optima successfully in a dynamic environment are the hardness of the fitness landscape, the frequency at which changes occur, and the severity of changes to the environment. The the three factors are respectively described below:

Hardness of the fitness landscape: Hardness is a term that is used in this thesis to describe the intuitive concept of the difficulty involved in optimising an objective function by an optimisation algorithm. Intuitively, an easy optimisation problem implies that a specific optimisation algorithm is likely to successfully locate a global optimum within relatively few iterations. Conversely, a hard optimisation problem implies that the algorithm typically requires a relatively large number of iterations to locate the global optimum, or that the algorithm lacks the ability to locate the global optimum. This implies that the algorithm is likely to either converge to an abitrary point or to a local optimum in the fitness landscape of a hard problem. Note that the No-Free-Lunch theorems [Wolpert and Macready, 1997] imply that not all optimisation problems are equally hard for all optimisation algorithms.

Several factors can influence the hardness of a fitness landscape, for example, the modality of the fitness landscape (i.e. the number of optima), the gradient at various points in the fitness landscape and the presence of ridges, valleys and plateaus. In addition, the number of dimensions of a search space usually influences the hardness, since increasing the number of dimensions increases the size of the search space. Note that there are exceptions to this, such as the Griewank function, which has been shown to become easier as the dimension increases [Locatelli, 2003].

Scientists have endeavored to quantify the hardness of fitness landscapes by means of metrics like ruggedness [Malan and Engelbrecht, 2010], fitness distance correlation

[Jones and Forrest, 1995], and the dispersion metric [Lunacek and Whitley, 2006]. To date, no single satisfactory problem hardness measure has been found [He *et al.*, 2007][Malan and Engelbrecht, 2009].

An optimisation algorithm has a limited number of iterations within which to locate a global optimum before a change in a dynamic environment occurs. The quality of the solution found by the optimisation algorithm is thus directly influenced by the hardness of the search landscape.

Frequency of changes: Changes in a dynamic environment can either be at discrete time intervals (the environment is static for periods between changes) or continuous (the environment changes every time it is sampled) [Trojanowski and Michalewicz, 1999a]. As the frequency of changes tends to infinity, changes intervals tend from discrete to continuous. As the frequency of changes tend towards zero, the dynamic environment tends towards a static environment.

Assume that a particular optimisation algorithm has the ability to locate a global optimum in specific static fitness landscape. This assumption does not imply that the algorithm will locate a global optimum in a dynamic version of the same fitness landscape, as the frequency of changes to the fitness landscape will influence the likelihood of locating a global optimum. For example, in situations where changes are highly infrequent (i.e. many iterations can be performed before a change occurs), the optimisation algorithm has more time to locate optima and the likelihood of locating a global optimum is thus increased. However, if changes are frequent, the optimisation algorithm will be able to perform fewer iterations between changes, and the chances of locating optima are reduced. Performance is consequently adversely affected by frequent changes as algorithms must locate optima in fewer iterations.

Severity of changes: Changes in a dynamic environment change the fitness landscape surface. The degree of correlation between the fitness landscape before a change and after a change depends on the severity of the change.

Factors that influence the severity of changes in a dynamic environment include: the amount by which the locations of optima are changed, how much the values of the

optima change, and whether new optima are introduced or old optima are removed [De Jong, 1999]. If the changes in the environment are so severe that the fitness landscape after the change bears no relation to the search space before the change, an optimisation algorithm cannot exploit any information collected regarding the fitness landscape before the change, and optimisation might as well resume from scratch [Branke *et al.*, 2000].

Conversely, very small changes in the environment, which result in the location of an optimum remaining close to its old position, makes it possible for an optimisation algorithm to exploit the information regarding the old position of the optimum in order to find the new position. The severity of changes thus determines how much useful information regarding the shape of the search space can be transferred from before a change in the environment to after the change. A large amount of useful information can be exploited by an optimisation algorithm to recover quickly from a change in the environment, while if only a small amount of useful information can be exploited, the recovery rate of the algorithm is likely to be reduced.

The interaction between change frequency and change severity has a considerable impact on whether it is feasible for an optimisation algorithm to optimise a dynamic environment. Figure 2.1 illustrates how the change frequency and change severity impact on a feasible region (where it is possible to solve the problem) which exists for any algorithm on any dynamic optimisation problem. The feasible region corresponds to either changes that are small enough or changes that are infrequent enough (or both), so that an optimisation algorithm is able to function adequately.

A high change severity, even in the presence of a low change frequency, results in successive environments that are too uncorrelated for any useful information from before the change to be transferred to after the change. Similarly, if the change frequency is too high in the presence of even small changes, the optimisation algorithm has too little time to recover from changes. There thus exist upper bounds on the frequency of changes and the severity of changes beyond which it is no longer viable to perform optimisation successfully. Figure 2.1 is shown to illustrate to the reader the general case, but similar curves can be drawn for all optimisation algorithms for dynamic environments, although

the gradient of the curve may differ from algorithm to algorithm.

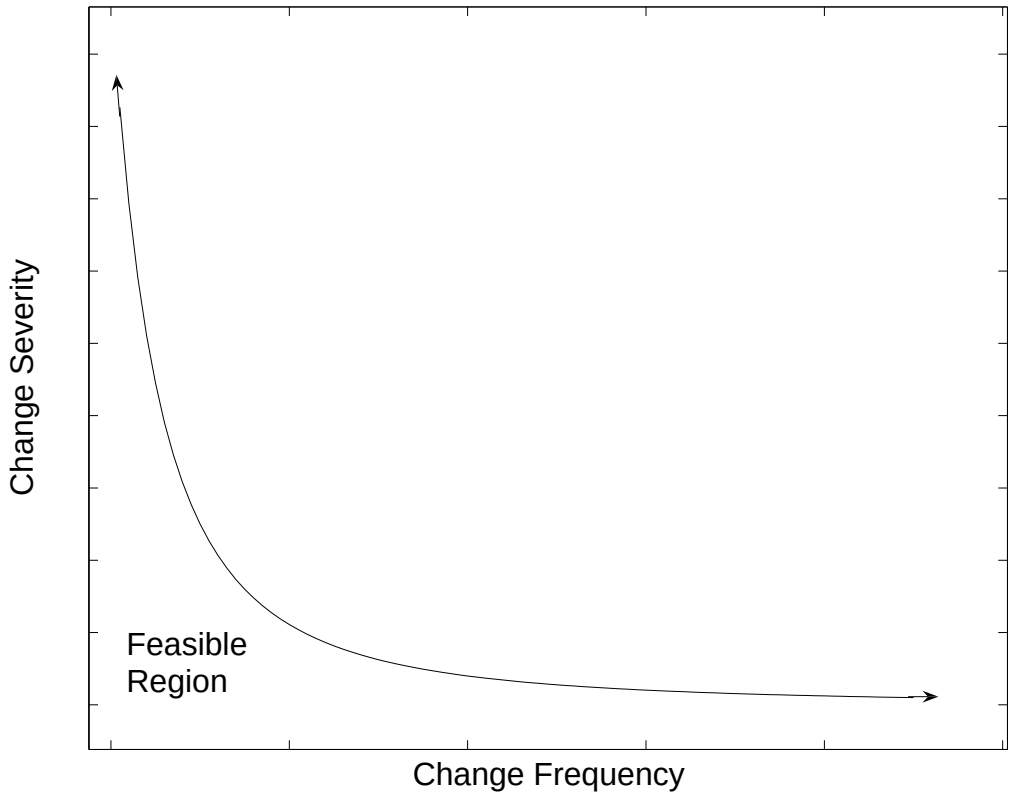

Figure 2.1: Feasible optimisation region in terms of change frequency and severity

This section described various types of dynamic environments. An investigation into the effectiveness of optimisation algorithms aimed at dynamic environments necessitates the evaluation of the optimisation algorithms on a wide range of environment types. The following two sections describe two benchmarks that are used in this study to simulate several types of dynamic environments discussed in this section.

2.5.3 Moving Peaks Benchmark

Branke [2007] created the moving peaks benchmark (MPB) to address the need for a single, adaptable benchmark that can be used to compare the performance of algorithms aimed at

dynamic optimisation problems. The benchmark consists of a moving peaks maximisation function to test the effectiveness of an algorithm. The multi-dimensional problem space of the moving peaks function contains several peaks, or optima, of variable height, width, and shape. These peaks move around with height and width changing periodically.

For n_p peaks in n_d dimensions, the moving peaks function is given by [Branke, 2002]:

$$F(\vec{x}, t) = \max\left\{B(\vec{x}), \max_{p=1,\ldots,n_p}\{f_p(\vec{x}, h_p(t), w_p(t), \vec{l_p}(t))\}\right\} \tag{2.30}$$

where $B(\vec{x})$ is a constant basis landscape and f is a peak shape function. $h_p(t)$, $w_p(t)$ and $\vec{l_p}(t)$ are the height, width and position of the respective peaks, given by

$$h_p(t) = h_p(t-1) + height\_severity \cdot N(0,1) \tag{2.31}$$

$$w_p(t) = w_p(t-1) + width\_severity \cdot N(0,1) \tag{2.32}$$

$$\vec{l_p}(t) = \vec{l_p}(t-1) + \vec{c_p}(t) \tag{2.33}$$

where *height_severity* and *width_severity* are the deviation of changes to the width and height, respectively, and $\vec{c_p}(t)$ is given by

$$\vec{c_p}(t) = \frac{C_s((1-\lambda)\vec{r} + \lambda\vec{c_p}(t-1))}{|\vec{r} + \vec{c_p}(t-1)|} \tag{2.34}$$

where C_s is the change severity, $\vec{r}$ is a random vector created by selecting uniform random numbers for each dimension, $r_j \sim U(-0.5, 0.5)$, and normalising the vector length to C_s, and $\lambda \in (0, 1)$ is the degree of correlation to previous shifts. For $\lambda = 1$ the direction of the positional shift is equivalent to the direction of the previous shift, while for $\lambda = 0$ the positional shift is not correlated to the previous shift.

The MPB allows the following parameters to be set:

- Number of peaks
- Number of dimensions
- Maximum and minimum peak width
- Maximum and minimum peak height
- Change period (the number of function evaluations between successive changes in the environment)

- Change severity (how much the locations of peaks are moved within the fitness landscape)
- Height severity (standard deviation of changes made to the height of each peak)
- Width severity (standard deviation of changes made to the width of each peak and consequently the gradient of each peak)
- Peak function
- Correlation (between successive movements of a peak)

A static basis function, $B(\vec{x})$, can optionally be added to the problem space if a more complex fitness landscape is required. The fitness landscape of the MPB is fundamentally a heterogeneous fitness landscape (refer to point 1 in Section 2.5.2) although a homogeneous fitness landscape can be simulated by setting the number of peaks to one.

Type I, II and III environments (see point 3 in Section 2.5.2) can be simulated by changing the parameters for change and height severity. Three scenarios of settings of MPB parameters were suggested by Branke [2007]. However, the majority of researchers using the MPB employ only variations of Scenario 2 settings, as these settings were used in an early paper by [Branke and Schmeck, 2003] and were subsequently used by other researchers to facilitate comparisons between algorithms. The Scenario 2 settings are listed in Table 2.1.

A graphical depiction of the moving peaks function in two dimensions with a conical peak function is given in Figure 2.2, and with a spherical peak function in Figure 2.3.

The value of the conical peak function with location $\vec{l}$ and height h is calculated in n_d dimensions as

$$f_{conical}(\vec{x}) = h - w \cdot \sqrt{\sum_{j=1}^{n_d}(x_j - l_j)^2} \tag{2.35}$$

while the value of the spherical peak function is calculated as

$$f_{spherical}(\vec{x}) = h - \sum_{j=1}^{n_d}(x_j - l_j)^2 \tag{2.36}$$

The most apparent difference between the conical and spherical functions is that, because spherical peaks do not have a constant gradient like the conical peaks, the absolute

Table 2.1: MPB Scenario 2 settings

Setting	**Value**
Number of Dimensions	5
Number of Peaks	10
Max and Min Peak height	[30,70]
Max and Min Peak width	[1.0,12.0]
Change period	5000
Change severity	1.0
Height severity	7.0
Width severity	1.0
Peak function	Cone
Correlation	[0.0,1.0]

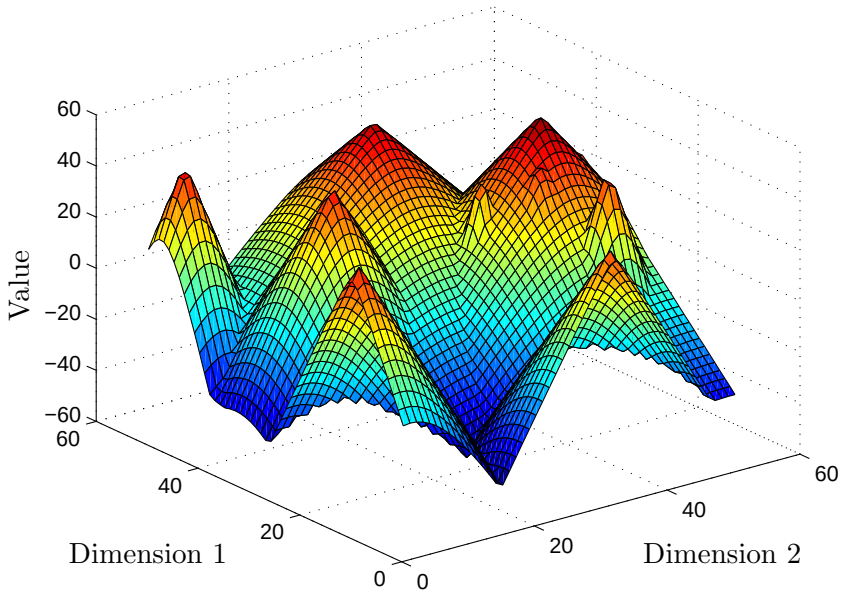

Figure 2.2: The moving peaks function using a conical peak function in two dimensions

minimum of the moving peaks function with spherical peaks is much lower than that of the conical peaks. It is thus possible to find much greater errors when using spherical peaks compared to using conical peaks. On the other hand, the steeper slopes on the spherical

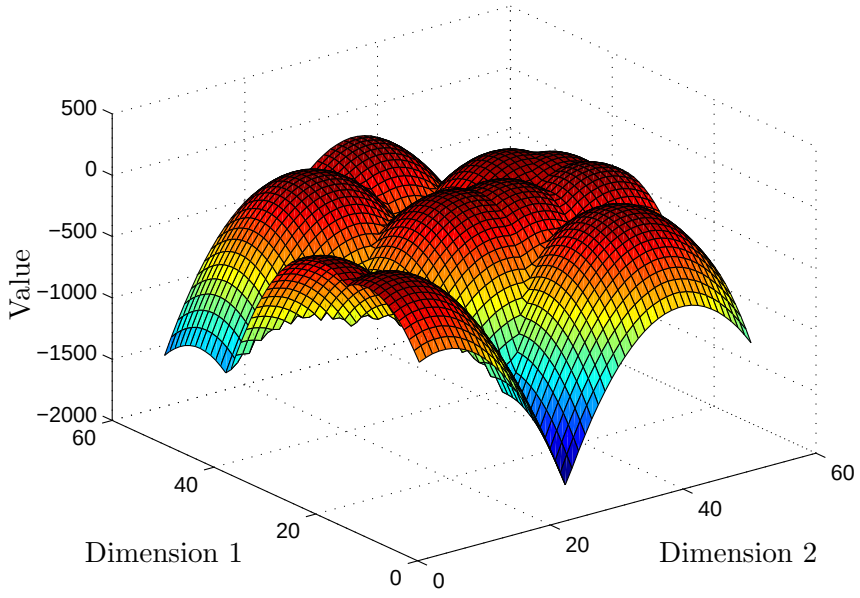

Figure 2.3: The moving peaks function using a spherical peak function in two dimensions

peaks should make initial optimisation easier, but only until a point close to the absolute minimum (where the gradient is small), where optimisation should become harder.

2.5.3.1 Extensions to the Moving Peaks Benchmark

Part of this study involves investigating dynamic environments in which the number of peaks fluctuates over time (see point 3 in Section 2.5.2). The MPB was therefore adapted in this thesis to allow the number of peaks to change when a change in the environment occurs. For the adapted MPB, the number of peaks, $n_p(t)$, is calculated as:

$$n_p(t) = \begin{cases} \max\{1,\ n_p(t-1) - M_{n_p} \cdot U(0,1) \cdot M_c\} \\ \qquad \text{if } U(0,1) < 0.5 \\ \min\{M_{n_p},\ n_p(t-1) + M_{n_p} \cdot U(0,1) \cdot M_c\} \\ \qquad \text{otherwise} \end{cases} \tag{2.37}$$

where M_{n_p} is the maximum number of peaks and M_c is the maximum fraction of M_{n_p} that can be added or removed from the population after a change in the environment. M_c thus controls the severity of the change in the number of peaks. For example, $M_c = 1$ will

result in up to M_{n_p} peaks being added or removed, while $M_c = 0.1$ will result in a change of up to 10% of M_{n_p}. M_{n_p} and M_c are included as parameters to the benchmark function.

2.5.3.2 Moving Peaks Benchmark Critical Discussion

The MPB has been used by several researchers [Trojanowski, 2007] [Janson and Middendorf, 2003][Moser and Hendtlass, 2007][Ayvaz *et al.*, 2011][Kiraz *et al.*, 2011] and has served as a valuable method of comparison between different algorithms. The simplicity of the method used to produce changes in the dynamic environment makes the MPB ideal for studying the scalability of an optimisation algorithm in terms of change severity. Unfortunately, apart from three broad scenarios, the benchmark does not contain general guidelines specifying a set of parameters to investigate. As a result, researchers often report on different sets of parameters which complicates comparisons among algorithms.

Another disadvantage of the MPB is that it only provides a limited number of peak functions. Benchmarks for static environments typically contain large sets of complex functions, for example, the Griewank, Ackley, Rastrigin and Weierstrass functions [Qu and Suganthan, 2010]. The MPB does not provide a platform on which to identify optimisation algorithms suited to specific underlying functions that are typically used as benchmarks in static environments. Furthermore, the MPB does not provide functionality to investigate the effect of a broad range of change types (refer to point 2 discussed in Section 2.5.2), as only random and linearly correlated change types are supported.

The MPB does not provide functionality to investigate dynamic environments where optima are explicitly introduced or removed from the dynamic environment. This limitation has been remedied by the extension made to the MPB which is described in Section 2.5.3.1.

2.5.4 Generalised Dynamic Benchmark Generator

The generalised dynamic benchmark generator (GDBG) of Li *et al.* [2008] [Li and Yang, 2008] is a recent benchmark created to investigate optimisation in dynamic environments. This benchmark was used during the special session on evolutionary computation in dynamic and uncertain environments at the 2009 IEEE Congress on Evolutionary Computation to compare the performance of algorithms submitted by several researchers [Korošec

and Šilc, 2009, Brest *et al.*, 2009, De França and Von Zuben, 2009, Li and Yang, 2009].

The GDBG dynamic function is defined as

$$F = f(\vec{x}, \vec{\phi}, t) \tag{2.38}$$

where $\vec{\phi}$ represents the system control parameters for height ($\vec{h}$), width ($\vec{w}$), location of the global optimum ($\vec{l}$, assuming a single global optimum), and underlying function rotation ($\vec{\theta}$). The notation ϕ_Ω with $\Omega \in \{\vec{h}, \vec{w}, \vec{l}, \vec{\theta}\}$ is used to refer to the control parameters in general. Minimum ($\phi_{\Omega,min}$), maximum ($\phi_{\Omega,max}$), and severity ($\phi_{\Omega,sev}$) values are defined by the creators of the benchmark for each control parameter.

Changes in the environment are enacted by periodically changing the control parameters. A function, *DynamicChanges*, provides the next values of each control parameter vector:

$$\vec{\phi}_\Omega(t+1) = DynamicChanges(\vec{\phi}_\Omega(t)) \quad \text{with } \Omega \in \{\vec{h}, \vec{w}, \vec{l}, \vec{\theta}\} \tag{2.39}$$

DynamicChanges takes the form of various change types, presented in Section 2.5.4.2.

2.5.4.1 Generalised Benchmark Generator Functions

The benchmark consists of six main functions. The first, F_1, is a peak function with similarities to the MPB of Branke. The main differences between these two benchmarks are the peak shapes and the way that peaks are moved. The peak function, F_1, at time t is given by:

$$F_1(\vec{x}, t) = \max_{p=1,\ldots,n_p} \left\{ h_p / \left(1 + w_p \cdot \sqrt{\sum_{j=1}^{n_d} \frac{(x_j - l_{p,j})^2}{n_d}} \right) \right\} \tag{2.40}$$

where h_p, w_p and $\vec{l_p}$ contain the values for height, width and position of the n_p peaks in n_d dimensions. Equation (2.40) results in a peak which height decreases with the square of the distance to the optimum of the peak.

The method used to randomly move peaks around in the MPB results in peaks being reflected when the boundary of the fitness landscape is reached, analogous to a ball bouncing off a wall. This is seen as a disadvantage, since it implies an unequal challenge per change for an optimisation algorithm [Li *et al.*, 2008]. The disadvantage is remedied in the GDBG by using a rotation matrix to affect positional changes in the environment

by independently rotating the dimensional components of the positions of the peaks as described in Algorithm 5 [Li and Yang, 2008].

Algorithm 5: Position rotation algorithm for the p-th underlying function

Randomly select $s = 2 \cdot \lfloor n_d/2 \rfloor$ dimensions (s is the largest even integer smaller or equal to n_d) from the n_d dimensions to compose a vector $\vec{o} = [o_1, o_2, ..., o_s]$;

Let $\vec{\theta}_p(t) = DynamicChanges(\vec{\theta}_p(t-1))$;

foreach *pair of dimensions,* (o_j, o_{j+1}) **do**

  construct a rotation matrix using $\vec{\theta}_p(t)$ as follows: $\mathbf{R}_{o_j,o_{j+1}}(\theta_{p,(j+1)/2})(t))$,

end

A transformation matrix $\mathbf{T}(t)$ is obtained by:

$\mathbf{T}_p(t) = \mathbf{R}_{o_1,o_2}(\theta_{p,1}(t)) \cdot \mathbf{R}_{o_3,o_4}(\theta_{p,2}(t)) \cdots \mathbf{R}_{o_{s-1},o_s}(\theta_{p,s/2}(t))$;

$\vec{l}_p(t+1) = \vec{l}_p(t) \cdot \mathbf{T}_p(t)$;

For positional changes, the values $\phi_{\theta,sev} = 1$, $\phi_{\theta,max} = \pi$, and $\phi_{\theta,min} = -\pi$ are used in *DynamicChanges*.

The vectors containing height and width information of the n_p peaks ($\vec{h} = [h_1, \ldots, h_{n_p}]$ and $\vec{w} = [w_1, \ldots, w_{n_p}]$) are changed using the *DynamicChanges* function:

$$\vec{h}(t+1) = DynamicChanges(\vec{h}(t)) \quad (2.41)$$

$$\vec{w}(t+1) = DynamicChanges(\vec{w}(t)) \quad (2.42)$$

Settings associated with F_1 are:

- The number of peaks: $n_p = 10, 50$
- Width range: $\phi_{w,max} = 10$ and $\phi_{w,min} = 1$
- Width severity: $\phi_{w,sev} = 0.5$
- Initial width: $initial\ width = 5$

A graphical depiction of a two-dimensional instance of F_1 is given in Figure 2.4. F_1 is a multi-modal maximisation problem with n_p optima (although some optima may be obscured by others).

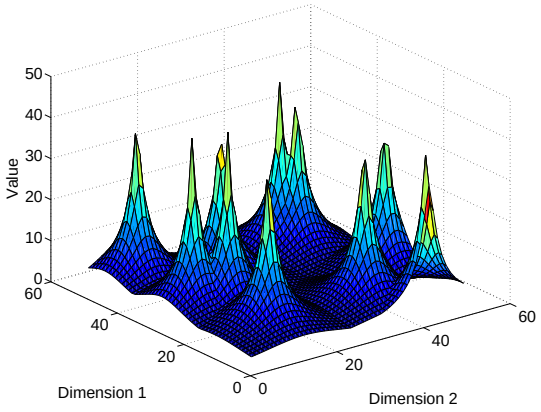

Figure 2.4: F_1 from the GDBG

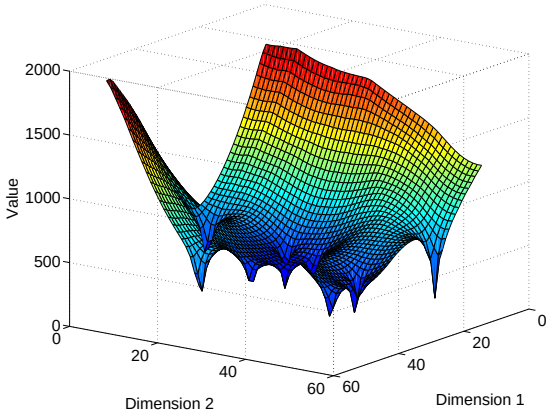

Figure 2.5: F_2 from the GDBG

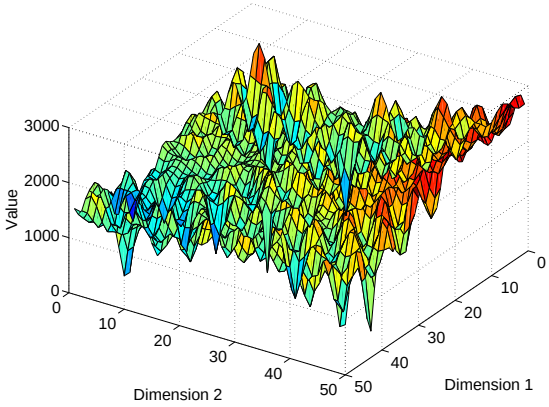

Figure 2.6: F_3 from the GDBG

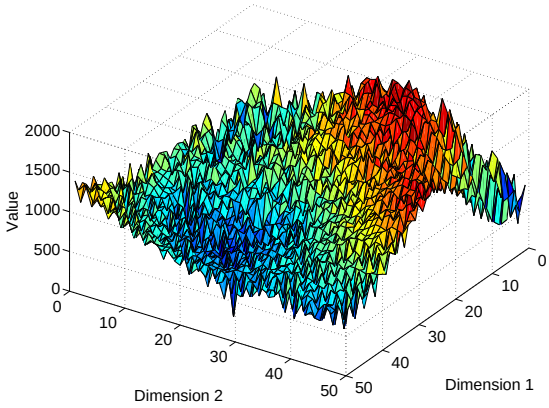

Figure 2.7: F_4 from the GDBG

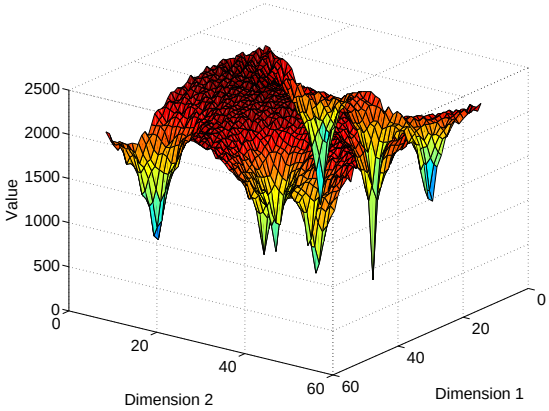

Figure 2.8: F_5 from the GDBG

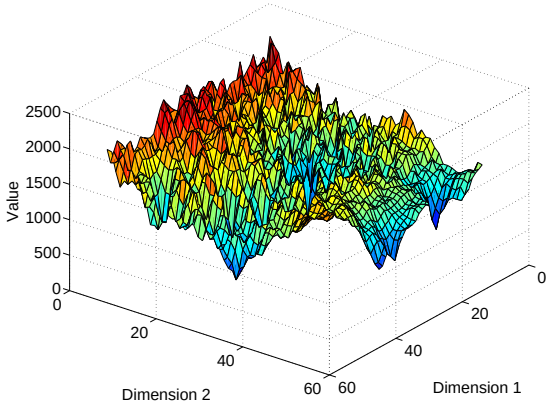

Figure 2.9: F_6 from the GDBG

Functions F_2 through F_6 are composition functions made up from basic functions standardly used as static benchmarks in the field of evolutionary computing. Each function is constructed from a composition of 10 underlying functions with different orientations, whose height and position are changed individually, resulting in a complex dynamic environment. The composite function values are calculated as:

$$F(\vec{x},t)=\sum_{p=1}^{10}\left(\psi_p\cdot\left(2000\cdot\frac{f_p\Big(\big(\vec{x}-\vec{l}_p(t)\big)/\sigma_{f_p}\cdot\mathbf{T}_p\Big)}{f_{p,opti}}+h_p(t)\right)\right) \tag{2.43}$$

where $\vec{l}_p(t)$ is the location of the global optimum (with value $f_{p,opti}$) of underlying function f_p, and $\mathbf{T}_p$ is the rotation matrix for each underlying function. $\mathbf{T}_p$ is generated once for each underlying function and is not changed thereafter. The value of ψ_p is calculated for each f_p in three steps:

1.
$$\psi_p=\exp\left(-\sqrt{\frac{\sum_{j=1}^{n_d}(x_j-l_{p,j})^2}{2n_d}}\right) \tag{2.44}$$

2.
$$\psi_p=\begin{cases}\psi_p & \text{if } \psi_p=\max_{a=1}^{10}\{\psi_a\}\\ \psi_p\cdot\left(1-(\max_{a=1}^{10}\{\psi_a\})^{10}\right) & \text{if } \psi_p\neq\max_{a=1}^{10}\{\psi_a\}\end{cases} \tag{2.45}$$

3.
$$\psi_p=\psi_p/\sum_{a=1}^{10}\psi_a \tag{2.46}$$

Note that equation (2.43) allows a single function to be composed of versions of itself with different orientations. The individual underlying functions that are used are listed in Table 2.2. Each function has a different range, so a stretch factor is employed to ensure a uniform range of the composition function $F(\vec{x})$. The stretch factor σ_{f_p} is calculated for each underlying function as

$$\sigma_{f_p}=\frac{V_{max,F}-V_{min,F}}{V_{max_{f_p}}-V_{min_{f_p}}} \tag{2.47}$$

where $[V_{max,F},V_{min,F}]^{n_d}$ is the search range of $F(\vec{x})$ which is set to $[5,5]^{n_d}$ and the search range $[V_{max_{f_p}},V_{min_{f_p}}]^{n_d}$ is that of the underlying function f_p.

Table 2.2: Details of the underlying benchmark functions of the GDBG

Name	**Function**	**Range**
Sphere	$f(\vec{x}) = \sum_{j=1}^{n_d} x_j^2$	[-100,100]
Rastrigin	$f(\vec{x}) = \sum_{j=1}^{n_d} (x_j^2 - 10\cos(2\pi x_j) + 10)$	[-5,5]
Weierstrass	$f(\vec{x}) = \sum_{j=1}^{n_d} \left(\sum_{a=0}^{20} \left(0.5^a \cos\left(2\pi \cdot 3^a (x_j + 0.5)\right)\right)\right)$ $-n_d \sum_{a=0}^{20} \left(0.5^a \cos(\pi \cdot 3^a)\right)$	[-0.5,0.5]
Griewank	$f(\vec{x}) = \frac{1}{4000} \sum_{j=1}^{n_d} x_j^2 - \prod_{j=1}^{n_d} \cos(\frac{x_j}{\sqrt{j}}) + 1$	[-100,100]
Ackley	$f(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n_d}\sum_{j=1}^{n_d} x_j^2}\right)$ $-\exp\left(\frac{1}{n_d}\sum_{j=1}^{n_d}\cos(2\pi x_j)\right) + 20 + e$	[-32,32]

Changes in the environment occur when $\vec{l_p}$ and $\vec{h}$ are changed using the *DynamicChanges* function:

$$\vec{h}(t+1) = DynamicChanges(\vec{h}(t)) \quad (2.48)$$

$$\vec{l_p}(t+1) = DynamicChanges(\vec{l_p}(t)) \; \forall \; p \in \{1, 2, \ldots, n_p\} \quad (2.49)$$

F_2 is made up of a composition of the Sphere function, illustrated in two dimensions in Figure 2.5. F_2 has 10 optima. The third function, F_3, contains a composition of Rastrigin's function, illustrated in two dimensions in Figure 2.6. F_4 is a composition of the Griewank function, illustrated in two dimensions in Figure 2.7. The fifth function, F_5, is conctructed from a composition of Ackley's function, illustrated in two dimensions in Figure 2.8. The last function is a composition of the Sphere, Ackley, Griewank, Rastrigin and Weierstrass functions. Figure 2.9 depicts F_6 in two dimensions. Functions F_3 to F_4 all have a large number of optima.

F_1 is maximised, while F_2 to F_6 are minimised. For all functions (F_1 to F_6) the following settings are used:

- Height range: $\phi_{h,max} = 10$ and $\phi_{h,min} = 1$
- Height severity: $\phi_{h,sev} = 5.0$
- Initial height: $initial\ width = 50$

2.5.4.2 Generalised Benchmark Generator Change Types

Six different change types are used in the benchmark to investigate algorithm performance under different types of changes. The experimenter selects one of the following change types which governs the functioning of the function *DynamicChanges*:

1. Small step changes, T_1:

$$\phi_{\Omega,q}(t+1) = \phi_{\Omega,q}(t) + 0.02 \cdot (\phi_{\Omega,max} - \phi_{\Omega,min}) \cdot U(-1,1) \cdot \phi_{\Omega,sev} \tag{2.50}$$

2. Large step changes, T_2:

$$\phi_{\Omega,q}(t+1) = \phi_{\Omega,q}(t) + (\phi_{\Omega,max} - \phi_{\Omega,min}) \cdot (0.02 \cdot sign(U(-1,1)) + 0.08 \cdot U(-1,1)) \cdot \phi_{\Omega,sev} \tag{2.51}$$

3. Random changes, T_3:

$$\phi_{\Omega,q}(t+1) = \phi_{\Omega,q}(t) + N(0,1) \cdot \phi_{\Omega,sev} \tag{2.52}$$

4. Chaotic changes, T_4:

$$\phi_{\Omega,q}(t+1) = 3.67 \cdot (\phi_{\Omega,q}(t) - \phi_{\Omega,min}) \cdot \left(1 - \frac{(\phi_{\Omega,q}(t) - \phi_{\Omega,min})}{(\phi_{\Omega,max} - \phi_{\Omega,min})}\right) \tag{2.53}$$

5. Recurrent changes, T_5:

$$\phi_{\Omega,q}(t+1) = \phi_{\Omega,min} + (\phi_{\Omega,max} - \phi_{\Omega,min}) \cdot \left(\sin\left(2\pi\left(\frac{t}{12} + \frac{q}{10}\right)\right) + 1\right)/2 \tag{2.54}$$

6. Recurrent changes with noise, T_6:

$$\phi_{\Omega,q}(t+1) = \phi_{\Omega,min} + (\phi_{\Omega,max} - \phi_{\Omega,min}) \cdot \left(\sin\left(2\pi\left(\frac{t}{12} + \frac{q}{10}\right)\right) + 1\right)/2 + N(0,0.8) \tag{2.55}$$

where $\phi_{\Omega,sev}$, $\phi_{\Omega,max}$ and $\phi_{\Omega,min}$ correspond to the upper bound, lower bound and severity values of the specific control parameter to be changed. The creators of the GDBG suggested that changes occur once every 100 000 function evaluations.

The GDBG also provides a change type that allows the number of dimensions to vary over time. This type of dynamic environment is outside the scope of this thesis.

2.5.4.3 Generalised Benchmark Generator Critical Discussion

Each of the six functions of the GDBG provide a heterogeneous fitness landscape (refer to point 1 discussed in Section 2.5.2) which consists of underlying functions typically found in benchmarks for static environments. The simulated dynamic environments can all be classified as type III in terms of change pervasiveness (refer to point 3 discussed in Section 2.5.2).

The GDBG provides six change types which allows researchers to investigate the performance of the optimisation algorithms under different types of changes. All the change types identified under point 2 in Section 2.5.2 are represented in the GDBG. A disadvantage of the rigidly defined and complex change types of the GDBG is that investigations into the scalability of algorithms in terms of change severity are not explicitly supported. The settings of the GDBG can be altered in a limited extent to vary the change severity, by using different change types, but much of what is intuitively understood as change severity is lost. The GDBG does not have a parameter, like it has with the change period, that can be varied over a range of values to observe the scalability of an algorithm with respect to change severity.

Recent years have seen the development of a considerable number of algorithms to solve dynamic optimisation problems (refer to Section 3.3). It is important to be able to compare these algorithms in order to determine which algorithms perform best on various problems. The next section reviews metrics that have been suggested to measure performance.

2.5.5 Performance Measures for Dynamic Environments

The purpose of performance measures is firstly to measure how effective an algorithms is at solving an optimisation problem, and secondly to provide a baseline for comparisons among optimisation algorithms. The goal of this section is to discuss proposed performance measures for dynamic optimisation problems. The most appropriate performance measure for this study can be identified from this discussion. According to Morrison [2003], a good performance measure should:

1. have intuitive meaning,

2. allow plain statistical significance testing, and

3. have a sufficiently large exposure to landscape dynamics.

The third requirement implies that performance measures which are normally used in static environments are not appropriate for dynamic environments, since these are, generally, only focused on the performance of an algorithm at the conclusion of the optimisation process. In dynamic environments, the effectiveness of an algorithm has to be considered over the entire span of the optimisation process and over all environment changes.

An important consideration when choosing a performance measure is the information that is available to the observer. Benchmark problems typically provide information regarding the function value and location of the global optimum, and when changes in the environment occur. In the case of the moving peaks benchmark (refer to Section 2.5.3) the function values and locations of local optima are also available. This information can be used to measure the performance of an algorithm more accurately. However, the above mentioned information may not be available when solving real world dynamic optimisation problems [Nguyen *et al.*, 2012].

A survey of the literature identified 13 performance measures:

1. Graphs of the average best-of-generation at each generation [Grefenstette, 1999] [Bäck, 1998], calculated at the g-th generation as:

$$H_{AG,g} = F(\vec{x}_{best}(g), t_{\vec{x}_{best}(g)}) \tag{2.56}$$

   where $\vec{x}_{best}(g)$ is the best performing individual in generation g and $F(\vec{x}, t)$ is a dynamic optimisation function. The time step in which $\vec{x}_{best}(g)$ is evaluated is $t_{\vec{x}_{best}(g)}$. The value of each $H_{AG,g}$ is averaged over many repeats of the same experiment, and are subsequently plotted on a graph of $H_{AG,g}$ versus generations.

   The graphs provide a visual representation of how fitness improves after each change in the environment. However, a visual comparison of the graphs of more than one algorithm does not provide clear information as to which algorithm performed the best [Morrison, 2003]. It is also impossible to do statistical significance testing between the results of multiple algorithms when using $H_{AG,g}$ graphs, as they support visual comparisons rather than numerical comparisons.

A further disadvantage of the $H_{AG,g}$ measure is that it is sampled once every generation, hence making the time between samples dependent on the population size. An algorithm with a large population size has a natural advantage over an algorithm with a small population size, since the larger population size ensures a broader exposure to the fitness landscape. A fair comparison of two algorithms using $H_{AG,g}$ thus requires equal population sizes, as $H_{AG,g}$ does not consider the computational cost per generation.

2. The average error of the best individuals just before a change in the environment [Trojanowski and Michalewicz, 1999b]:

$$H_{AEBI} = \frac{\sum_{c=1}^{n_c} E(\vec{x}_{best}(t_c), c)}{n_c} \tag{2.57}$$

where n_c is the total number of changes in the environment, t_c is the time step before the c-th change in the environent, $\vec{x}_{best}(t)$ the best individual found since the last change in the environment and $E(\vec{x}, c)$ is the error of individual $\vec{x}$ immediately before the c-th change in the environment.

This performance measure provides an indication of how well the fitness landscape is optimised before changes in the environment and allows for statistical significance testing. A further advantage of this performance measure is that it allows for uniform treatment of function maximisation and minimisation problems, since the error and not the function value is used. It is thus always desirable to achieve a low value for H_{AEBI}.

A disadvantage of this performance measure is that it only considers the performance immediately before changes in the environment. The performance during periods between changes are ignored. An algorithm that recovers quickly after changes in the environment will not be rated better than an algorithm with a slow recovery time.

3. The lowest error value found of the best individual just before a change, over all changes [Li *et al.*, 2008]:

$$H_B = \min_{c=1,\ldots,n_c} \{E(\vec{x}_{best}(t_c), c)\} \tag{2.58}$$

This measure is meant to be used in conjunction with performance measure number 2, and gives the minimum error value found. Very little can be learnt by using this measure, since an algorithm that achieves a single very low error value at some point in the optimisation process, but high error values during the rest of the optimisation process, would be judged to be more effective than an algorithm that consistently produces moderately low error values.

4. The worst error value found of the best individual just before a change, over all changes [Li *et al.*, 2008]:

$$H_W = \max_{c=1,\ldots,n_c} \{E(\vec{x}_{best}(t_c), c)\} \tag{2.59}$$

This measure is meant to be used in conjunction with performance measure numbers 2 and 3, and gives the worst just-before-change error value. H_W does not represent an accurate reflection of an algorithm's performance, since the error value at just a single point in the optimisation process is reported.

5. The online performance is the running average of all fitness evaluations [Branke, 2002]:

$$H_{ON} = \frac{\sum_{t=1}^{n_t} F(\vec{x}(t), t)}{n_t} \tag{2.60}$$

where $\vec{x}(t)$ is the individual evaluated at function evaluation t, $F(\vec{x}, t)$ is the dynamic function and n_t is the total number of function evaluations.

A disadvantage of this measure is that fitness values of all individuals are taken into account (not only the best performing individuals). An algorithm is penalised for exploring sub-optimal regions of the fitness landscape even if it locates the global optimum. This performance measure is thus biased towards algorithms with low population diversity and fast convergence.

On the other hand, an observer using this performance measure does not require information regarding when changes in the environment occur and the location or function value of the global optimum.

6. The offline performance is the running average of the best-so-far fitness found since the last change in the environment [Branke, 2002]:

$$H_{OP} = \frac{\sum_{t=1}^{n_t} F(\vec{x}_{best}(t), t)}{n_t} \tag{2.61}$$

where n_t is the total number of function evaluations and $F(\vec{x}, t)$ is a dynamic function.

This measure addresses the disadvantages of measure number 5 by averaging over the function value of the best individual found since the last change in the environment at each time step. A disadvantage of using H_{OP} is that function maximisation and minimisation problems cannot be treated uniformly, since a large value for H_{OP} is desired for maximisation problems and a low value is desired for minimisation problems. Furthermore, averaging over the fitness makes intuitive interpretation of the results difficult, since the optimum value of the dynamic fitness function may vary over time, leaving the observer without any knowledge of how good a particular fitness value is.

7. The offline error is the running average of the lowest-so-far error found since the last change in the environment [Branke, 2002]:

$$H_{OE} = \frac{\sum_{t=1}^{n_t} E(\vec{x}_{best}(t), t)}{n_t} \tag{2.62}$$

where n_t is the total number of function evaluations and $E(\vec{x}_{best}(t), t)$ is the error of the best individual found since the last change in the environment.

This performance measure is similar to measure number 6, but averages over the error of the best found individual since the last change in the environment. It addresses a disadvantage of performance measure number 2 in that the error during the entire optimisation process is taken into account (not only the error just before changes). The measure thus also rewards algorithms for recovering quickly after changes in the environment, not only for achieving a low error value.

An advantage of averaging over the error and not the fitness provides a more intuitive interpretation of the results, since the observer knows that the ideal error value is

zero. In addition, by making use of the error rather than the function value allows for uniform treatment of maximisation and minimisation problems. H_{OE} can only be used when information regarding the value of the global optimum and when changes in the environment occurs, is available.

The offline error performance measure was suggested as part of the moving peaks benchmark (refer to Section 2.5.3), and has become one of the most commonly used performance measures for dynamic optimisation problems.

8. The collective mean fitness is the average of best-of-generation over all generations [Morrison, 2003]:

$$H_{CM} = \frac{\sum_{g=1}^{n_g} F(\vec{x}_{best}(g), t_{\vec{x}_{best}(g)})}{n_g} \tag{2.63}$$

where $\vec{x}_{best}(g)$ is the best performing individual in generation g, n_g is the total number of generations and $F(\vec{x}, t)$ is a dynamic optimisation function. The time-step in which $\vec{x}_{best}(g)$ is evaluated is $t_{\vec{x}_{best}(g)}$.

This performance measure requires no information regarding when changes in the environment occur, or the value of the global optimum. However, as was the case with performance measure number 6, averaging over the function value complicates interpretation of the results.

Another disadvantage is that the population size may influence the value of H_{CM} because the average is taken over the number of generations. For example, an algorithm with a small population size may have a larger value for n_g over a constant number of fitness evaluations than an algorithm with a large population size. The average is taken over the function value of the best individual in each generation, so the algorithm with the smaller population size is more likely to have generations included in the average where the best individual performed relatively poorly.

9. The average minimum Euclidean distance to optimum at each generation [Weicker

and Weicker, 1999]:

$$H_{AD} = \frac{\sum_{g=1}^{n_g} \left(\min_{i=1,\ldots,n_I} \{ \|\vec{l}_F(t) - \vec{x}_i(t)\|_2 \} \right)}{n_g} \tag{2.64}$$

where n_g is the total number of generations and $\vec{l}_F(t)$ is the location of the global optimum of the dynamic function (assuming a single global optimum). This measure requires information regarding the location of the global optimum.

The major disadvantage of this performance measure is that only the global optimum is taken into account. Algorithms are not rewarded of locating local optima. Ideally, the global optimum should be located, but local optima may have function values very close to that of the global optimum, and could represent an adequate solution to the optimisation problem. A further disadvantage of using H_{AD} is that it ignores both the function value and the error of individuals. The gradient around the global optimum can be steep for a particular optimisation problem which would result in a high error value for an individual that is relatively close, but this high error value would not be reflected in H_{AD}.

10. The ratio of the difference between the best-of-generation and the worst fitness found within a window of recent generations, over the difference of the best fitness found within the window and the worst fitness found within the window [Weicker, 2002]:

$$H_{RW} = \frac{\sum_{g=1}^{n_g} \left(\frac{F(\vec{x}_{best}(g), t_{\vec{x}_{best}(g)}) - W_{worst}(g)}{W_{best}(g) - W_{worst}(g)} \right)}{n_g} \tag{2.65}$$

$$W_{worst}(g) = \min_{g'=g-\omega,\ldots,g} \left\{ \min_{i=1,\ldots,n_I} \{ F(\vec{x}_i(g'), t_{\vec{x}_i(g')}) \} \right\} \tag{2.66}$$

$$W_{best}(g) = \max_{g'=g-\omega,\ldots,g} \left\{ \max_{i=1,\ldots,n_I} \{ F(\vec{x}_i(g'), t_{\vec{x}_i(g')}) \} \right\} \tag{2.67}$$

where $W_{best}(g)$ is the function value of the best performing individual within a window of ω generations and $W_{worst}(g)$ is the function value of the worst performing individual within a window of ω generations. The i-th individual in generation g is represented by $\vec{x}_i(g)$, and the individual is evaluated at time step $t_{\vec{x}_i(g)}$. The best individual within generation g is denoted by $\vec{x}_{best}(g)$. The above equations assume

a function maximisation problem. This performance measure does not require information regarding the value of the global optimum or when changes occur. A value close to 1.0 represents good performance while a value close to zero represents poor performance.

The mayor disadvantage of using this performance measure is that performance is measured relative to the difference between the best and worst performing individuals within the window. Consequently, a poor performing population that converged to the extend where all individuals have a very similar fitness value, will still receive a relatively high value for H_{RW}. Misleading results can thus ensue from using this performance measure.

A further disadvantage of using H_{RW} is that results will be affected by the window size ω and the population size, since both influence the number of function evaluations that are included within the window.

11. The sampled relative error, H_{RE} [Li *et al.*, 2008]. Let the relative error $RE(t)$ be defined for maximisation problems as:

$$RE(t) = \frac{F(\vec{x}_{best}(t), t)}{F(\vec{l}_F(t), t)} \tag{2.68}$$

and for minimisation problems as:

$$RE(t) = \frac{F(\vec{l}_F(t), t)}{F(\vec{x}_{best}(t), t)} \tag{2.69}$$

A relative error close to 1.0 indicates good performance while values smaller than 1.0 indicate poorer performance. The relative error is sampled n_s times between successive changes in the environment, i.e. $RE(t_{1,c}), RE(t_{2,c}), \ldots, RE(t_{n_s,c})$ before the c-th change in the environment. The sampled relative error performance measurement, H_{RE}, is calculated as:

$$H_{RE} = \frac{\sum_{c=1}^{n_c} \left(\frac{RE(t_{n_s,c})}{1 + \sum_{a=1}^{n_s} (1 - RE(t_{a,c}))/n_s} \right)}{n_c} \tag{2.70}$$

where n_c is the total number of changes in the environment. This performance measure requires information regarding when changes in the environment occur and the value of the global optimum.

There are several disadvantages to H_{RE} as a performance measure. Firstly, H_{RE} is inappropriate for problems where the function value of the global optimum is equal to zero since this will always result in a RE value of zero for a minimisation problem and a divide by zero for a maximisation problem. Secondly, the value of the global optimum must be positive for function minimisation problems, otherwise RE values of greater than 1.0 is found. Thirdly, errors are not treated consistently for minimisation and maximisation problems, because RE increases linearly for maximisation problems as $F(\vec{x}_{best}(t), t)$ tends to $F(\vec{l}_F(t), t)$, while RE increases hyperbolically for minimisation problems as $F(\vec{x}_{best}(t), t)$ tends to $F(\vec{l}_F(t), t)$.

In addition to the above mentioned disadvantages, the fact that errors are only sampled periodically means that the performance of the algorithm is not taken into account during the intermediate periods of the optimisation process.

12. Peak cover, a performance measure specifically created for the moving peaks benchmark (refer to Section 2.5.3), measures the average ratio of the number of peaks on which there are individuals over the number of peaks whose optima are not obscured by other peaks [Branke, 2002]:

$$covered\_peaks = \sum_{p=1}^{n_p} \begin{cases} 1 \text{ if } \left(\exists i \in \{1, \ldots, n_I\} \text{ such that } f_p(\vec{x}_i, t) = F(\vec{x}_i, t)\right) \\ \qquad \text{and } \left(f_p(\vec{l}_p, t) = F(\vec{l}_p, t)\right) \\ 0 \text{ otherwise} \end{cases} \tag{2.71}$$

$$not\_hidden\_peaks = \sum_{p=1}^{n_p} \begin{cases} 1 \text{ if } f_p(\vec{l}_p, t) = F(\vec{l}_p, t) \\ 0 \text{ otherwise} \end{cases} \tag{2.72}$$

$$H_{PC} = \frac{\sum_{g=1}^{n_g} \frac{covered\_peaks}{not\_hidden\_peaks}}{n_g} \tag{2.73}$$

$$\tag{2.74}$$

where $f_p(\vec{x}, t)$ is the function value of the p-th peak, $\vec{l}_p$ is the location of the optimum of the p-th peak, $F(\vec{x}, t)$ is the dynamic function that contains the p peaks and n_p is the number of peaks. A common strategy for optimising dynamic problems is to track all optima in the environment in order to recover quickly from changes in the environment (refer to Section 3.3). H_{PC} measures how well all optima are tracked.

Although several algorithms attempt to track all optima, it can be argued that the possibility should not be excluded that an effective algorithm could be developed that does not make use of this strategy. Accordingly, H_{PC} is, in general, not an appropriate performance measure.

A further disadvantage of the peak cover measure is that it can only be used when information regarding the locations of all peaks are available, and if $F(\vec{x}, t)$ is calculated using the maximum value of all peaks, as described in point 1(b)ii on page 33. This is not the case with all benchmark functions and very unlikely in real world problems.

13. Best known peak error, a performance measure specifically created for the moving peaks benchmark, was designed by Bird and Li [2007] to be used in conjunction with performance measure number 12. The goal of this measure is to measure the convergence speed of an algorithm once it has been found to cover a peak. This is achieved by calculating the minimum error found for each peak at the end of each generation:

$$\xi_{p,g} = \min_{i=1,\ldots,n_I} \{E(\vec{x}_i(g), t_{\vec{x}_i(g)}) \mid f_p(\vec{x}_i, t) = F(\vec{x}_i, t)\} \tag{2.75}$$

where $\xi_{p,g}$ is the minimum error found on the p-th peak during generation g, $\vec{x}_i(g)$ is the location of the i-th individual during the g-th generation and $t_{\vec{x}_i(g)}$ is the time step in which $\vec{x}_i(g)$ is evaluated.

When a change in the environment occurs, the index, a_c , of the peak with the lowest error during the last generation before the change is found:

$$a_c = p \text{ where } \xi_{p,c\times n_{g,c}} = \min_{p'=1,\ldots,n_p} \{\xi_{p',c\times n_{g,c}}\} \tag{2.76}$$

where $n_{g,c}$ is the number of generations that takes place between two changes in the environment. The best known peak error is then calculated as:

$$H_{BKPE} = \frac{\sum_{c=1}^{n_c} PE_{a_c,c}}{n_g} \tag{2.77}$$

where n_g is the total number of generations, n_c is the total number of changes and the function $PE_{p,c}$ is defined as:

$$PE_{p,c} = \sum_{g=(c-1)\times n_{g,c}+1}^{c\times n_{g,c}} \xi_{p,g} \tag{2.78}$$

This performance measure requires information regarding when changes in the environment occur and the function values of all the local optima.

The major disadvantage of using H_{BKPE} is that only the error of the best known peak is taken into consideration. Since the global optimum is not taken into account, an algorithm with fast convergence to inferior optima can receive a high ranking from H_{BKPE}.

A further disadvantage is that the measure is sampled every generation. The assumption is that the number of function evaluations required to evaluate a generation is much smaller than the number of function evaluations between changes in the environment. This is not the case when a large population size is used, or when changes in the environment occur frequently.

The above discussion of the 13 performance measures pointed out several disadvantages of sampling the error after each generation. A further disadvantage is that the number of fitness evaluations per generation is not constant in several modern algorithms aimed at dynamic optimisation problems (refer to Section 3.3). The number of generations that an algorithm performs will consequently not always be the same when experiments are repeated (when allowing a constant number of fitness evaluations to ensure equal exposure to the fitness landscape).

Considering the positive and negative aspects of the performance measures discussed in this section it was concluded that the most appropriate performance measure to use in this thesis is number 7, the offline error. The offline error measure has several benefits: It conforms to all three requirements of Morrison [2003]: it has intuitive meaning, allows statistical significance testing, and has sufficient exposure to landscape dynamics since the error at all function evaluations contribute to the final offline error. A further advantage of the offline error is that it does not suffer from any of the disadvantages associated with generation based performance measures. In addition, error values are taken into account

as opposed to function values, which means that maximisation and minimisation problems can be treated uniformly. A potential disadvantage of the offline error is that it requires information regarding when changes in the environment occur and the function value of the global optimum. However, this is not seen as a problem, as the required information is available from the benchmark functions used in this thesis.

The offline error averages over the lowest errors found since the last change in the environment (referred to by Branke [2002] as the *current error*). The average current error over several repeats of the optimisation process can be graphed analogously to performance measure number 1 (average best-of-generation graphs) to provide a visual representation of the optimisation algorithm's performance during the optimisation process. Figure 2.10 illustrates the functioning of the offline error measurement and the current error obtained by an elementary random guessing algorithm (given in Algorithm 6) on the Scenario 2 settings of the MPB. Ten changes in the environment are depicted. The offline error is the average of all previous current errors and thus produces a smoother curve than the current error. The current error spikes after each change in the environment and drops as better solutions are found.

Algorithm 6: Elementary random guessing algorithm

$t = 0$;
while *maximum number of function evaluations is not exceeded* **do**
    Uniformly select random vector $\vec{x}(t)$ from the n_d dimensional search space;
    Evaluate $F(\vec{x}(t), t)$;
    $t = t + 1$;
end

Random guessing is not an effective optimisation strategy, and it will be shown in the next chapter that much better results are found even with algorithms not tailored to dynamic environments.

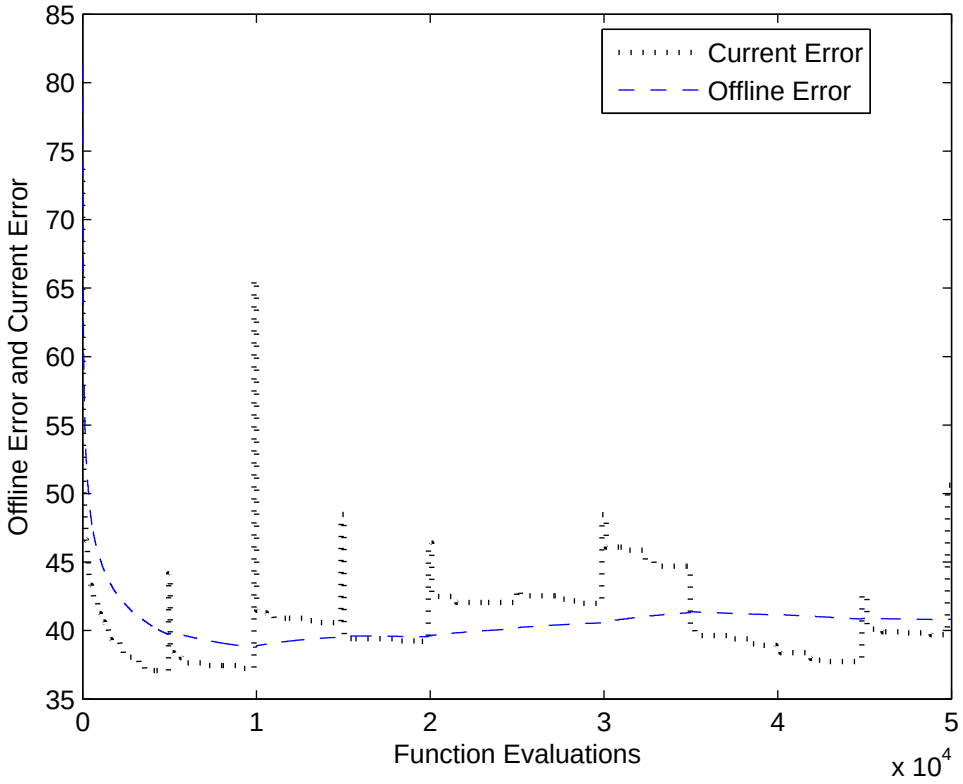

Figure 2.10: Current and offline errors found on the MPB through evaluating random individuals

2.6 Conclusions

This chapter provided background information regarding optimisation with emphasis on DE. DE is a evolutionary optimisation algorithm that employs the spatial difference between individuals to perform mutations. Common DE schemes were reviewed along with the standard DE control parameters. The control parameters can be altered to change the exploration and exploitation characteristics of the DE algorithm. However, fine-tuning the control parameters for a specific problem is a time consuming manual task. Work aimed at reducing control parameters was reviewed with special emphasis on self-adapting parameters.

Dynamic optimisation environments were formally defined. It was found that dynamic environments are typically classified in the literature based on the types of changes that

occur, the fitness landscape composition and the pervasiveness of changes. It was argued that the three most important factors influencing the intricacy of an dynamic optimisation problem is hardness of the fitness landscape, the frequency at which changes occur, and the severity of changes to the environment. An thorough investigation into the effectiveness of an optimisation algorithm should include a scalability study on how the algorithm scales over various values of the three factors that influence the intricacy of the dynamic environment.

Two benchmarks were discussed that can be used to simulate dynamic environments. Both are used in this thesis to investigate the performance of an algorithm with respect to various settings of the number of dimensions and the change period. The first benchmark that was discussed is the moving peaks benchmark. This benchmark has been used by several other researchers, and is simple enough to provide an intuitive understanding of the fitness landscape. The MPB is ideally suited to investigate the scalability of algorithms in terms change severity. The MPB allows researchers to set the number of peaks in the environment, and was extended by the author of this thesis to support the simulation of dynamic environments in which the number of peaks fluctuates over time. The extended MPB is thus ideal for studying how the number of optima in the environment influences the performance of an optimisation algorithm.

The second benchmark is the generalised dynamic benchmark generator. This benchmark simulates dynamic environments composed of benchmark functions typically used in static optimisation. The GDBG provides six different change types. The GDBG can be used to evaluate the performance of an optimisation algorithm on six different functions and change types.

The chapter concluded with description of common performance measures for dynamic environments which were identified in the literature. The advantages and disadvantages of 13 performance measures were discussed. The offline error performance measure, which averages the lowest error value found since the last change in the environment over all function evaluations, was concluded to be the most appropriate performance measure for the current study.

The next chapter introduces related research on optimising dynamic environments with specific focus on DE-based algorithms.

Chapter 3

DYNAMIC ENVIRONMENT OPTIMISATION ALGORITHMS

This chapter considers approaches to adapting optimisation algorithms to solve dynamic optimisation problems. The fundamental problems involved with applying differential evolution to dynamic environments are explained. An outline of related research in the field is given to provide context on the current state of the field. Two algorithms for dynamic optimisation problems which are based on differential evolution are described in detail.

3.1 Introduction

Optimisation algorithms, developed for static environments, tend to be ineffective when applied to dynamic optimisation problems [Noroozi *et al.*, 2011][Li and Yang, 2009]. Section 3.2 describes how low diversity is the main reason for DE's lack of transferability from static to dynamic environments. A secondary problem of outdated information is also described.

A review of related work by other researchers on adapting population-based algorithms to dynamic optimisation problems (DOPs) is given in Section 3.3. Three main approaches for adapting static optimisation algorithms to dynamic environments are identified from existing literature in Section 3.3.1. These approaches are: increasing diversity, memory

and parallel search. The literature review revealed seven general strategies which are commonly used by optimisation algorithms to achieve the three general approaches. The seven strategies are discussed in Section 3.3.2. Specific algorithms based on GA, PSO, DE and other main algorithms are discussed in Section 3.3.3 and each is shown to employ one or more of the seven strategies.

This thesis investigates the use of DE in dynamic environments. Accordingly, two algorithms found in the literature, which are based on DE, are discussed in detail. DynDE, a multi-population algorithm, with measures to increase diversity, is described in Section 3.4.1. A more recent algorithm, *jDE*, is described in Section 3.4.2.

Dynamic optimisation algorithms, typically, have to react to changes in the environment, which makes it necessary for these algorithms to detect changes. Change detection is discussed in Section 3.5. Conclusions are drawn in Section 3.6.

3.2 Differential Evolution in Dynamic Environments

This section describes two properties of DE that make the algorithm ineffective when solving DOPs. The first, loss of diversity, is discussed in Section 3.2.1. Section 3.2.2 describes the problem of outdated information.

3.2.1 Loss of Diversity

The initial population of DE (and of most evolutionary algorithms) is formed from individuals that are randomly dispersed in the search space. This ensures that the algorithm explores a large area of the fitness landscape during the first generations. Eventually, however, DE converges to an optimum, with all the individuals clustered around a single point, resulting in a loss of diversity. However, the eventual loss of diversity in static environments is not a problem, because having all individuals clustered around the optimum assists with exploitation at the end of the search process.

The loss of diversity is, however, a major reason why evolutionary algorithms are ineffective in dynamic environments. The population lacks the diversity necessary to locate the position of new global optima when changes in the environment occur after the population starts to converge [Zaharie and Zamfirache, 2006]. This phenomenon can

be illustrated by means of an example using DE. Consider Figure 3.1 which depicts the offline error, current error and diversity of the populations (calculated using equation (2.7)) averaged over 30 independent runs on the MPB of the basic DE algorithm described in Section 2.4. Ten changes in the environment are illustrated. Changes occur once every 5 000 function evaluations.

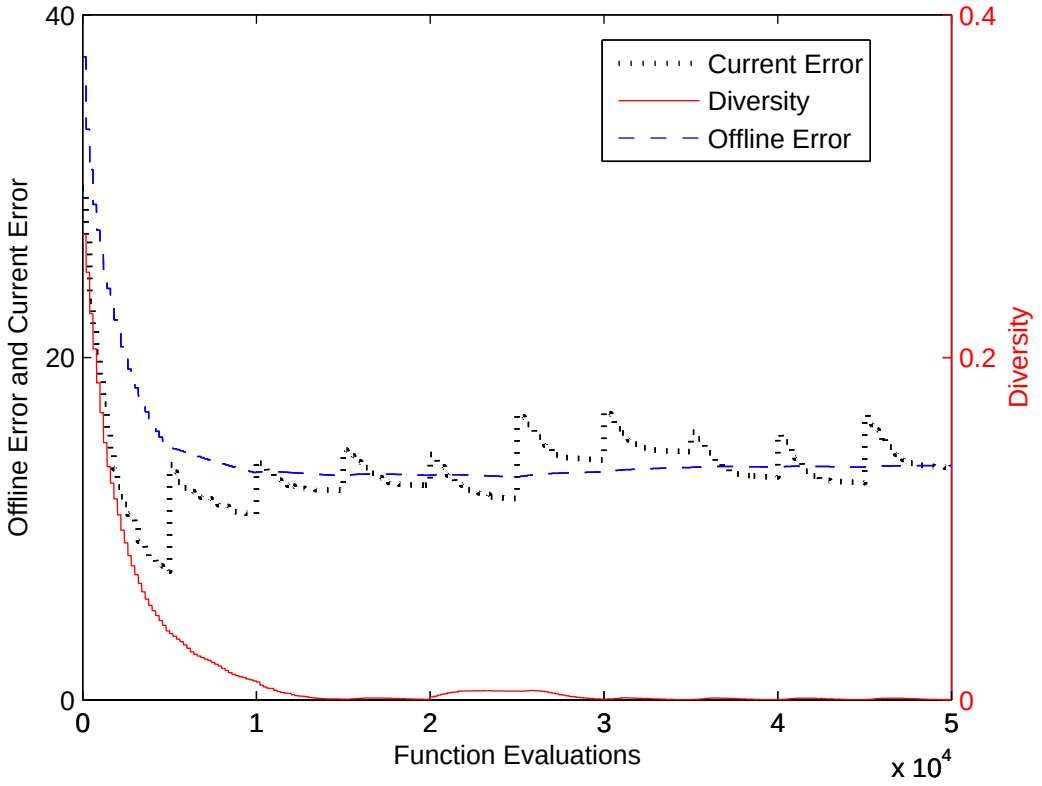

Figure 3.1: Diversity, current error and offline error of basic DE on the MPB

During the period between the commencement of the algorithm and the first change in the environment, the current error and the offline error drop sharply as the DE algorithm converges to one optimum in the environment. The fact that individuals are clustered increasingly closely together is illustrated by the sharp drop in diversity. Directly after the first change in the environment, the current error increases, since the optima have shifted. Although the current error does improve after the first and subsequent changes, it never reaches the low value that was found before the first change occurred. Diversity continues

to drop until it reaches a value close to zero shortly after the second change in the environment (at about 1000 function evaluations in Figure 3.1). This means that all individuals are clustered around a single optimum and that the rest of the fitness landscape is left completely unexplored. This clustering is especially detrimental in DE, since mutations are performed based on the spatial differences between individuals. Consequently, very small mutations are applied as a result of spatially close individuals which restricts further exploration of the fitness landscape. The average diversity per generation was 0.0017, averaged over 30 repeats of DE on the MPB. This value is relatively low and explains why normal DE is not effective in dynamic environments.

3.2.2 Outdated Information

DE performs selection as a competition between parents and offspring. Re-evaluation of parent individuals for each competition comparison in a static environment is unnecessary and ineffective, since the fitness values at particular points in the fitness landscape never change. Re-evaluation thus implies wasted function evaluations.

However, in dynamic environments, the function values of individuals become outdated when the environment changes. As a result, the algorithm can no longer rely on the fitness values of parent individuals to contribute meaningfully to the search process. The incorrect fitness values of the parent individuals may guide the search to regions of the fitness landscape with inferior fitness. A parent with an outdated, high fitness value, which in reality is less fit than its offspring, may be erroneously preserved by the parent versus offspring competition.

The outdated information problem can be solved by re-evaluating the function values of individuals when changes in the environment occur. The changed fitness landscape is reflected in the updated fitness values of parent individuals. Consequently, valid results can be found from the parent versus offspring competition.

3.3 Related Work

Several of the earlier investigations into optimisation in dynamic environments involved approaches based on genetic algorithms [Holland, 1975]. More recently, attention has been

given to other population-based optimisation algorithms like particle swarm optimisation (PSO) [Kennedy and Eberhart, 1995] and differential evolution [Price *et al.*, 2005][Storn, 1996].

This section provides a review of algorithms developed for solving DOPs, with the focus on those algorithms directly related to the objectives of this study. Figure 3.2 shows the layout of this section.

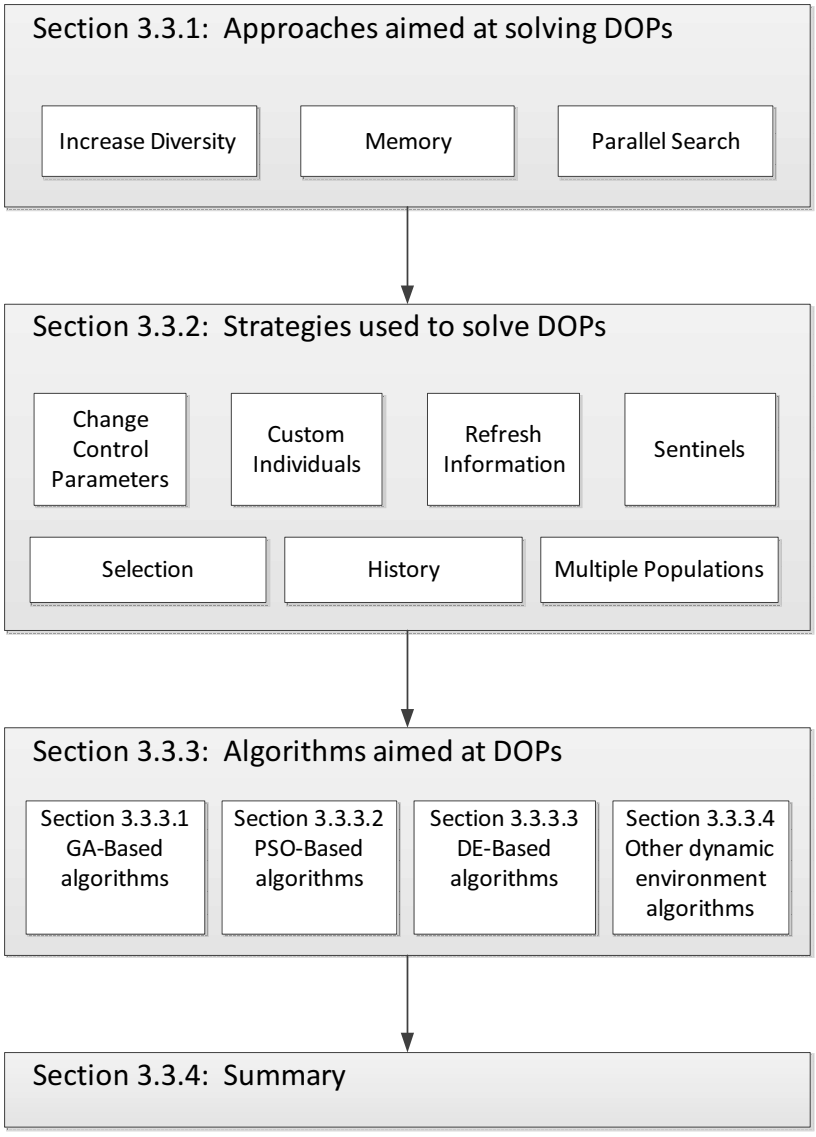

Figure 3.2: Layout of the related work section

The discussion is grouped into four parts. The first part, given in Section 3.3.1, de-

scribes the three main categories into which approaches, tailored to DOPs, can be categorised. The objective of the section is to explain *what* algorithms aimed at DOPs try to achieve. Three broad categories are identified and motivated. The second part, given in Section 3.3.2, describes specific algorithmic strategies aimed at DOPs. This section explains *how* the approaches described in Section 3.3.1 are achieved. Seven strategies are identified and explained. The strategies are discussed generically as far as possible, without details relating to specific implementations. The third part, given in Section 3.3.3, describes particular algorithms aimed at DOPs in terms of the underlying algorithms (GA, PSO, DE, and other). For each algorithm, the underlying strategies that are employed are highlighted. The categorisation of the algorithms of Section 3.3.3 into the strategies identified in Section 3.3.2 is summarised in Section 3.3.4.

3.3.1 Approaches aimed at solving Dynamic Optimisation Problems

A review of the literature identified three main categories of approaches used to adapt algorithms for dynamic environments. At least one of these is present in most algorithms aimed at DOPs. These categories can be seen as the goals which algorithms, aimed at DOPs, try to achieve. The three categories (adapted from [Jin and Branke, 2005]) are:

Increase Diversity: Section 3.2.1 argued that one of the main reasons why traditional population-based algorithms fail, when applied to dynamic environments, is that the algorithms converge to a solution and then lack the diversity required to locate new optima once the environment changes. High diversity results in a larger area of the search space being covered by the population, which increases the probability of relocating optima after a change in the environment occurs [Ursem, 2002]. Jin and Branke [2005] and Nguyen *et al.* [2012] pointed out that most algorithms try to remedy the low diversity problem by either explicitly increasing diversity after a change in the environment is detected, or by maintaining more diversity during the optimisation process.

Memory: Optima in dynamic environments shift, disappear, or new optima may appear when changes in the environment occur. However, unless the changes are large, information regarding where optima were located before a change can still be used

to locate new optima in their vicinity after the change. A change in the environment can also lead to the conversion of global optima to local optima and vice versa. Knowing the location of all (or at least several) of the local optima, can accelerate the location of a new global optimum after a change in the environment.

A cyclic dynamic environment (discussed in Section 2.5.2) periodically returns to the same state. A long-term memory of previously found, good solutions can be used to recognise locations of optima when a complete cycle has occurred. Memory has also been used to extrapolate future states of the environment in situations where a correlation between successive changes exists.

Preserving information, in the form of a memory of the search space characteristics from before a change in the environment, can thus be utilised to improve the exploration of the fitness landscape after a change in the environment [Nguyen *et al.*, 2012]. Note that the potential benefits of memory-approaches diminish as the severity of changes in the environment increases or becomes more chaotic.

Parallel Search: Evolutionary and swarm algorithms are inherently parallel search algorithms by virtue of multiple individuals in the population. Many algorithms further extend this parallelism by employing independent parallel searches to locate optima, mostly by using multiple independent sub-populations [Nguyen *et al.*, 2012]. Several promising optima can be investigated simultaneously by using multiple populations, which increases the likelihood of discovering a global optimum. It is common for algorithms to prevent sub-populations from converging to the same optimum and to control how individuals are distributed among the sub-populations.

The following section details the mechanisms employed by algorithms to achieve the above mentioned approaches.

3.3.2 Strategies used to solve Dynamic Optimisation Problems

Seven common strategies, used to create optimisation algorithms for dynamic environments, were identified in this study and are presented in this section. Strategies typically fall within one or more of the approach categories, which were identified in the previous

section. The strategies are described, as far as possible, independently of the base algorithms (specific GA, PSO and DE algorithms that employ these strategies are discussed in the following section). The seven strategies are:

1. **Change Control Parameters:** Several control parameters of evolutionary and swarm algorithms directly affect diversity, for example, the mutation rate used in a GA and the scale factor used in DE [Zaharie, 2002a]. Other parameters, for example, population size [Teo, 2006], DE scheme (refer to Section 2.4.2) and PSO neighbourhood structure, affect the convergence rate which indirectly influences diversity. Several researchers experimentally determined the most effective control parameters to use in conjunction with their algorithms on dynamic environments [Mendes and Mohais, 2005]. Other algorithms adapt control parameters during the course of the algorithm's execution [Brest *et al.*, 2009].

   Changed control parameters are generally used to increase the diversity of the population. This assists with locating global optima (since a larger area of the search space is explored) and helps prevent convergence of the entire population, which hampers the algorithm's ability to respond to changes in the environment.

2. **Custom Individuals:** Several approaches to increasing diversity have focused on changing the behaviour of all, or a portion of, individuals within the population [Cruz *et al.*, 2011]. The custom individuals are normally used to increase diversity by perturbating their locations using random noise [Blackwell and Branke, 2006]. By employing these custom individuals as a portion of the population, the normal individuals within the population can converge to an optimum while the population, as a whole, can still have a relatively high diversity due to the influence of the custom individuals.

3. **Refresh Information:** To prevent populations from converging to a single point (i.e. losing all diversity), steps can be taken to introduce fresh genetic material into the population [Grefenstette, 1999]. Common approaches include reinitialising the entire population, incorporating random individuals into the population, reinitialising the personal best value in PSO, and employing an aging metaphor to replace

individuals and populations that have survived for many generations with random individuals.

Refreshing information has the benefit of increasing diversity, but also of assisting the parallel search of the fitness landscape as new global optima may be discovered after a population is reinitialised. Depending on the algorithm, the potential exists of losing information when reinitialising individuals and populations. This approach is consequently used, mostly in conjunction with multiple populations, so that information regarding the locations of global optima would not be lost.

4. **Sentinels:** Individuals that are kept at constant positions within the search space are referred to as sentinels. Sentinels are commonly used to detect changes in the environments (see Section 3.5), but a uniformly distributed group of sentinels in a population can also increase diversity by providing genetic material from areas of the search space not represented by normal individuals [Morrison, 2004]. For example, a sentinel that acquired a relatively high fitness after a change in the environment can be used to guide the population towards an optimum.

5. **Selection:** Diversity can be explicitly controlled by employing a custom selection operator. For example, a selection operator that not only selects individuals based on fitness, but which is also more likely to select individuals that are located in underpopulated areas of the search space [Oppacher and Wineberg, 1999]. Individuals that have converged to a single point thus have a lower probability of surviving to the next generation. The resulting population is thus likely to be more widely dispersed within the search space.

6. **History:** Keeping track of good solutions within the fitness landscape can be beneficial in cyclic dynamic environments. An archive which acts as a memory of the search history is utilised by algorithms to store good solutions over several generations [Cruz *et al.*, 2011]. When optima reappear in the same position as previously, individuals from the archive can redirect the search to these positions by contributing genetic material to the main population [Ramsey and Grefenstette, 1993].

   A secondary benefit of an archive of previous solutions is that it can be used to

increase diversity by injecting genetic material from areas of the fitness landscape other than the area to which the main population has converged.

With reference to the three categories discussed in the previous section, the use of an archive is an example of an explicit memory of the fitness landscape.

7. **Multiple Populations:** The use of multiple populations falls into all three of the previously mentioned categories: diversity, memory and parallel search. By enforcing a spatial distance between sub-populations it can be ensured that sub-populations converge to different optima in the environment, which in turn lowers the risk of converging to a local optimum (especially in situations where the number of sub-populations is roughly equal to the number of optima). By virtue of more widely distributed individuals, diversity is increased [Cruz *et al.*, 2011]. Keeping track of several optima means that more information about the search space is available when changes in the environment occur; in effect a clearer memory of the previous shape of the fitness landscape is available [Ursem, 2000]. Multiple populations is a form of implicit memory of the locations of optima before a change in the environment. Knowing the locations of pre-change optima is useful when changes in the environment are relatively small, as these locations are a good starting point for searches for the new locations.

The majority of evolutionary algorithms make use of at least one of the seven strategies described in this section. The exact details of how these strategies are implemented in the various algorithms aimed at DOPs are given in the following section.

3.3.3 Algorithms aimed at Dynamic Optimisation Problems

An overview of specific algorithms for optimising dynamic environments is given in this section, with reference to the seven strategies identified in the previous section. These algorithms are discussed in the following sections in terms of their respective base algorithms. Section 3.3.3.1 describes algorithms based on genetic algorithms, Section 3.3.3.2 describes algorithms based on particle swarm optimisation, Section 3.3.3.3 describes algorithms based on differential evolution, and Section 3.3.3.4 describes algorithms with hybrids or other bases.

3.3.3.1 Genetic Algorithm-Based Algorithms

Early GA approaches to solving DOPs endeavoured to increase population diversity after changes occurred in the dynamic environment by increasing the mutation rate. Cobb [1990] suggested dramatically increasing the mutation rate of a GA after a change occurred, while Vavak *et al.* [1997] advocated a more gradual increase. Both these algorithms can be classified under the Change Control Parameters strategy as the mutation rate is one of a GA's control parameters. A high mutation rate encourages exploration by introducing genetic material from currently unexplored regions of the fitness landscape. This is useful after a change in the environment since optima would, potentially, have shifted to unpopulated areas of the search space.

Later algorithms sustain high diversity throughout the optimisation process as opposed to only increasing diversity directly after changes in the environment. Approaches aimed at maintaining a high amount of diversity during the entire execution of the GA algorithm include Grefenstette's random immigrants algorithm [Grefenstette, 1999], which introduces random individuals into a GA's population after each generation. The random individuals act as a steady source for new genetic material which assists in locating new optima. This approach can be classified as a Refresh Information strategy. Morrison [2004] proposed an algorithm which, in contrast to introducing random individuals, makes use of sentinels that are uniformly distributed throughout the search space to increase diversity. Sentinels are used in selection and crossover to create individuals for the next generation, but are not replaced or mutated. Sentinels placed in a grid, rather than random placement, guarantee a uniform sampling of the fitness landscape, making it more likely that the algorithms will be able to detect and respond to changes in the environment. The sentinels introduce diverse genetic material into the population that would otherwise have converged to a single point. This algorithm is classified under the Sentinels strategy. A disadvantage of Morrison's sentinel approach is that a large number of sentinels must be maintained to achieve a useful coverage of the fitness landscape. Function evaluations are effectively wasted on sentinels that are, for the most part, located in inferior regions of the fitness landscape.

The thermodynamic GA [Mori *et al.*, 1996], [Mori *et al.*, 1997], [Mori *et al.*, 1998]

explicitly controls the population's diversity throughout the optimisation process by selecting individuals for the next population based not only on their fitness, but also on the rarity of their genes. This Selection strategy helps to prevent the convergence of the population to a single point, thus improving the chance of locating new optima after a change in the environment.

Utilising the search history with a History strategy is useful in dynamic optimisation problems where the shape of the fitness landscape is oscillating or cyclic, i.e. changes in the dynamic environment will result in the environment's returning to the same configuration at a future stage. For these problems, maintaining an explicit memory of good solutions has been found to be especially useful. Ramsey and Grefenstette [1993] made use of a knowledge base of individuals that performed well in previous generations. These individuals are inserted into the population when a change occurs that results in a previously seen environment. Yang [2005] made use of a memory of individuals which are used as memory-based immigrants. During each generation, the individuals in the memory are re-evaluated and the best memory individual is repeatedly mutated to create a number of immigrant individuals. These immigrants then replace the worst-performing individuals in the normal population. This results in a constant influx of genetic material from previously found good solutions, which is useful when optima reappear at previous positions.

Algorithms typically use a memory archive of finite size. Simões and Costa [2009a] showed that the optimal memory size is algorithm and problem dependent. Once the memory reaches its maximum size, the algorithms must replace an element in the memory when a new element is added. In general, strategies for replacing memory individuals aim to keep the memory population as diverse as possible. Typical strategies include, replacing the memory individual that is spatially closest to the new individual, or removing the individual that will result in the highest variance in spatial positioning in the memory archive. Zhu *et al.* [2011] showed that the typical memory updating strategies may fail to accomplish the goal of high memory diversity since the same memory index can successively be replaced. Zhu *et al.* [2011] accordingly suggests a memory archive update strategy which replaces most similar individuals (if the new individual is more fit than the memory individual), but also keeps track of how often each memory archive index was updated in

the recent past. If an index was updated more than a specified number of times, the new individual rather replaces the oldest memory individual. This History strategy prevents the successive replacement of the same individual and was shown to result in a more diverse archive population.

Simões and Costa [2009b] used a memory of good individuals, each associated with a particular state of the environment (the location of a good optimum in a particular time in the dynamic environment). The memory individuals are used to predict when changes in the environment will occur and also what state the environment is likely to assume. Nonlinear regression is used to predict when changes are likely to occur. Markov chains are used to predict the next state of the environment. Individuals from the memory archive which represent the predicted state are introduced into the main population after a prediction. The memory individuals thus serve to guide the main population to solutions found earlier in the search process. The algorithm is consequently classified as using a History strategy. The techniques proposed by Simões and Costa [2009b] to predict the environment state will only work when a pattern between changes in the environment exists and if the environment can assume a relatively small number of states (roughly equal to the number of states stored in the Markov chain).

Memory is employed in the majority of algorithms to retain information regarding the location of optima in the environment before a change occurred, especially in non-cyclic dynamic environments. The goal is to track optima through the fitness landscape as changes occur. This is generally achieved by using multiple, independent populations to locate various optima. A key feature of these approaches is that independent populations are allowed to search for optima in parallel. Three of the seminal GA algorithms employing this strategy are self-organizing scouts (SOS) [Branke *et al.*, 2000][Branke, 2002][Branke and Schmeck, 2003], shifting balance GA (SBGA) [Oppacher and Wineberg, 1999] and the multinational GA (MGA) [Ursem, 2000]. All three of these approaches make use of a custom technique to intelligently distribute individuals among the populations.

SOS uses a large base population to search for optima in the fitness landscape. When an optimum is located, a small scout population is left to guard and further optimise the optimum while the base population continues to search for new optima. The area guarded by the scout population is then excluded from the fitness landscape of the base population

by reinitialising base population individuals that stray within a threshold distance from the best individual in the scout population. Diversity within scout populations is maintained by using a mutation step-size which is proportional to the area of the search space covered by the scout population (a Change Control Parameters strategy). This ensures that individuals from a scout population do not converge completely, but explore a relatively small area of the search space.

Individuals are distributed between the various scout populations and the base population based on a quality measure calculated for each sub-population. The quality measure depends on the proportional fitness and proportional increase in fitness in the last generation of each sub-population in comparison with the other sub-populations. Each population is allocated a subset of the total number of individuals equal to the proportion of the quality measure of the population over the sum of the quality measures of all populations. Populations with a high fitness and recent large improvements in fitness are thus allocated more individuals. Maximum and minimum population sizes of the base and scout populations are parameters to the algorithm. SOS can be classified as using a Multiple Population strategy by virtue of the base and scout populations.

SOS aims to keep the bulk of the individuals in a single base population searching for new optima, while SBGA groups a single core population around the best found optima and uses smaller populations, called colonies, to search for new optima. The purpose of a SOS scout population is to exploit an optimum that was found by the base population, while the goal of a SBGA colony population is to find new optima. Colonies are kept away from the core population by selecting parents for reproduction based on their distance from the core population. By selecting parents that are far from the core population, their offspring are generally also located far from the core population. SBGA thus uses a Selection and a Multiple Population strategy. The information contained in the colony populations is shared with the core population by means of migrant individuals that are periodically transferred from colonies to the core population. This influx of genetic material from the colonies can cause the core population to shift its location to promising regions of the fitness landscape discovered by colonies. Alternatively, if migrant individuals have poor fitness values, the selection operators will not consider these migrant individuals for parents, and these individuals will consequently die out. The SBGA process allows for

the broad exploration of the fitness landscape and the discovery of several optima by the colonies. Optima are consequently tracked by the colonies, which assists in the location of new global optima after changes in the environment occur.

The MGA algorithm [Ursem, 2000] allocates individuals to sub-populations based on the optimum on which each individual is located. MGA uses *hill-valley detection* to form sub-populations. Hill-valley detection randomly samples points between two individuals to determine whether the two individuals are located on the same optimum in order to group individuals into sub-populations. Multiple optima are detected if any of the intermediate points that are sampled yields a lower fitness value than the two individuals (assuming maximisation). Groupings ensure that each sub-population resides on a different optimum.

Parent individuals selected for the creation of offspring for a given sub-population are not only selected from that sub-population but also from other sub-populations (this is in contrast to SOS and SBGA). When offspring are created, hill-valley detection determines whether each offspring individual is to remain in the current sub-population, whether the offspring individual should join another sub-population, or whether the offspring individual should be placed in an entirely new sub-population. Sub-populations are prevented from losing all their individuals by a selection operator that divides the fitness of an individual by its sub-population size. Small sub-populations thus have an evolutionary advantage over larger sub-populations (since a smaller denominator is used) and are likely to continue to exist and contribute genetic material to successive generations. Independent sub-populations increase diversity, indirectly store information regarding the shape of the fitness landscape, and contribute to locating optima in parallel. MGA uses the Selection and Multiple Populations strategies.

3.3.3.2 Particle Swarm Optimisation-Based Algorithms

Considerable success has been achieved in applying modern optimisation algorithms, such as PSO, to dynamic optimisation problems. One of the first investigations into using PSO in dynamic environments is the algorithm suggested by Carlisle and Dozier [2000] wherein the personal best value of particles were periodically reset to their current position. This has the effect of refreshing information about the environment and encouraging particles to explore other parts of the search space (the algorithm thus uses the Refresh Information

strategy). This algorithm was subsequently improved by the authors [Carlisle and Dozier, 2002] by first determining whether the current position is in fact better than the existing personal best value before resetting the value.

The Refresh Information strategy was also used in another of the early PSO-based algorithms, created by Hu and Eberhart [2002]. The authors suggested that diversity should be increased by reinitialising the particles in the PSO algorithm when a change in the environment occurs.

Later diversity-increasing approaches include charged particles [Blackwell and Bentley, 2002], where each particle of a PSO is assigned a virtual charge and is then allowed to repel other particles based on the laws of electrostatics. The repulsive force between particles prevents the entire population from converging to a single point, which allows the discovery of multiple optima and supports exploration after a change in the environment. This is an example of an algorithm employing the Custom Individuals strategy. Another example of this strategy is the idea of increasing diversity by reinitialising a portion of the swarm of particles within a hyper-sphere, centred around the best particle within the swarm [Blackwell and Branke, 2004, 2006]. These particles are called *quantum particles* and were implemented in a PSO algorithm. The cloud radius determines how far from the best particle the quantum particles are dispersed.

Investigations into finding an appropriate neighbourhood structure for PSO [Li and Dam, 2003], [Janson and Middendorf, 2004] have been conducted, since these parameters greatly affect the diversity of the swarm. For example, using a star network topology in PSO results in faster convergence but greater loss of diversity than using a ring social structure where information regarding the global best position is spread through the swarm more gradually. In general, a neighbourhood structure that increases diversity is more effective in dynamic environments, since diverse particles are able to explore the fitness landscape more effectively after a change in the environment. These approaches are classified under the Change Control Parameters strategy.

The dynamic heterogeneous PSO (DHPSO) is an algorithm based on the idea that particles in the swarm do not necessarily need to follow the same behaviour, or even maintain the same behaviour during the entire optimisation process [Leonard *et al.*, 2011]. During a particular period in the optimisation process it may, for example, be useful for a

subset of the particles to focus on exploitation, while the other particles perform a more explorative role. DHPSO allows particles that have stagnated (i.e. particles that have not shown any improvement for ten iterations) to randomly select a new behaviour from one of the following five options: standard PSO, social PSO (the cognitive component of the standard PSO is discarded), cognitive PSO (the social component of the standard PSO is discarded), bare bones PSO [Kennedy, 2003], and modified bare bones PSO [Kennedy, 2003]. This process allows the particle behaviour to adapt to the most appropriate model during different stages of the optimisation process. DHPSO responds to changes in the environment by reinitialising half of all particles when change occurs and re-evaluating the other half to inhibit the problem of outdated information. The algorithm thus utilised the Change Control Parameters and Refresh Information strategies.

Parrott and Li [2004] suggested a multiple swarm PSO approach to solving DOPs, called speciation. Multiple swarms correspond to the idea of multiple populations in GAs, and have the same benefits: increased diversity and parallel discovery of optima. Speciation can thus be classified as a Multiple Populations strategy. The social component of PSO provides a simple method to divide the swarm into sub-swarms when using speciation. A particle is allocated to a sub-swarm if the Euclidean distance between the position of the particle and the best particle in the sub-swarm (referred to as the species seed) is below a certain threshold value. Species seeds are determined by processing a list of all particles sorted in descending order of fitness. The set of species seeds is constructed by adding each particle, encountered in the list, that is not within the threshold distance from any of the particles already in the set of species seeds [Parrott and Li, 2006]. The global best value of each particle within a sub-swarm is set to the personal best value of the species seed. The sizes of sub-swarms are dynamic. Particles can migrate to another sub-swarm by moving too far away from their current sub-swarm's best particle, or by moving closer to another sub-swarm's best particle. It is allowable for a sub-swarm to contain only one particle in which case the particle merely acts as scout guarding an optimum. As a mechanism to prevent sub-swarms from becoming too large, a maximum sub-swarm size is defined. When a sub-swarm's size exceeds this limit, the particles with the lowest fitness are randomly reinitialised and allocated to appropriate sub-swarms. The Refresh Information strategy is used by this algorithm when reinitialising particles, and also by re-evaluating the personal

best value of each particle after each generation. The speciation algorithm provides a natural mechanism for particles to form, join and split sub-swarms.

Blackwell and Branke [2006] introduced a multiple swarm PSO-based algorithm (MPSO) that is based on three components: exclusion, anti-convergence and quantum individuals. In contrast to earlier algorithms, all sub-swarms contain the same number of particles. The aim of having multiple sub-swarms is that each sub-swarm should be positioned on its own, promising optimum in the environment. Unfortunately, sub-swarms often converge to the same optimum, hence decreasing diversity. Exclusion [Blackwell and Branke, 2004] is a technique meant to prevent sub-swarms from clustering around the same optimum by means of reinitialising sub-swarms that stray within a threshold Euclidean distance from better performing sub-swarms. This threshold distance is called the exclusion radius which is calculated as $r_{excl} = (V_{max,F} - V_{min,F})/(2{n_p}^{\frac{1}{n_d}})$ where $V_{max,F}$ and $V_{min,F}$ are the upper and lower search range of function F in the n_d dimensions (assuming equal ranges for all dimensions), and n_p is the number of optima.

The anti-convergence component of the algorithm was incorporated to prevent stagnation of the particles in the search space. This is achieved by reinitialising the weakest sub-swarm if it is found that all sub-swarms have converged. Convergence of a sub-swarm is detected if all particles within the sub-swarm fall within a threshold Euclidean distance of each other. This threshold is called the convergence radius. MPSO thus uses the Multiple Populations, Refresh Information (through anti-convergence reinitialisation), and Custom Individuals (by merit of the quantum individuals) strategies.

MPSO has the disadvantage that the severity of changes in the environment is a parameter to the algorithm which is used to calculate the cloud radius. While this information is readily available when employing a benchmark function, it is unlikely that this information will be known to the algorithms in real-world dynamic optimisation problems.

The MPSO algorithm was adapted and improved by del Amo *et al.* [2010] to reinitialise or pause sub-swarms that perform badly. For each swarm a history of improvements (the difference in fitness from successive iterations) is stored. A swarm is flagged as performing badly if, for five iterations, its improvement falls below 15% of its best improvement since the last change in the environment and if the swarm is in the set containing the 20% of swarms that have the lowest fitness value. Once a swarm is thus flagged, the algorithm,

using a history of previous changes, then estimates the number of iterations before the next change in the environment. Should less than 20% of the estimated period between changes remain, the swarm is paused until the next change in the environment (hence preventing it from wasting function evaluations). Otherwise, the swarm is reinitialised to continue searching for new optima. This algorithm thus uses the Multiple Populations, Refresh Information, and Custom Individuals strategies present in MPSO and incorporates a History strategy. Disadvantages of this algorithm include the fact that equally sized periods between changes in the environment are assumed and that constants used by the algorithm are problem dependent. For example, using 20% of the period between changes to determine whether to reinitialise or to pause the swarm may not be an effective choice for all environments. An environment in which the change period is large may enable a swarm to locate a new optimum in less than 20% of the change period, in which case the swarm should be reinitialised rather than paused. Conversely, reinitialisation of the swarm would be less likely to lead to the discovery of an optimum in an environment with a low change period, and a value larger than 20% should be used as the cutoff for pausing unproductive sub-swarms.

Blackwell [2007] adapted MPSO of Blackwell and Branke [2006] by self-adapting the number of swarms in the search space. This algorithm (referred to as MPSO2) is aimed at situations where the number of optima in the dynamic environment is unknown. Swarms are generated when the number of free swarms that have not converged to an optimum (M_{free}) have dropped to zero. Conversely, swarms are removed if M_{free} is higher than n_{excess}, a parameter of the algorithm. A swarm is classified as converged if all particles are located within a diameter of $2r_{conv}$. The values of r_{conv} and r_{excl} are calculated using $r_{excl} = r_{conv} = (V_{max,F} - V_{min,F})/(2n_k^{\frac{1}{n_d}})$, where n_k is the number of sub-swarms, $V_{max,F}$ and $V_{min,F}$ is the upper and lower search range of function F in the n_d dimensions (assuming equal ranges for all dimensions). MPSO2 can find an effective value for the number of sub-swarms by introducing and removing a sub-swarm when necessary. This algorithm thus uses the Multiple Populations, Refresh Information, and Custom Individuals strategies present in MPSO and incorporates a Change Control Parameters strategy.

Blackwell *et al.* [2008] adapted MPSO2 by converting all particles to quantum particles for one iteration after a change in the environment occured. Experimental results showed

that this adaptation requires fewer quantum particles during the period between changes in the environment. By evaluating fewer particles during each iteration, the number of iterations that can be performed between changes in the environment is increased.

Li *et al.* [2006] improved the speciation algorithm of Parrott and Li [2004] by introducing ideas from [Blackwell and Branke, 2004], namely quantum individuals to increase diversity, and anti-convergence to detect stagnation and subsequently reinitialise the worst-performing populations. A Custom Individuals strategy was thus added to the algorithm of Parrott and Li [2004]. This algorithm is called Speciation-based PSO (SPSO). The same adaptation that was made to MPSO2, namely converting all particles to quantum particles for one iteration after a change in the environment and reducing the number of quantum particles for the other iterations, was suggested for SPSO [Blackwell *et al.*, 2008].

Cellular PSO [Hashemi and Meybodi, 2009a] incorporates ideas from cellular automata into a PSO algorithm aimed at DOPs. The search space is divided into equally sized cells. Particles are allocated to cells based on their spatial positions within the search space. The particles within each cell perform a local best search based on the best performing particle within the cell and the cell's neighbourhood. A particle migrates from one cell to another when position updates place it outside the bounds of its original cell. A threshold is used to prevent too many migrating to a specific cell. When the number of particles within a cell exceeds the threshold, randomly selected particles from the overcrowded cell are allocated to randomly selected cells within the search space. Parameters to the algorithm include the number of cells, the neighbourhood size, and the maximum number of particles per cell threshold. The motivation for partitioning particles into cells is similar to that of forming sub-swarms and also provides the benefits of parallel search and tracking multiple optima. Cellular PSO is thus classified as using a Multiple Populations strategy and, through reinitialising particles, a Refresh Information strategy.

A Custom Individuals strategy was incorporated into Cellular PSO by Hashemi and Meybodi [2009b]. This adaptation changes a portion of each cell's particles to quantum particles for a specified number of iterations after a change in the environment. The fraction of particles that are changed to quantum particles, and the number of iterations that quantum particles must be maintained, are parameters to the algorithm.

Kamosi *et al.* [2010b] proposed an algorithm that utilises a single parent swarm to

locate promising areas in the search space and a dynamic number of child swarms to exploit the promising areas. This multi-swarm optimisation algorithm is referred to here as MSO. MSO commences with zero child swarms. The local best particle of the parent swarm is monitored and each time that it improves, a child swarm is created around it. All the particles from the parent swarm, that are located within a certain radius, r_{cs}, of the local best particle, migrate to the newly created child swarm. If the migrating particles exceed the maximum child swarm size, the surplus particles are reinitialised. Conversely, if too few particles migrate from the parent population, particles are randomly created using a uniform distribution within a radius of $r_{cs}/3$ from the local best particle of the child swarm. Randomly initialised particles are added to the parent swarm to replace particles lost in the migration process. Child swarms optimise their respective areas of the fitness landscape. Exclusion is used to prevent child swarms from converging to the same optimum.

A particle from the parent swarm that strays within a threshold distance, r_{cs}, of the local best particle of a child swarm is compared to the local best particle in terms of fitness. If the parent swarm particle has a higher fitness, a copy of it replaces the local best particle within the child swarm before the parent swarm particle is reinitialised. The algorithm reacts to changes in the environment by re-evaluating all parent swarm particles. Particles in child swarms act as quantum particles for one iteration after a change in the environment. MSO thus uses the Multiple Populations, Custom Individuals and Refresh Information strategies. Besides the normal PSO parameters, MSO has the following parameters: the number of particles in the parent swarm, the number of particles in each child swarm, the child swarm radius (r_{cs}), the exclusion radius, and the cloud radius.

Kamosi *et al.* [2010a] improved MSO by introducing a hibernation metaphor to place child swarms into a "sleep" state when they become unproductive. This approach (referred to here as hibernating MSO (HMSO)) is motivated by the fact that function evaluations are wasted on a child swarm which have located the summit of the optimum that it tracks. Function evaluations are freed up for other swarms by stopping the optimisation process for child swarms that have converged. A child swarm hibernates if the following two conditions are both satisfied: Firstly, the difference in fitness between the local best solution in the child swarm and the global best solution found in the fitness landscape

exceeds a specified threshold (i.e. the swarm has a comparatively low fitness). Secondly, the swarm radius is smaller than a threshold value (the swarm has converged). Swarms are awakened from hibernation when a change in the environment occurs.

Kamosi *et al.* [2010a] showed that HMSO performs better than MSO on the majority of the benchmark instances that were investigated. The HMSO algorithm yields lower errors than MSO because, by hibernating weaker sub-swarms that have converged, more function evaluations are made available to other swarms to discover and exploit new optima. A disadvantage of HMSO is that it introduces two new parameters: the convergence radius and threshold difference between the local and global best solution fitness values that must be exceeded.

Novoa-Hernández *et al.* [2011] removed the quantum particle component of the MPSO algorithm of Blackwell and Branke [2006]. The authors introduced a diversity increasing scheme directly after a change in the environment by replacing the worst-performing particles within each sub-swarm (half of the sub-swarm was replaced). The replacement particles are randomly created in a hypersphere centred at the best particle in the sub-swarm formed using a uniform distribution with a specified radius. Novoa-Hernández *et al.* [2011] also introduced a control rule which allows bad sub-swarms to "sleep" and consequently save function evaluations. This algorithm is referred to as MPSOD.

Sub-swarms are placed in a sleep state (i.e. temporarily removed from the optimisation process) if the following three conditions hold. Firstly, no recent changes occurred in the environment. Secondly, the sub-swarm must have converged to within a specified radius. Thirdly, the fitness of the sub-swarm must be classified as "bad". Membership to a fuzzy set of bad sub-swarms is determined using a function which assigns a "badness" value to each sub-swarm based on the average fitness of all the sub-swarms (fit_a) and the most fit sub-swarm (fit_b). Sub-swarms with a fitness below average is assigned a badness value of one. Otherwise the badness value is equal to the ratio $(fit_b - fit_k)/(fit_b - fit_a)$ where fit_k is the fitness of the sub-swarm that is being classified. Sub-swarms for which the badness value is smaller than a specified threshold are classified as bad. The sleeping swarms of MPSOD is similar to the hibernating swarms of HMSO and only differs in when the swarms are removed from the optimisation process. Placing sub-swarms in a sleep state allows more function evaluations to be allocated to other swarms, which leads to

more exploitation of optima that have been found and the discovery of new optima. This algorithm uses the same strategies as MPSO: Multiple Populations, Refresh Information, and Custom Individuals.

The clustering PSO (CPSO) algorithm of Yang and Li [2010] clusters the main swarm into independent sub-swarms after each change in the environment. A Multiple Populations strategy is thus used. The particles are allocated to sub-swarms based on their spatial proximity. The total number of particles and the maximum sub-swarm size are parameters to the algorithm. CPSO transfers information from one environment to the next by means of a memory archive containing good solutions found before the change (a History strategy). The best particle within a sub-swarm is added to the memory archive when the convergence of a sub-swarm is detected (all particles fall within a threshold radius). Converged sub-swarms are subsequently removed. Sub-swarms which search overlapping areas of the search space are detected by calculating the percentage of particles from one sub-swarm that fall within the search area of another. Two sub-swarms are merged if the percentage of overlap exceeds a threshold value, whereupon the weakest excess particles are removed to prevent the size of the merged sub-swarm from exceeding the maximum sub-swarm size.

Yang and Li [2010] also use a custom replacement strategy for the best particle within each sub-swarm. When a new particle with a better fitness than the best particle within the sub-swarm is found, components from each dimension of the new particle are iteratively incorporated into the old best particle and evaluated. A dimensional component is only permanently incorporated into the best particle if it results in an improved fitness. The motivation for this strategy is that it prevents promising information from some of the dimensions from being lost due to bad information in other dimensions. The approach is classified as a Selection strategy as particles are not automatically flagged as the best particle within the sub-swarm based on fitness.

CPSO responds to changes in the environment by removing all sub-swarms, reinitialising the main swarm and replacing the weakest particles in the newly created main swarm with the particles in the memory archive. The memory archive is subsequently cleared and the clustering process is repeated to form sub-swarms.

CPSO differs from most other algorithms in that the total number of particles dimin-

ishes after each change in the environment due to the removal of sub-swarms that have converged and the removal of excess particles when sub-swarms are merged. The algorithm prevents a situation where zero particles remain by introducing random particles into the search space when all particles have been removed.

The main ideas from CPSO where extended by Li and Yang [2012] to form CPSOR, an algorithm aimed specifically at situations where changes in the environment are difficult or impossible to detect. CPSOR uses the same techniques for clustering and removing redundant particles as CPSO, but in contrast to CPSO, changes in the environment are not explicitly detected or responded to by CPSOR. A threshold value, which is a parameter to the algorithm, is used as a minimum ratio of current population size over the original population size. Random particles are introduced when the ratio falls below the the threshold value, whereupon the clustering and particle removal processes are allowed to proceed as in CPSO. A strength of CPSOR is that it does not rely on detecting changes. CPSOR was shown to be more effective than several other algorithms in environments where changes occurred only locally in the fitness landscape (as opposed to changes that affect the entire fitness landscape and are consequently easier to detect). A weakness of CPSOR is that it introduces a new parameter which may be problem dependent.

3.3.3.3 Differential Evolution-Based Algorithms

The first application of the DE algorithm to DOPs occurred relatively recently. Consequently, researchers could apply earlier successful DOP approaches from GA and PSO-based algorithms when adapting DE for dynamic environments.

Section 3.3.2 described how Custom Individuals can be used to increase diversity. An approach similar to quantum particles, called Brownian individuals, is utilised by DynDE, a DE-based algorithm for dynamic environments [Mendes and Mohais, 2005]. Use of Brownian individuals involves the creation of individuals close to the best individual by adding a small random value, sampled from a zero-mean normal distribution, to each component of the best individual. Mendes and Mohais [2005] adapted the ideas from [Blackwell and Branke, 2004] to create their multi-population algorithm, DynDE, which uses exclusion to prevent populations from converging to the same peak. Exclusion reinitialises a sub-population when it strays too closely to another and can thus be seen as a

Refresh Information strategy. Furthermore, it was determined that the DE/best/2/bin best scheme is the most effective for use by DynDE (a Change Control Parameters strategy). The DE/best/2/bin scheme actually reduces diversity, since DE/best/2/bin encourages fast convergence. However, since multiple, independent populations are used which increase diversity, it was found to be beneficial for sub-populations to converge quickly to their respective optima. A Multiple Populations strategy is thus also used in DynDE. Mendes and Mohais [2005] showed that DynDE was at least as effective as its PSO-based counterparts. DynDE is discussed in detail in Section 3.4.1.

The favoured populations DE (FPDE) [du Plessis and Engelbrecht, 2008] is an adaptation to DynDE created in a preliminary study to the algorithms presented in this thesis. The adaptation aims to prevent wasting function evaluations on sub-populations with inferior performance. Normal DynDE is executed for ϱ_1 generations after a change in the environment. Thereafter, only the ϱ_3 sub-populations with the lowest error are executed for ϱ_2 generations while all other sub-populations are paused. Normal DynDE resumes after the ϱ_2 generations. The parameters ϱ_1, ϱ_2 and ϱ_3 must be manually tuned and it was found that FPDE yields only minor improvements over DynDE.

Brest *et al.* [2009] proposed a self-adaptive multi-population DE algorithm, called *jDE*, for optimising dynamic environments. This work focused on adapting the DE scale factor and crossover probability and is based on a previous algorithm [Brest *et al.*, 2006] for optimising static environments. The motivation behind using self-adaptive control parameters is that the algorithm can change the scale factor and crossover probability during the optimisation process to appropriate values as the environment changes. A further benefit is that the self-adaptive appoach results in fewer parameters to tune manually. This approach is classified as a Change Control Parameters strategy. *jDE* also contains components that are similar to other dynamic optimisation algorithms. For example, exclusion is used to prevent sub-populations from converging to the same optimum (a Refresh Information strategy). An aging metaphor is used to reinitialise sub-populations that have stagnated on a local optimum. Each individual's age is incremented every generation. Offspring inherit the age of parents, but this age may be reduced if the offspring performs significantly better than the parent. Sub-populations of which the best individual is too old are reinitialised so that the sub-population can discover other optima. The aging mechanism

allows the algorithm to discover new optima continuously and is classified as a Refresh Information strategy.

A further mechanism is used to prevent convergence within sub-populations: An individual is reinitialised if the Euclidean distance between the individual and the best individual in the population is too small, thus increasing diversity. The algorithm also utilises a form of memory called an archive. The best individual is added to the archive every time a change in the environment occurs. One of the sub-populations is always created by randomly selecting an individual from the archive and by adding small random numbers to each of the individual's components. The archive is a History strategy and information from the archive is incorporated through a Custom Individuals strategy. *jDE* is discussed in more detail in Section 3.4.2.

Recently, Noroozi *et al.* [2011] proposed a DE-based algorithm aimed at dynamic environments, referred to as Cellular DE. Cellular DE is a DE implementation of Cellular PSO which was discussed in the previous section. This algorithm divides the search space into equally sized cells which are used to delineate sub-populations (a Multiple Populations strategy). A limit is placed on the maximum number of individuals that are allowed to be located in each cell. Once this limit is exceeded, the worst performing individual within the overpopulated cell is reassigned to a randomly selected cell. The random reassignment of individuals is classified as a Refresh Information strategy. The algorithm employs a variant of the DE/current-to-best/bin scheme to create mutant vectors (a Change Control Parameter strategy). The best individual within the target vector's cell and surrounding cells is used as $\vec{x}_{best}$ in the scheme. Cellular DE track multiple optima in the dynamic environment by allowing individuals to converge to cells representing the areas where optima occur.

3.3.3.4 Other Dynamic Environment Algorithms

This section describes algorithms for dynamic environments that are either hybrids of GA, PSO and DE algorithms, or those that are not based on these algorithms. Usage of the seven strategies of Section 3.3.2 is indicated where possible.

An algorithm that uses both a DE population and a PSO swarm was proposed by Lung and Dumitrescu [2007]. This algorithm is called collaborative evolutionary-swarm

optimisation (CESO) and uses a Multiple Populations strategy. A swarm optimised by using PSO refines the best found optimum, while a DE variant, called crowding differential evolution, is used to maintain diversity and locate local optima. Crowding DE [Thomsen, 2004] is an algorithm aimed at locating multiple optima in static environments. Crowding DE changes the offspring operator of DE so that the most similar individual in the parent population, rather than the target vector, is replaced. This idea is similar to an approach used in niching, where the fitness of an individual is divided by a sharing function which is proportional to the number of other individuals within a threshold distance of the individual [Goldberg and Richardson, 1987][Sareni and Krahenbuhl, 1998], and is classified as a Selection strategy. Crowding DE maintains a highly diverse population in which each individual is ideally located on an optimum. Multiple optima are thus tracked simultaneously. The particles in the PSO swarm are replaced by the individuals in the DE population when a change in the environment occurs or when the distance between the best particle in the swarm and the best individual in the population drops below a threshold value.

CESO was improved by incorporating a second DE population [Lung and Dumitrescu, 2010]. The new algorithm is referred to as the evolutionary swarm cooperative algorithm (ESCA). The second DE population is used to maintain a search history and acts as a memory of previously found optima for the first DE population. ESCA thus also uses a History strategy.

Several investigations have focused on the self-adaptation of control parameters (a Change Control Parameters strategy). Angeline *et al.* [1996] made use of a self-adaptive evolutionary programming (EP) algorithm to evolve finite machines to predict the next value of an input string. The string to be predicted was changed dynamically during the execution of the algorithm. Evolution strategies (ES) have been applied to DOPs by several researchers [Arnold and Beyer, 2002], [Weicker and Weicker, 1999], [Bäck, 1998].

Moser and Hendtlass [2007] proposed the multi-phase multi-individual extremal optimisation (MMEO) algorithm. Extremal optimisation (EO) [Boettcher and Percus, 1999] makes use of a single individual which is mutated. EO consequently finds the optimum of the fitness landscape through hill climbing. EO was adapted by Moser and Hendtlass [2007] to contain several individuals, each of which used five steps to find optima. Firstly,

a stepwise sampling of the fitness landscape is performed to locate areas that potentially contain optima. From these potential points, an individual performs a local search to find a local optimum. During the local search phase, the individuals are checked to ensure that no duplicates (individuals that are optimising the same peak) exist. If changes in the environment occur, individuals are optimised further using a local search. Finally, individuals are fine tuned using finer-grained hill climbing. Multiple individuals were introduced by Moser and Hendtlass [2007] to locate and track multiple optima in the dynamic environment. Accordingly MMEO can be classified as using the Multiple Populations strategy.

Moser [2009] refined the MMEO algorithm by making use of the Hooke-Jeeves search algorithm [Hooke and Jeeves, 1961] to perform the local search. Further improvements to the MMEO local search was reported in [Moser and Chiong, 2010] by tuning the step sizes to more problem appropriate values. Currently, Moser and Chiong [2010] report the lowest offline error on the moving peaks benchmark in the literature.

3.3.4 Summary

The previous sections described how sub-components of each algorithm can be categorised into the seven strategies identified in Section 3.3.2. This information is summarised in Table 3.1. The table shows that most of the algorithms reviewed can be classified as using at least one of the seven strategies.

The majority of the algorithms employ several of the strategies to realise the three main approaches to solving DOPs: increasing diversity, memory, and parallel search. As the field matured over time, strategies like Multiple Populations and Refresh Information have been used in an increasing number of algorithms, although the specific implementation details still vary considerably. Conversely, the Sentinel strategy has not been widely adopted.

Despite the large number of available algorithms the problem of optimisation in dynamic environments has not been completely solved. New research which reports on improvements over previous algorithms is continuously published and the field remains an active area of research internationally. DE, in comparison to GA and PSO, has only been applied to DOPs relatively recently and there is still a comparatively small number of algorithms that uses DE as a base. The potential of improving the current algorithms thus exists and is the focus of this research.

Table 3.1: General strategies followed per algorithm

Algorithm	Change Control Parameters	Custom Individuals	Refresh Information	Sentinels	Selection	History	Multiple Populations
Angeline *et al.* [1996]	✓						
Blackwell and Bentley [2002]		✓					
Blackwell and Branke [2006] (MPSO)		✓	✓				✓
Blackwell [2007] (MPSO2)	✓	✓	✓				✓
Branke [2002], Branke and Schmeck [2003] (SOS)	✓						✓
Brest *et al.* [2009] (*jDE*)	✓	✓	✓			✓	✓
Carlisle and Dozier [2000]			✓				
Cobb [1990]	✓						
del Amo *et al.* [2010]		✓	✓			✓	✓
Grefenstette [1999]			✓				
Hashemi and Meybodi [2009b] (Cellular PSO)		✓	✓				✓
Hu and Eberhart [2002]			✓				
Kamosi *et al.* [2010b] (MSO)		✓	✓				✓
Kamosi *et al.* [2010a] (HMSO)		✓	✓				✓
Leonard *et al.* [2011]	✓		✓				
Li and Dam [2003], Janson and Middendorf [2004]	✓						
Li *et al.* [2006] (SPSO)		✓	✓				✓
Lung and Dumitrescu [2007] (CESO)					✓		✓
Lung and Dumitrescu [2010] (ESCA)					✓	✓	✓
Mendes and Mohais [2005] (DynDE)	✓	✓	✓				✓
Mori *et al.* [1996], Mori *et al.* [1997], Mori *et al.* [1998]					✓		
Morrison [2004]				✓			
Moser and Hendtlass [2007], Moser and Chiong [2010]							✓
Noroozi *et al.* [2011] (Cellular DE)	✓	✓	✓				
Novoa-Hernández *et al.* [2011]		✓	✓				✓
Oppacher and Wineberg [1999] (SBGA)					✓		✓
Parrott and Li [2004]			✓				✓
Ramsey and Grefenstette [1993]						✓	
Simões and Costa [2009b]						✓	
Ursem [2000] (MGA)					✓		✓
Vavak *et al.* [1997]	✓						
Yang [2005]						✓	
Yang and Li [2010]					✓	✓	✓
Zhu *et al.* [2011]						✓	

3.4 Differential Evolution for Dynamic Optimisation Problems

The previous section gave an outline of common strategies used by algorithms aimed as DOPs and gave an overview of specific algorithms for dynamic environments. The focus of this study is on DE and this section consequently gives a more detailed description of two DE-based algorithms for solving DOPs. The first is DynDE which is described in Section 3.4.1. DynDE is used as the base algorithm for extensions and improvements presented in the following chapters. The second algorithm is *jDE* which is a recent state-of-the-art algorithm based on DE. *jDE* is discussed in Section 3.4.2.

3.4.1 DynDE

DynDE is a differential evolution algorithm developed by Mendes and Mohais [2005] to solve dynamic optimisation problems. DynDE makes use of approaches similar to those applied by Blackwell and Branke [2006] to PSO in dynamic environments. The most successful versions of DynDE make use of multiple populations, exclusion and Brownian individuals to adapt DE for dynamic environments. These three components are described in Sections 3.4.1.1 to 3.4.1.3. Section 3.4.1.4 gives a discussion on the DE scheme used by DynDE, followed by a general discussion on the DynDE algorithm in Section 3.4.1.5.

3.4.1.1 Multiple Populations

Typically, a static problem space may contain several local optima. These optima typically move around in a dynamic environment and also change in height and shape. This implies that a local optimum may become a global optimum when a change in the environment occurs. Consequently, not only the movement of global optima in the problem space must be tracked, but also the local optima. An effective method of tracking all optima is to maintain several independent sub-populations of DE individuals, one sub-population on each optimum. In their most successful experiments Mendes and Mohais [2005] used 10 sub-populations, each containing six individuals.

3.4.1.2 Exclusion

In order to track all optima, it is necessary to ensure that all sub-populations converge to different optima. If all populations converged to the global optimum it would defeat the purpose of having multiple populations. Mendes and Mohais [2005] used exclusion to prevent sub-populations from converging to the same optimum. Exclusion compares the locations of the best individuals from each sub-population. If the spatial difference between any two of these individuals becomes too small, the entire sub-population of the inferior individual is randomly reinitialised. A threshold is used to determine if two individuals are too close. This threshold, or exclusion radius, is calculated as

$$r_{excl} = \frac{V_{max,F} - V_{min,F}}{2{n_p}^{\frac{1}{n_d}}} \tag{3.1}$$

where $V_{max,F}$ and $V_{min,F}$ are the upper and lower search range of function F in the n_d dimensions (assuming equal ranges for all dimensions), and n_p is the number of peaks. Equation (3.1) shows that the exclusion threshold increases with an increase in number of dimensions and decreases if the number of peaks is increased.

Let $\vec{x}_{best,k}$ be the best individual in sub-population P_k, $k \in \{1, 2, \ldots, n_k\}$. Exclusion is thus performed as described in Algorithm 7 (assuming a minimisation function, F).

Algorithm 7: DynDE Exclusion

```
for k1 = 1, ..., nk do
    for k2 = 1, ..., nk do
        if ||x_best,k1 − x_best,k2||2 < r_excl and k1 ≠ k2 then
            if F(x_best,k1) < F(x_best,k2) then
                Reinitialise population P_k2
            else
                Reinitialise population P_k1
            end
        end
    end
end
```

3.4.1.3 Brownian Individuals

In cases where a change in the environment results in the positional movement of some of the optima, it is unlikely that all of the sub-populations will still be clustered around the optimal point of their respective optima, even if the change is small. In order for the individuals in the sub-populations to track the moving optima more effectively, the diversity of each population should be increased. Mendes and Mohais [2005] successfully used Brownian individuals for this purpose. In every generation, a predefined number of the weakest individuals is flagged as Brownian. These individuals are then replaced by new individuals created by adding a small random number, sampled from a zero centred Gaussian distribution, to each component of the best individual in the sub-population. A Brownian individual, $\vec{x}_{brown}$, is thus created from the best individual, $\vec{x}_{best}$, using

$$\vec{x}_{brown} = \vec{x}_{best} + \vec{r} \tag{3.2}$$

where $\vec{r}$ is a random vector with $r_j \sim N(0, r_{brown})$ and r_{brown}, the Brownian radius, is the standard deviation of the Gaussian distribution. Mendes and Mohais [2005] showed that a suitable value to use for r_{brown} is 0.2.

The addition of the small random values to the components of the Brownian individuals ensures that the population never completely converges. This is especially important in a DE-based algorithm, since mutation step-sizes depend on the vector differences between individuals. Without the Brownian individuals, it is conceivable that the entire population could converge to a single point in the fitness landscape which would result in all vector differences being zero. The evolution process would cease and the algorithm would be unable to respond to changes in the environment.

3.4.1.4 DE Scheme

Mendes and Mohais [2005] experimentally investigated several DE schemes to determine which one is the best to use in DynDE. The schemes investigated were DE/rand/1/bin, DE/rand/2/bin, DE/best/1/bin, DE/best/2/bin, DE/rand-to-best/1/bin, DE/current-to-best/1/bin and DE/current-to-rand/1/bin. It was shown that the most effective scheme to use in conjunction with DynDE is DE/best/2/bin, where each temporary individual is

created using

$$\vec{v} = \vec{x}_{best} + \mathcal{F} \cdot (\vec{x}_1 + \vec{x}_2 - \vec{x}_3 - \vec{x}_4) \tag{3.3}$$

with $\vec{x}_1 \neq \vec{x}_2 \neq \vec{x}_3 \neq \vec{x}_4$ and $\vec{x}_{best}$ being the best individual in the sub-population.

3.4.1.5 DynDE Discussion

Section 3.3 identified three categories of approaches used by dynamic optimisation algorithms (increasing diversity, memory and parallel search). All three of these categories are present in the subcomponents of DynDE, i.e. Brownian individuals increases diversity and multiple populations are used to maintain a memory of the fitness landscape and to encourage parallel searching.

The performance of DynDE is thoroughly investigated in Chapter 4. However, Figure 3.3 depicts the offline error, current error and diversity of the DynDE algorithm on the MPB with Scenario 2 settings for 10 changes in the environment to illustrate how effective DynDE is on dynamic environments compared to normal DE. Averages over 30 runs were used.

A comparison between Figures 3.1 and 3.3 shows the considerable improvement of DynDE over DE. Firstly, where the diversity of the DE algorithm sharply declines to a point close to zero, the diversity of DynDE remains high. The average diversity per generation over all repeats was found to be 0.2661 for DynDE, compared to the value of 0.0017 for DE. The frequent perturbations on the diversity curve in Figure 3.3 shows how diversity is perpetually increased by Brownian individuals and sub-populations reinitialised by exclusion.

Secondly, the graphs clearly show considerably lower offline and current errors for DynDE than for DE. The offline error illustrated in Figure 3.1 increases after the first few changes in the environment, while DynDE's offline error consistently decreases (see Figure 3.3). The current error of DynDE frequently approaches zero between changes in the environment, while the current error of DE remains high, especially after the first change in the environment.

Despite the relative effectiveness of DynDE, a disadvantage of this algorithm is that several new problem dependent parameters are introduced. A total of four parameters is

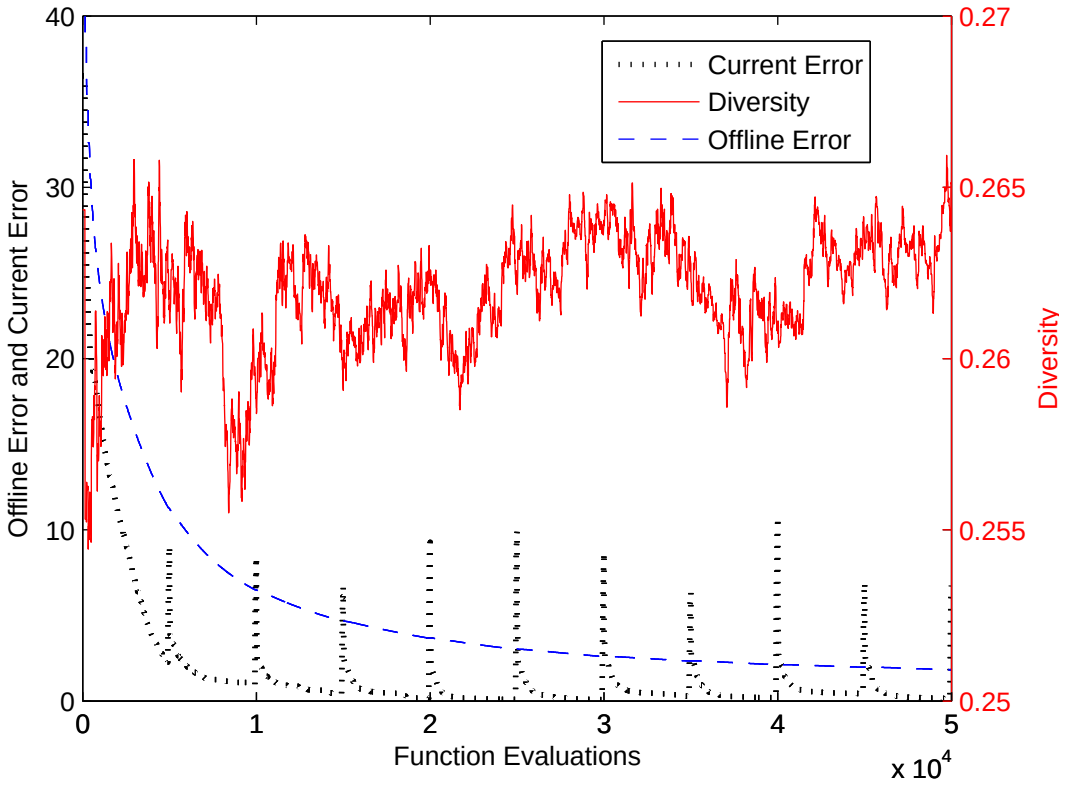

Figure 3.3: Diversity, Current error and Offline error of DynDE on the MPB

present in DynDE that are not present in DE:

1. the size of sub-populations,
2. the number of sub-populations,
3. the number of Brownian individuals per sub-population, and
4. the Brownian radius, r_{brown}, used to create Brownian individuals.

While Mendes and Mohais [2005] experimentally determined guidelines for these parameters, further fine tuning may be required for different problems. Furthermore, equation (3.1) requires knowledge about the number of optima in the environment. Even assuming that a constant number of optima is present, this information is generally not available in real-world problems.

3.4.2 jDE

The 2009 Congress on Evolutionary Computation (CEC2009) ran a dynamic optimisation competition. This competition used the generalised dynamic benchmark generator (GDBG) [Li *et al.*, 2008] to compare the results of competing algorithms. The winning algorithm was *jDE*, developed by Brest *et al.* [2009]. This section describes the five main algorithmic components that are utilised by *jDE* to solve dynamic optimisation problems. The components, self-adaptive control parameters, multiple populations, aging, localised exclusion, and custom reinitialisation are respectively discussed in Sections 3.4.2.1 to 3.4.2.5. A general discussion of the *jDE* algorithm is given in Section 3.4.2.6.

3.4.2.1 Self-Adaptive Control Parameters

The base algorithm of *jDE* is the self-adaptive DE algorithm of Brest *et al.* [2006] discussed in Section 2.4.4. Each individual, $\vec{x}_i$, stores its own value for the crossover factor, Cr_i, and scale factor, $\mathcal{F}_i$. Before the DE mutation and crossover steps, new scale and crossover factors ($\mathcal{F}_{new_i}$ and Cr_{new_i}) are calculated as follows:

$$\mathcal{F}_{new_i} = \begin{cases} \mathcal{F}_l + U(0,1) \cdot \mathcal{F}_u \text{ if } (U(0,1) < \tau_1) \\ \mathcal{F}_i \text{ otherwise} \end{cases} \tag{3.4}$$

$$Cr_{new_i} = \begin{cases} U(0,1) \text{ if } (U(0,1) < \tau_2) \\ Cr_i \text{ otherwise} \end{cases} \tag{3.5}$$

where τ_1 and τ_2 are the probabilities that the factors will be adjusted. Brest *et al.* [2006] used 0.1 for both τ_1 and τ_2. F_l and F_u are parameters to the algorithm. *jDE* uses $F_l = 0.36$ based on recommendations made by Zaharie [2002a], and $F_u = 0.9$. During initialisation, the initial values for the scale and crossover factors associated with each individual are $F_{initial} = 0.5$ and $Cr_{initial} = 0.9$.

The purpose of the self-adaptive component is to find values for the scale and crossover factors that are appropriate to the fitness landscape. Subsequently, the scale and crossover factors are constantly adapted to find values appropriate for each point in time as changes in the environment occur.

3.4.2.2 Multiple Populations

Multiple populations are used to increase diversity and facilitate independent parallel exploration of the fitness landscape. Each sub-population explores a different region of the fitness landscape and consequently serves as a memory of local and global optima. A local optimum may become a global optimum after a change in the environment, making it beneficial to track multiple optima. Brest *et al.* [2009] used 5 sub-populations containing 10 individuals each.

3.4.2.3 Aging

An aging metaphor is employed at the individual level to prevent stagnation of sub-populations. Individuals that are determined to be too old can be reinitialised in order to allow the influx of fresh genetic material. A sub-population tracking a local optimum is probabilistically reinitialised when the best individual in the sub-population becomes too old. This makes it possible for the sub-population to converge to another (potentially global) optimum. A particular sub-population may thus discover several optima between changes in the environment.

Let $\vec{x}_{i,k}$ be the i^{th} individual in sub-population P_k. Let $\vec{x}_{best,k}$ refer to the best individual in sub-population P_k, and $\vec{\hat{y}}$ be the global best individual over all populations. Each individual $\vec{x}_{i,k}$ stores its own age, denoted by $A_{i,k}$. Individuals or sub-populations are reinitialised based on their age as described in Algorithm 8.

Algorithm 8: Aging and Reinitialisation

```
if x_{i,k} ≠ ŷ then
    if x_{i,k} = x_{best,k} and A_{i,k} > τ3 and U(0,1) < τ4 then
        Reinitialise sub-population P_k
    end
    else if A_{i,k} > τ5 and U(0,1) < τ6 then
        Reinitialise x_{i,k}
    end
end
```

The parameter τ_3 is the age that the best individual in a sub-population must reach

before the sub-population is reinitialised with probability τ_4. τ_5 is the age a normal individual must reach before the individual is reinitialised with probability τ_6. Parameters used by Brest *et al.* [2009] are $\tau_3 = 30$, $\tau_4 = 0.1$, $\tau_5 = 25$ and $\tau_6 = 0.1$. There is thus a 10% chance that individuals that are too old, or populations of which the best individual is too old, are reinitialised. The age value of each individual is incremented after each generation.

Offspring partially inherit their age from parents. When a new offspring individual is created, it is assigned an age based on its fitness and Euclidean distance from the individual it is to replace. Offspring individuals that are close, either in fitness or in Euclidean distance, to the individual they are to replace, are likely to be assigned a high age. Diversity is increased in the population by this process, since individuals with a high age will likely survive for a shorter period (and contribute less genetic material) than individuals with a low age. The inheritance procedure is shown in Algorithm 9, where $\vec{u}_i$ is the offspring individual that is to replace $\vec{x}_i$.

Algorithm 9: Age Inheritance

if $\|\vec{x}_i - \vec{u}_i\|_2 < \tau_7$ **or** $F(\vec{x}_i) - F(\vec{u}_i) < \tau_8$ **then**
  $A_i = \min\{A_i, \tau_9\}$
else
  $A_i = \min\{A_i, \tau_{10}\}$
end

The thresholds, τ_7 and τ_8, in Algorithm 9 determine how spatially close or similar in fitness an offspring individual must be to receive a high age. τ_9 is the age that an individual is assigned when it is too close to its parent individual, and τ_{10} is the age it receives when it is not. Parameters used by Brest *et al.* [2009] are $\tau_7 = 0.01$, $\tau_8 = 0.1$, $\tau_9 = 20$ and $\tau_{10} = 5$.

3.4.2.4 Exclusion

Exclusion is used to prevent sub-populations from converging to the same optimum. This increases diversity and assists parallel optima discovery. Exclusion is performed in the same way as in DynDE (see Algorithm 7), with the one difference being that a constant

value $\tau_{11} = 0.05$ is used as the exclusion threshold.

A further exclusion measure is used within sub-populations to increase diversity, namely local exclusion. Local exclusion prevents the occurrence of spatial differences of zero between individuals within each sub-population. The DE mutation step-sizes depend on the distance between difference vectors. Convergence to a single point is thus undesirable as evolution would effectively end and the algorithm would not be able to respond to changes in the environment.

Individuals that are located below a small Euclidean threshold distance from the best individual in the sub-population are reinitialised. Local exclusion is summarised in Algorithm 10, where the number of individuals within sub-population k is $n_{I,k}$. τ_{12} is the local exclusion threshold and a value of $\tau_{12} = 0.0001$ was used by Brest *et al.* [2009].

Algorithm 10: Local Exclusion

```
for k = 1, ..., n_k do
    for i = 1, ..., n_{I,k} do
        if ||x⃗_{best,k} − x⃗_{i,k}||_2 < τ_12 and x⃗_{best,k} ≠ x⃗_{i,k} then
            Reinitialise x⃗_{i,k}
        end
    end
end
```

3.4.2.5 Custom Reinitialisation

jDE employs an archive population, P_{ar}, of good solutions as a form of memory (in addition to using multiple populations). This archive starts out empty and grows by one individual every time a change in the environment is detected, at which time the best individual found before the change is added to the archive. The archive is utilised when individuals are reinitialised as described in Algorithm 11.

The archive is analogous to the *hall of fame* used in co-evolution. Through the archive, the current population can effectively draw on the experience of previous generations by searching in areas of the fitness landscape where optima were previously found. This is particularly useful in situations where the change type of the environment is cyclic which

results in the periodic return of the environment to the same state. The archive also provides useful information when the changes in the environment are small, which makes the locations of previous optima relevant for several generations.

An approach similar to the creation of Brownian individuals is used to perturb individuals drawn from the archive. Algorithm 11 makes use of random vectors $\vec{r}_1$ (with $r_{1,j} \sim N(0,1)$) and $\vec{r}_2$ (with $r_{2,j} \sim U(-0.5, 0.5)$).

Algorithm 11: Custom Reinitialisation of $\vec{x}_{i,k}(t)$

if $U(0,1) < \tau_{13}$ **and** $k = 1$ **and** $|P_{ar}| > 0$ **then**
  $a = U(1, |P_{ar}|)$;
  $\vec{z} = P_{ar}[a]$;
  $\varsigma = \tau_{14}/i$;
  if $(i \bmod 2) = 1$ **then**
    $\vec{x}_{i,k}(t+1) = \vec{z} + \varsigma \cdot \vec{r}_1$
  else
    $\vec{x}_{i,k}(t+1) = \vec{z} + \varsigma \cdot \vec{r}_2$
  end
else
  Generate $\vec{x}_{i,k}(t+1)$ randomly from a uniform distribution
end

Note that only the first sub-population makes use of the archive, to avoid genetic material from the archive dominating the entire population. τ_{13} is the probability that an individual will be generated using information from the archive. τ_{14} controls the magnitude of random perturbations. Parameters used by Brest *et al.* [2009] are $\tau_{13} = 0.5$ and $\tau_{14} = 0.1$.

3.4.2.6 *jDE* Discussion

jDE contains several novel components, for example, aging and local exclusion, that have never been used in previous algorithms. A significant disadvantage of *jDE*, however, is the number of control parameters used in the algorithm. In addition to parameters such as the number of sub-populations and sub-population size which are found in other

algorithms, *jDE* introduces parameters F_l and F_u to control the adaptation of the scale factor. Fourteen other general parameters, labelled τ_1 through τ_{14} are also introduced. Although Brest *et al.* [2009] provide guideline values for these parameters, and although the algorithm may not be equally sensitive to all parameters, the likelihood that parameters would have to be fine-tuned for specific problems is high. Appendix A shows that *jDE* is very sensitive to at least two of the parameters, τ_3 and τ_5. Given the large number of parameters, a manual fine-tuning process could be extremely time consuming.

3.5 Detecting Changes in the Environment

Most evolutionary dynamic optimisation algorithms respond to changes in the environment. When the period between changes is unknown, it is necessary for the algorithm to detect changes. Change detection is not trivial, since changes could be localised in small areas of the fitness landscape, or could involve the introduction of an optimum into an area of the search space where no individuals are located. It is thus necessary to have an appropriate strategy to detect changes. Standard change detection approaches include the periodic re-evaluation of the best individual in the population [Hu and Eberhart, 2002], stationary sentinels [Morrison, 2004], or random points in the problem space [Carlisle and Dozier, 2002]. Discrepancies between current and previous fitness values indicate a change in the environment [Nguyen *et al.*, 2012]. Richter [2009] did a thorough study on change detection in environments where it is difficult to detect changes.

The effective detection of changes is a complex problem worthy of a study on its own. The change detection approach used by an algorithm aimed at DOPs impacts the results of the algorithm, because if changes are not correctly detected, the algorithm cannot respond appropriately to changes in the environment. This dependence on the detection strategy could bias results.

The problem of change detection was consequently delineated as outside the scope of the current study. The algorithms that are evaluated in the following chapter use an automatic detection strategy whereby the algorithm can test for changes in the environment without using any function evaluations. The change period is not given as a parameter to the algorithm, but the algorithms are allowed to detect changes flawlessly by interrogat-

ing the benchmark function once every generation. One of the normal change detection strategies can thus be integrated into the algorithms when needed. Throughout this study, where comparisons are drawn with the published results of another algorithm in the literature, the change detection approach used by the other algorithm is used to obtain results for the comparison.

3.6 Conclusions

The purpose of this chapter was to introduce problems experienced by static optimisation algorithms applied to dynamic environments and to discuss progress made in the EA and SI research community to address these problems.

Section 3.2 motivated why diversity is a major cause of poor performance of DE on DOPs. It was shown that DE suffers from a lack of diversity in dynamic environments. Outdated information was highlighted as a secondary problem of DE in dynamic environments.

Several algorithms that were developed to solve DOPs were discussed in Section 3.3. Three broad categories for dynamic optimisation approaches were identified, namely: increasing diversity, employing memory, and parallel searching. Seven general strategies commonly used to achieve the three approaches where highlighted.

Two state-of-the-art differential evolution algorithms, namely DynDE and *jDE*, were discussed in detail in Section 3.4. DynDE is used in the next chapter as base algorithm for the approaches proposed in this thesis. *jDE* is used for the purpose of comparing the new approaches to a state-of-the-art DE-based algorithm. Both algorithms contain appropriate components which make them more suited to dynamic environments than DE, but it was also found that several new parameters were introduced. These parameters may require fine-tuning when DynDE and *jDE* are applied to real-world problems. The performance of DynDE and *jDE* on benchmark functions is investigated in the next chapter.

Section 3.5 contained a discussion on the automatic detection dynamic environment changes. The detection of changes are outside the scope of this study and an automatic detection strategy is used.

The next chapter discusses new adaptations to DynDE which are proposed by the

author. These adaptations are aimed at improved performance in dynamic environments. A comparative evaluation of the new algorithm is performed to determine whether performance is improved.

Chapter 4

NOVEL EXTENSIONS TO DYNDE

The purpose of this chapter is to propose and evaluate new adaptations to the DynDE algorithm. These adaptations include: an approach that allows sub-populations to compete for fitness evaluations based on performance and an approach aimed at improving the effectiveness of the exclusion component of DynDE. The two adaptations are combined to form a third approach. The new approaches are empirically compared to DynDE and analysed in terms of their performance characteristics on a large set of benchmark problems.

4.1 Introduction

The previous chapter described the DynDE algorithm as a multi-population, DE-based algorithm aimed at DOPs. The literature review found that DynDE employs strategies that are generally used by other algorithms to optimise dynamic environments (refer to Table 3.1). DynDE is a comparatively simple algorithm which has been shown to be effective in solving DOPs [Mendes and Mohais, 2005].

A potential problem with DynDE is the fact that function evaluations are equally distributed to all the sub-populations that are used to track optima. Function evaluations are wasted on optimising local optima, whereas merely locating the general location of

local optima would suffice for the purpose of tracking the optima. The algorithms of Kamosi *et al.* [2010b] and Novoa-Hernández *et al.* [2011], which were developed concurrently with the algorithms presented in this study, periodically remove badly performing sub-populations from the evolution process to allow more function evaluations for the better performing sub-populations. The competitive population evaluation (CPE) approach that is presented in this chapter proposes that sub-populations should compete for fitness evaluations. Lower error values can be found using fewer function evaluations, by evolving sub-populations sequentially based on performance, rather than in parallel.

DynDE makes use of exclusion to prevent sub-populations from converging to the same optima by reinitialising a sub-population that moves within the exclusion radius of another. The exclusion approach ignores the possibility that multiple optima may exist within the exclusion radius, which would result in only one of the optima being tracked. This problem is addressed by a second novel adaptation to DynDE, called reinitialisation midpoint check (RMC). RMC changes the mechanism by which exclusion is implemented in DynDE to avoid reinitialisation of sub-populations that are not positioned on the same optimum. The fitness of the midpoint between the best individuals in the sub-populations is checked to determine whether the sub-populations are located on the same optimum. Exclusion is not performed if multiple optima are detected.

This chapter is structured as follows: Section 4.2 describes how the DynDE exclusion threshold calculation can be changed so that information regarding the number of optima in the fitness landscape is not required. CPE is presented in Section 4.3. Section 4.4 describes the RMC approach. The combination of CPE and RMC to form competing differential evolution (CDE) is discussed in Section 4.5. Section 4.6 describes experiments to investigate the performance and scalability of DynDE, CPE, RMC and CDE. Specific research questions are formulated and the experimental procedure is given in Sections 4.6.1 and 4.6.2. Section 4.6.3 investigates appropriate settings for the sub-population size and number of Brownian individuals for DynDE. The scalability of DynDE, CPE, RMC and CDE as the change period, number of dimensions, severity of changes, change types, and underlying function are varied, is investigated in Section 4.6.4. Section 4.6.5 investigates whether the novel approaches proposed in this chapter significantly improve DynDE. The convergence behaviour of CDE is compared and contrasted with that of DynDE in Section

4.6.6. The performance of CDE is compared in Section 4.6.7 to that of DynDE, using the average lowest error before changes in the environment as performance measure.

A comparison of CDE to other algorithms in the literature is given in Section 4.7. The general applicability of the approaches presented in this chapter is investigated by incorporating the approaches into *jDE* in Section 4.8. Final conclusions are drawn in Section 4.9.

4.2 Alternative Exclusion Threshold Approach

This thesis considers dynamic environments where information regarding the number of optima is not available to the optimisation algorithm. Section 3.4.1.2 described the calculation of the exclusion threshold as used by Mendes and Mohais [2005] and Blackwell and Branke [2006]. The exclusion threshold, as calculated using equation (3.1), assumes that the number of optima is known. Knowledge of the number of optima is generally not available when solving a DOP, consequently, this thesis proposes that the exclusion threshold rather be calculated using the number of sub-populations as follows:

$$r_{excl} = \frac{V_{max,F} - V_{min,F}}{2{n_k}^{\frac{1}{n_d}}} \tag{4.1}$$

where n_k is the number of sub-populations, and $V_{max,F}$ and $V_{min,F}$ are the upper and lower search range of function F in the n_d dimensions (assuming equal ranges for all dimensions). The threshold is thus dependent on the number of available sub-populations. A small number of sub-populations results in a large exclusion threshold, thus assigning a large area of the search space to each sub-population. Conversely, a large number of sub-populations results in a small exclusion threshold, consequently assigning a relatively small area of the search space to each sub-population. Equation (4.1) was also used by Blackwell [2007] in the self-adapting multi-swarms algorithm discussed in Section 3.3.3.2.

Note that, for the moving peaks benchmark (MPB) experiments discussed in this chapter, the same number of populations as the number of optima was used. The results can thus be directly compared to results from researchers that made use of equation (3.1) to calculate the exclusion threshold, since the same threshold value was used.

4.3 Competitive Population Evaluation

The effectiveness of an algorithm on static optimisation problems is usually measured by the error of the best solution found at the end of the optimisation process. Although many approaches aim to reduce the execution time of optimisation algorithms (i.e. reach a low error value as soon as possible), the error during the course of the optimisation process is of secondary concern. In contrast, optimisation in dynamic environments implies that a solution is likely to be required at all times (or at least just before changes in the environment occur), not just at the termination of the algorithm. In these situations, it is imperative to find the lowest error value as soon as possible after changes in the environment have occurred.

The offline error performance measure (discussed in Section 2.5.5) measures an algorithm's efficiency at finding solutions quickly by calculating the performance of an optimisation algorithm as the average of the lowest error value over every function evaluation, as opposed to averaging errors immediately prior to changes in the environment. The offline error can thus be reduced by finding solutions earlier (without necessarily finding a better solution). An algorithm that finds solutions faster will also be beneficial in situations where evaluation criteria are only concerned with the error value immediately prior to changes in the environments (as is the case with performance measure number two in Section 2.5.5), since the algorithm will be more effective than a standard algorithm, when the number of function evaluations between changes is reduced. A dynamic optimisation algorithm can thus be improved, not only by reducing the error, but also by making the algorithm reach its lowest error value (before a change occurs in the environment) in fewer function evaluations.

The above argument is the motivation for a new approach to finding solutions earlier which is proposed in this thesis, namely competitive population evaluation (CPE) [du Plessis and Engelbrecht, 2012b]. CPE is an extension to DynDE discussed in Section 3.4.1. The primary goal of the new approach is not to decrease the error value found by DynDE, but rather to make the algorithm reach the lowest error value in fewer function evaluations. The proposed algorithm aims to achieve this by initially allocating all function evaluations, after a change in the environment, to the sub-population that has the current

best solution. Thereafter function evaluations are allocated to other sub-populations.

The mechanism used by CPE to allocate function evaluations is based on the performance of sub-populations. The best-performing sub-population is evolved on its own until its performance drops below that of another sub-population. The new best-performing sub-population then evolves on its own until its performance drops below that of another sub-population. This process is repeated until a change in the environment occurs. Ideally, CPE would locate the global optimum early, while locating local optima later. CPE thus differs from DynDE in that peaks are not located in parallel, but sequentially. The CPE process is detailed in Algorithm 12. Changes in the environment are always followed by the evolution of all sub-populations for two generations. These two generations are necessary in order to calculate the performance value, $\mathcal{P}_k$, for each of the P_k sub-populations. However, after the initial two generations, the CPE process evolves only one sub-population per generation. CPE is thus responds directly to changes in the environment, unlike algorithms like CPSOR (refer to Section 3.3.3.2).

Algorithm 12: Competitive population evaluation

while *termination criterion not met* **do**

  Allow the standard DynDE algorithm to run for two generations;

  repeat

    for $k = 1, \ldots, n_k$ **do**

      Calculate the performance value, $\mathcal{P}_k(t)$

    end

    Select sub-population P_a such that $\mathcal{P}_a(t) = \min_{k=1,\ldots,n_k} \{\mathcal{P}_k(t)\}$;

    Evolve only sub-population P_a using DE for one generation;

    $t = t + 1$;

    Perform exclusion according to Algorithm 7;

    Create Brownian individuals;

  until *a change in the environment occurs*;

end

The performance value, $\mathcal{P}_k$, of sub-population P_k depends on two factors: The current

fitness of the best individual in the sub-population and the error reduction of the best individual during the last evaluation of the sub-population. Let n_k be the number of sub-populations, $\vec{x}_{best,k}$ the best individual in sub-population P_k, and $F(\vec{x}_{best,k}, t)$ the fitness of the best individual in sub-population P_k during iteration t. The performance $\mathcal{P}_k(t)$ of population P_k, after iteration t, is given by:

$$\mathcal{P}_k(t) = (\Delta F(\vec{x}_{best,k}, t) + 1)(R_k(t) + 1) \tag{4.2}$$

where

$$\Delta F(\vec{x}_{best,k}, t) = |F(\vec{x}_{best,k}, t) - F(\vec{x}_{best,k}, t-1)| \tag{4.3}$$

For function maximisation problems, $R_k(t)$ is calculated as:

$$R_k(t) = |F(\vec{x}_{best,k}, t) - \min_{a=1,\ldots,n_k} \{F(\vec{x}_{best,a}, t)\}| \tag{4.4}$$

and for function minimisation problems,

$$R_k(t) = |F(\vec{x}_{best,k}, t) - \max_{a=1,\ldots,n_k} \{F(\vec{x}_{best,a}, t)\}| \tag{4.5}$$

The best performing sub-population is, therefore, the sub-population with the highest product of fitness and improvement. The motivation for the addition of 1 to the first and second terms in equation (4.2) is to prevent a performance value of zero being assigned to a sub-population. Without the addition in the second term, the least fit population will always be assigned a performance value of zero (since the product of the first and second term will be zero) and will never be considered for searching for an optimum. Similarly, without the addition in the first term, a good performing population that does not show any improvement during a specific iteration, is assigned a performance value of zero and will never be considered for evaluation again. The addition of 1 to the first and second terms is thus included to ensure that every sub-population could, potentially, have the highest performance value and subsequently be given function evaluations.

The absolute values of $\Delta F(\vec{x}_{best,k}, t)$ and $R_k(t)$ are taken to ensure that the performance values are always positive. When a population is reinitialised due to exclusion (see Section 3.4.1.2), the fitness of the best individual is likely to be worse than before reinitialisation (since the sub-population now consists of randomly generated individuals).

This results in a large $\Delta F(\vec{x}_{best,k}, t)$ value. The sub-population is consequently assigned a relatively large performance value, making it likely that the sub-population would be allocated fitness evaluations in the near future.

The CPE approach can be clarified by an example. Figure 4.1 plots the error per function evaluation of the best individual in each of three sub-populations (labelled Population 1, Population 2 and Population 3), found during an actual run of the DynDE algorithm on the MPB using Scenario 2 settings. During the period that is depicted, no changes occurred in the environment. Note that one of the sub-populations (Population 3) converged to a global optimum, as is evidenced by its error value approaching zero, while the others likely converged to local optima.

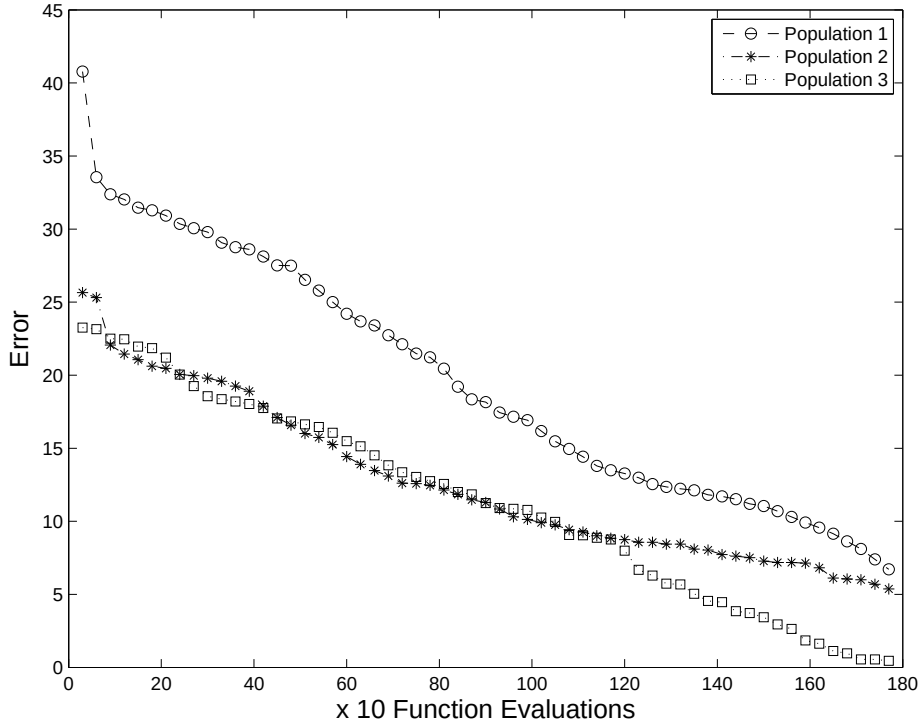

Figure 4.1: Error profile of three DynDE populations in a static environment

The performance of each of the populations was calculated using equation (4.2) on page 111. A plot of the performance of each of the populations is given in Figure 4.2. Population 2 and Population 3 alternated between having the highest performance value, for approximately the first half of the period depicted. After the first half of the depicted

period, Population 3 consistently received the highest performance value, as it converged to a global optimum. Population 3 received, on average, a higher performance value than the other two populations. Population 1, which had the highest overall error, received a relatively low average performance value. Note that it was only after about 1 000 function evaluations that Population 2 and Population 3 received a lower performance value than the initial performance value of Population 1.

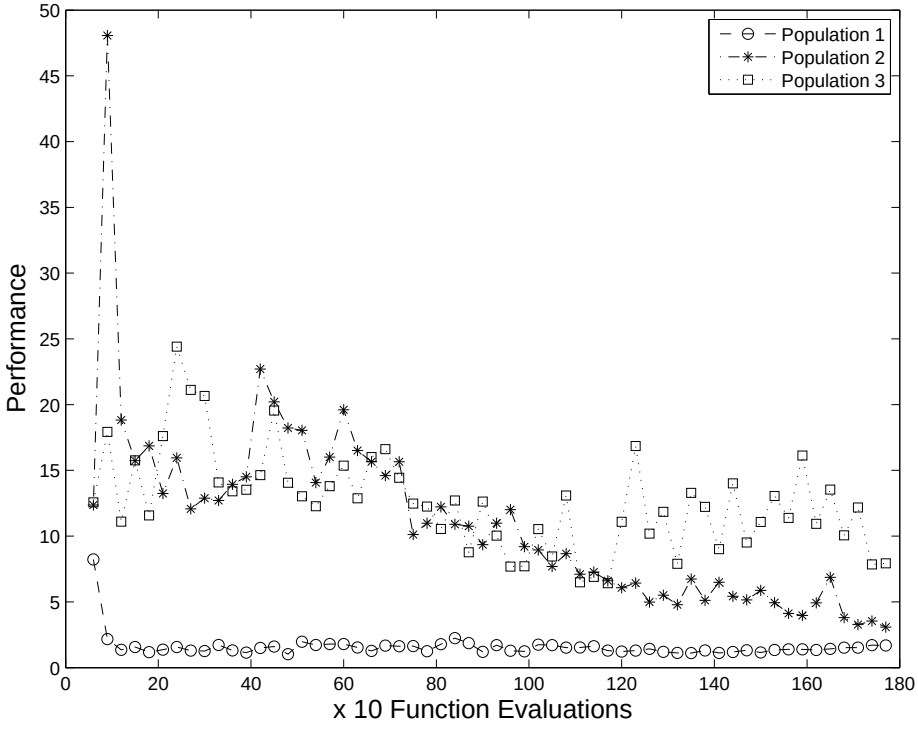

Figure 4.2: Performance, $\mathcal{P}$, of each population

The behaviour of each of the populations, when using the CPE process (refer to Algorithm 12) to selectively evolve sub-populations on their own based on their respective performance values, can now be observed by using the calculated performance values. The plots of the error of each of the populations per function evaluation when using CPE are given in Figure 4.3. Population 2 and Population 3 were alternately evolved for the first few iterations, followed by a period where only Population 2 was evolved. After about 250 function evaluations, Population 3 was evolved (except for a few intermittent iterations around 700 function evaluations) until it converged to the global optimum at about 1 000

function evaluations. Population 2 was evolved during the next period which lasted until about 1 200 function evaluations, while Population 1 was evolved during the last period.

Note that the lowest error was reached after 1 000 function evaluations in Figure 4.3, while this point was only reached after 1 800 function evaluations in Figure 4.1. The performance values of the three populations when using CPE are given in Figure 4.4. Observe that for considerable periods, the performance values of some of the sub-populations remain constant. This occurs because only one sub-population is evolved at a time, during which the performance value of the other populations remains unchanged.

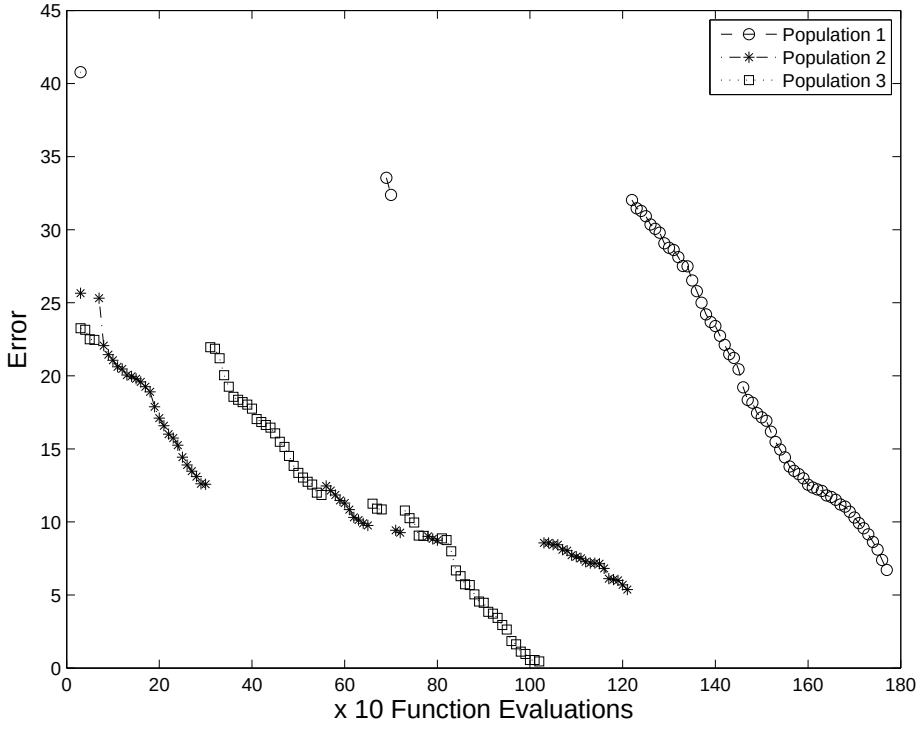

Figure 4.3: Error profile of three populations when using competitive evaluation

More successful sub-populations are clearly allocated more function evaluations earlier in the evolution process. Optima are located sequentially by CPE and in parallel by DynDE. Figure 4.5 depicts the effect that using CPE had on the offline error for the three sub-populations example. The curve of the offline error, when competing sub-populations are used, exhibits a steeper downward gradient than the curve of normal DynDE, due to earlier discovery of the global optimum. Overall, the offline error was reduced by more

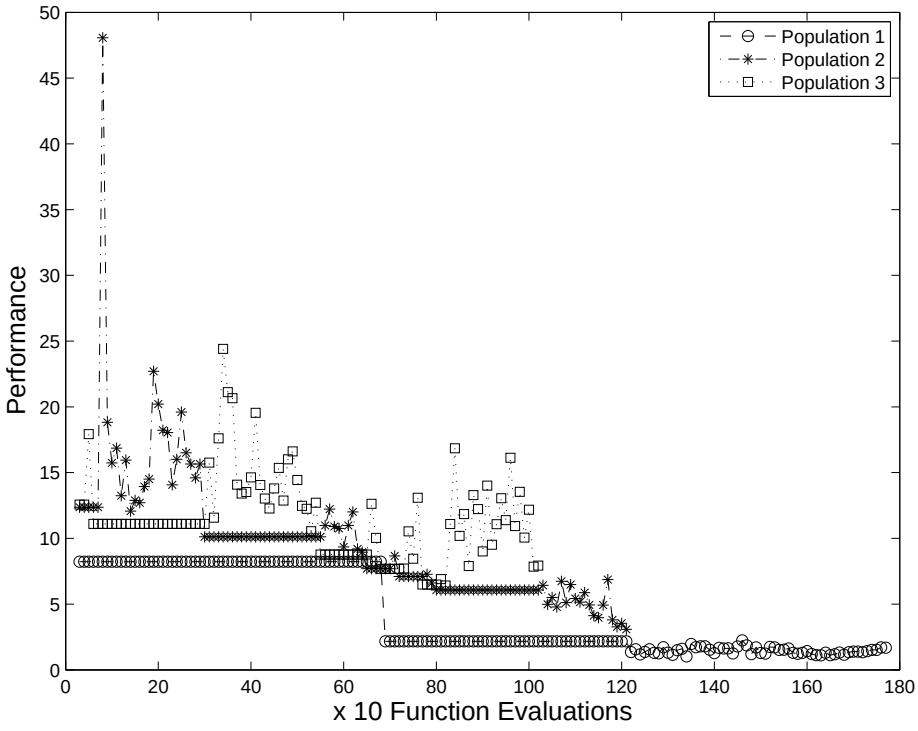

Figure 4.4: Performance, $\mathcal{P}$, of each population when using competitive evaluation

than 30% after 1 800 function evaluations.

The lowest error value is reached sooner by competitively choosing the better performing populations to evolve before other populations, thus reducing the average error. This technique has the added advantage that better performing populations will receive more function evaluations that would otherwise have been wasted on finding the maximum of the sub-optimal peaks. The overall error value should consequently also be reduced.

An advantage of CPE is that it only utilises information that is available in normal DynDE, so that no extra function evaluations are required. A potential disadvantage of the CPE process is that too few function evaluations may be allocated to the weaker sub-populations. Consequently, the weaker sub-populations may not locate local optima which may become global optima after changes in the environment. This is more likely to be a problem in situations where changes in the environment are infrequent, since normal DynDE would have enough function evaluations available to optimise all optima thoroughly. The benefits of locating optima early are also expected to diminish as the

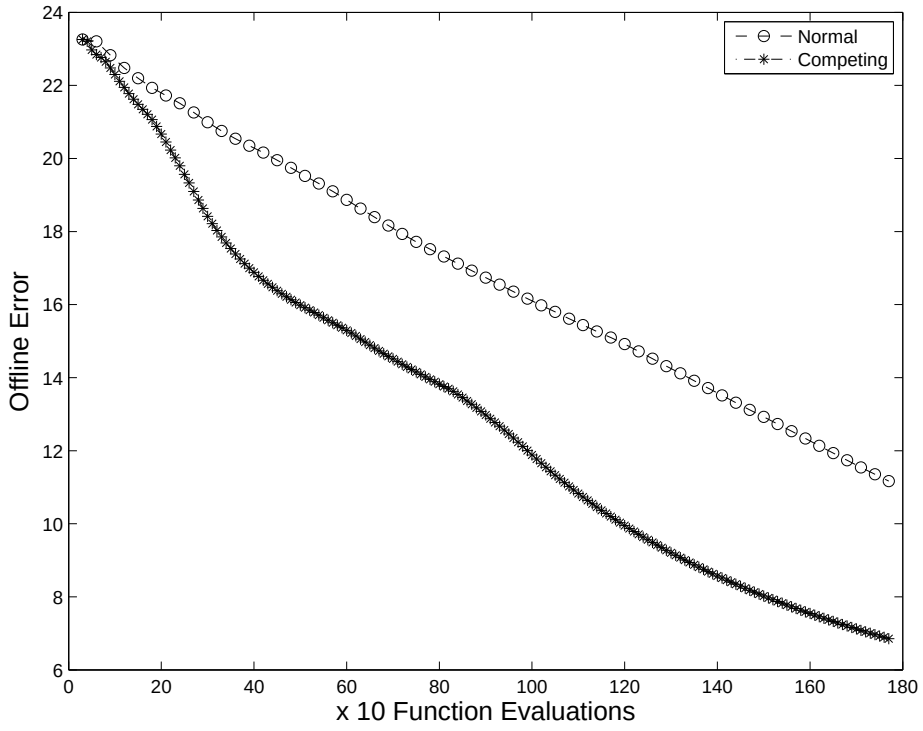

Figure 4.5: Comparison of offline error for normal and competitive evaluation

period between changes in the environment increases, because the offline error would be averaged over a larger number of function evaluations, thus reducing the impact of initial lower errors. CPE is thus expected to be most applicable to rapidly changing dynamic environments.

4.4 Reinitialisation Midpoint Check

The exclusion approach, which was described in Section 3.4.1.2, causes DynDE to reinitialise the weaker sub-population when two sub-populations are located within the exclusion radius of each other. This approach does not take into account the case when two optima are located extremely close to each other, i.e. within the exclusion threshold of one another. One of the sub-populations will be reinitialised in these situations, leaving one of the optima unpopulated. This section proposes that this problem be partially remedied by determining whether the midpoint between the best individuals in each sub-population

constitutes a higher error value than the best individuals of both sub-populations. If this is the case, it implies that a trough exists between the two sub-populations and that neither should be reinitialised (refer to Figure 4.6, scenario A). This approach is referred to in this thesis as the reinitialisation midpoint check (RMC) approach. Algorithm 13 details how exclusion will be performed when RMC is used (assuming function minimisation).

Algorithm 13: RMC Exclusion

```
for k_1 = 1, ..., n_k do
    for k_2 = 1, ..., n_k do
        if ||x_best,k1 − x_best,k2||_2 < r_excl and k_1 ≠ k_2 then
            Let x_mid = (x_best,k1 + x_best,k2)/2;
            if F(x_mid) > F(x_best,k1) and F(x_mid) > F(x_best,k2) then
                if F(x_best,k1) < F(x_best,k2) then
                    Reinitialise population P_k2
                else
                    Reinitialise population P_k1
                end
            end
        end
    end
end
```

It is apparent that RMC does not work in all cases. Scenarios B and C of Figure 4.6 depict situations where multiple optima within the exclusion threshold are not detected by a midpoint check. Scenario C further constitutes an example where no point between the two optima will give a higher error, thus making it impossible to detect two optima by using any number of intermediate point checks.

This approach is similar to, but simpler, than *hill-valley detection* suggested by Ursem [2000] (described in Section 3.3), since only one point is checked between sub-populations. The midpoint check approach provides a method of detecting multiple optima within the exclusion threshold without being computationally expensive or using too many function evaluations, since only one point is evaluated.

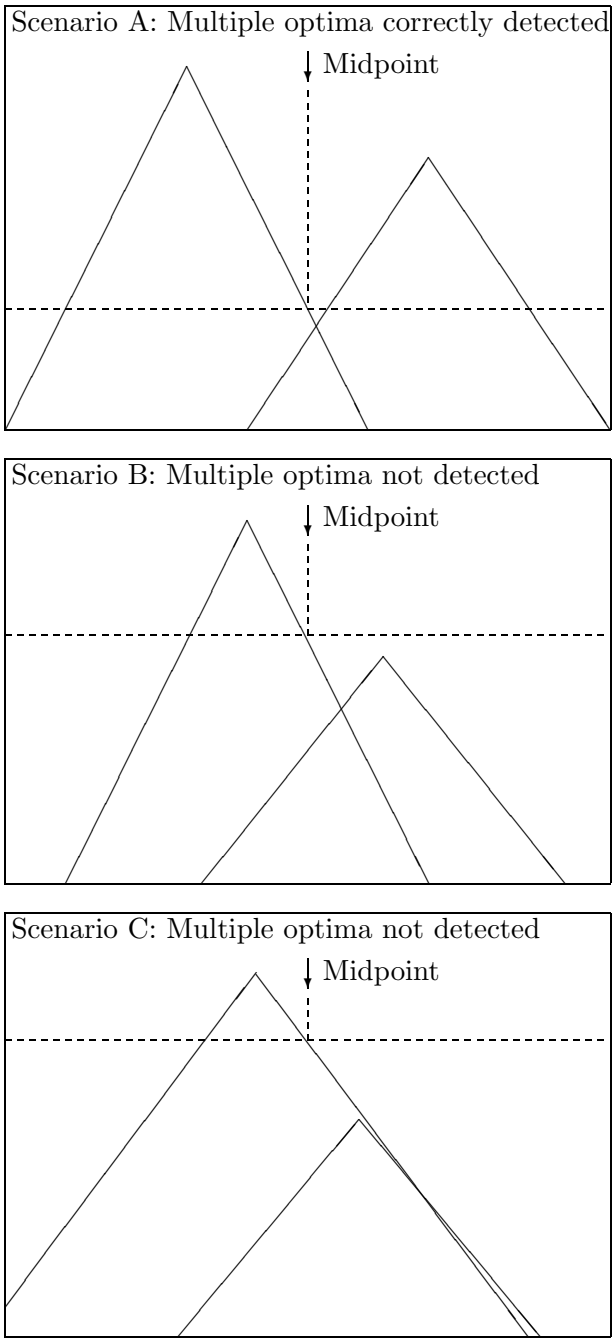

Figure 4.6: Midpoint checking scenarios

4.5 Competing Differential Evolution

The RMC and CPE algorithms are mutually independent and can thus be combined into a single algorithm. This algorithm is referred to as competing differential evolution (CDE). CDE thus consists of the standard DynDE algorithm, with the addition of a midpoint check before reinitialising populations when they are within the exclusion threshold of each other, and the evaluation of populations based on their performance. The pseudo-code for CDE is identical to that of CPE given in Algorithm 12, but with the exclusion step using Algorithm 13.

4.6 Experimental Results

This section describes the empirical analysis of DynDE, CPE, RMC and CDE and discusses the results. Experiments are designed to answer specific research questions given in Section 4.6.1. The general experimental procedure is given in Section 4.6.2. The analysis pertaining to each research question is given in Sections 4.6.3 to 4.6.7 respectively.

4.6.1 Research Questions

The following research questions were identified as pertinent for analysing the algorithms proposed in this chapter:

1. *What are effective settings for sub-population size and number of Brownian individuals?* Mendes and Mohais [2005] recommended using a small sub-population size of six individuals. This can be seen as counterintuitive since larger sub-population sizes should increase diversity. The effect of larger sub-populations is investigated. Various settings for the number of Brownian individuals are also investigated to determine the most effective value.

2. *How do DynDE, RMC, CPE and CDE scale under factors that influence the complexity of a DOP?* Section 2.5.2 described the factors that determine an optimisation algorithm's ability to locate optima in a dynamic environment. The factors that are investigated are: the number of dimensions of the search space, the number of function evaluations between changes in the environment (change period), the severity of

changes, change types, and the function describing the fitness landscape. The scalability study analyses the performance of the algorithms when varying combinations of the previously mentioned factors.

3. *Are RMC, CPE and CDE more effective dynamic optimisation algorithms than DynDE?* The results found when measuring the performance of the algorithms suggested in this chapter on benchmark problems are compared to the results of DynDE.

4. *How does the convergence behaviour of CDE differ from that of DynDE?* The convergence behaviours of DynDE and CDE are compared in terms of diversity, current error and the resulting offline error to assist in explaining the trends observed in the analyses of research questions 1 and 2.

5. *How does the average lowest error, found just before changes in the environment, differ between DynDE and CDE?* CPE aims to locate optima earlier, but is not specifically designed to achieve a lower overall error value before changes occur in the environment. Similarly, RMC is not directly focused on lowering the error, but rather on improving the optima tracking process. This research question is investigated to determine how the performance of the new CDE algorithm differs from that of CDE in terms of the average lowest error values before changes in the environment.

The experimental procedure followed to investigate the research questions is outlined in Section 4.6.2.

4.6.2 Experimental Procedure

The performance of DynDE, CPE, RMC and CDE was evaluated on the MPB and the GDBG. The simplicity of the MPB makes it ideal for investigating the effect of different change severities, while the GDBG provides functionality to investigate different change types and underlying functions. Accordingly, a *standard set* of experiments is defined here. Variations of the Scenario 2 settings on the MPB which are included in the standard set are given in Table 4.1. The settings include several values for change severity and

both the cone and sphere peak functions. The combinations of change severity and peak function results in 12 MPB variations which are included in the standard set.

Table 4.1: MPB Scenario 2 Variations

Setting	**Value**
Number of dimensions (n_d)	5
Number of Peaks	10
Max and Min Peak height	[30,70]
Max and Min Peak width	[1.0,12.0]
Change period (Cp)	5000
Change severity (C_s)	1.0, 5.0, 10.0, 20.0, 40.0, 80.0
Height severity	7.0
Width severity	1.0
Function (F)	Cone, Sphere
Correlation	0.0

The GDBG problems considered here consist of six change types (T_1 to T_6) and six functions (F_1 to F_6). Refer to Section 2.5.4 for details on these functions and change types. The creators of the GDBG suggested that two instances of F_1 be used: one containing 10 peaks and one containing 50 peaks. These instances are denoted by F_{1a} and F_{1b} respectively. The combinations of change types and functions thus yield a total of 42 GDBG environments that are included in the standard set. The GDBG variations are summarised in Table 4.2. The default values for the number of dimensions and change period are in accordance with the settings for the IEEE WCCI-2012 competition on evolutionary computation for dynamic optimisation problems [Li *et al.*, 2011].

Table 4.2: GDBG Variations

Setting	**Value**
Number of dimensions (n_d)	5
Change period (Cp)	50000
Function (F)	F_{1a}, F_{1b}, F_2, F_3, F_4, F_5, F_6
Change type (Ct)	T_1, T_2, T_3, T_4, T_5, T_6

The 54 environments in the standard set thus contain 12 environments from the MPB and 42 environments from the GDBG. The standard set is varied to investigate the effect of different numbers of dimensions and various values for the change period (number of function evaluations between changes in the environment). The variations on the standard are given in Table 4.3.

Table 4.3: Standard Set Variations

Setting	**Values**
Number of dimensions (n_d)	5, 10, 25, 50, 100
Change period (Cp)	100, 500, 1000, 5000, 10000, 25000, 50000, 100000

A stopping criterion of 60 changes in the environment was used for all experiments as suggested in [Li *et al.*, 2008], [Li *et al.*, 2011]. The offline error was used as performance measure for all environments and experiments were repeated 30 times to facilitate drawing statistically valid conclusions from the results. Mann-Whitney U tests were used to test statistical significance when comparing algorithms. Unless otherwise stated, the algorithms used the parameter settings described in Chapter 3.

4.6.3 Research Question 1

What are effective settings for sub-population size and number of Brownian individuals?

Mendes and Mohais [2005] investigated sub-population sizes of 6 and 10 with 2 and 5 Brownian individuals respectively. Experimental results on the MPB showed that the smaller sub-population size was more effective. The experiments conducted by Mendes and Mohais [2005] can be criticized for not being extensive enough for three reasons. Firstly, only two sub-population size values (which are reasonably close to each other) were tested. Secondly, the ratio of Brownian versus normal individuals is not the same in the two tests: the sub-population of size 6 used $33.\dot{3}\%$ of each sub-population as Brownian, while the sub-population of size 10 used 50% of each sub-population as Brownian. The results may thus have been determined by the percentage of Brownian individuals rather than by the sub-population size. Thirdly, the experiments used only the MPB without investigating any variations of the Scenario 2 settings.

This section presents the results of a more extensive investigation to determine the most effective sub-population size and the appropriate proportion of Brownian individuals. The goal of the experimental investigation is to find settings that works well over a broad range of dynamic environments. Four settings for sub-population size were tested: 6, 12, 24, and 48. Six was selected as the smallest sub-population size tested, to prevent the DynDE's DE/best/2/bin scheme from always using the same five individuals to form the mutant vector. Seven values for the percentage of Brownian individuals in each sub-population were investigated: 0%, 16.$\dot{6}$%, 33.$\dot{3}$%, 50%, 66.$\dot{6}$%, 83.$\dot{3}$%, and 100%. These percentages correspond to a sub-population of size six, using zero, one, two, three to six Brownian individuals respectively. The other sub-population sizes that were tested are all multiples of six, so an integer number of Brownian individuals was used in each case.

The four sub-population sizes and the seven percentages of Brownian individuals resulted in 28 combinations of settings being investigated. The standard set of environments (described in Section 4.6.2) was used to test each of the settings. The standard set was also varied for each of the change periods and numbers of dimensions listed in Table 4.3. A total of 648 experiments was thus conducted for each of the 28 settings.

The results for each setting were compared with those of each of the other settings to determine which resulted in the lowest offline error. A Mann-Whitney U test was used in each case to determine whether differences in average offline error were statistically significant at a 95% confidence level. Table 4.4 gives a count for each setting of the number of experiments that yielded a statistically significantly lower offline error than any of the other settings. A maximum count of 17 496 can be found for each setting as 28 settings were each investigated on 648 experiments. Table 4.5 lists a similar count as Table 4.4, but here the counter was only incremented if a particular setting performed better than **all** other settings for a particular experiment. A maximum value of 648 can thus be achieved.

The results presented in Tables 4.4 and 4.5 indicate that using 16.$\dot{6}$% Brownian individuals yields the most effective algorithm. The two instances that scored the highest values in both tables occurred in rows that used 16.$\dot{6}$% Brownian individuals. The conclusion can thus be drawn that using 16.$\dot{6}$% Brownian individuals gives the best performance over a wide range of environments.

Table 4.4: Number of environments in which each setting resulted in better performance than other settings

	Sub-Population Size			
Percentage Brownian	**6**	**12**	**24**	**48**
0%	1585	5172	5396	4251
16.6̇%	14248	14613	12425	8854
33.3̇%	12891	13038	11076	7703
50%	11432	10509	8558	5774
66.6̇%	9656	8440	6432	4275
83.3̇%	7376	6173	4554	2892
100%	6159	5036	3438	2001

Table 4.5: Number of environments in which each setting performed the best

	Sub-Population Size				
Percentage Brownian	**6**	**12**	**24**	**48**	**Total Best**
0%	1	17	14	1	33
16.6̇%	251	56	6	0	313
33.3̇%	1	0	1	0	2
50%	0	0	0	0	0
66.6̇%	0	0	0	0	0
83.3̇%	0	0	0	0	0
100%	15	5	3	0	23
Total Best	268	78	24	1	

The results for the most effective number of sub-populations is less clear-cut. The highest value in Table 4.4 is for a sub-population size of 12 (14 613 out of a possible 17 496), while the highest value in Table 4.5 is for a sub-population size of 6 (251 out of a possible 648). The sub-population size of 12 outperformed more of the other settings per experiment, however the sub-population size of 6 most often performed the best per experiment. Further analysis is thus required to determine whether a sub-population size of 6 or 12 is the most effective.

Table 4.6 gives results comparing the comparative performance of the four sub-population

sizes. The percentage Brownian individuals are kept constant at $16.\dot{6}\%$ for this analysis. Each row of the table gives the number of times (out of a possible 648) that DynDE, using the indicated number of sub-populations, performed better than the sub-population size, shown in the respective columns. Totals are provided for the number of times that each setting performed better (given in the last column) and worse (given in the last row) than other settings.

Table 4.6: Number of environments in which each sub-population size resulted in better performance than other sub-population sizes (row vs column) when using $16.\dot{6}\%$ Brownian individuals

	Sub-Population Size				
Sub-Population Size	**6**	**12**	**24**	**48**	**Total Better**
6	-	340	424	487	1251
12	199	-	476	547	1222
24	150	61	-	529	740
48	98	53	20	-	171
Total Worse	447	454	920	1563	

The sub-population size of six experiments yielded the highest total number of better values (1 251 out of a possible 1 944) and the smallest total number of worse values (447). DynDE, with sub-population size of six, was better than its closest contender (population size of 12) in 340 cases and worse in only 199 cases. These results provide evidence that using a sub-population size of six is the most effective approach. This conclusion is strengthened by summing the number of times that each of the sub-population sizes performed better than all the other three sub-population sizes for each experimental setting. This information is summarised in Table 4.7. The experiments with a sub-population size of six performed the best, considerably more often, than the other sub-population sizes. It can thus be concluded that using a sub-population size of six yields the best results over a broad range of dynamic environments.

A final analysis is given here to confirm the appropriateness of using $16.\dot{6}\%$ Brownian individuals in a sub-population of 6 individuals. Table 4.8 lists the number of experiments in which each of the percentages of Brownian individuals performed the best when a sub-

Table 4.7: Number of environments in which each sub-population size resulted in the best performance when using 16.6̇% Brownian individuals

Sub-Population Size	**6**	**12**	**24**	**48**
Number Best	340	81	23	20

population size of 6 was used. The experiments that used 16.6̇% Brownian individuals outperformed the experiments using other percentages by a wide margin.

Table 4.8: Number of environments in which each percentage of Brownian individuals resulted in the best performance when using a sub-population size of 6

Percentage Brownian	**0%**	**16.6̇%**	**33.3̇%**	**50%**	**66.6̇%**	**83.3̇%**	**100%**
Number Best	18	351	1	0	0	0	53

The 18 cases where using no Brownian individuals yielded better offline errors than the other percentage settings were all in environments with a change period of either 100 or 500. These rapidly changing environments allow DynDE to perform very few generations (about one generation with a change period of 100 and five generations with a change period of 500). DynDE is unable to converge to optima effectively with so few generations, which means that the Brownian individuals are created around inferior individuals and function evaluations are consequently wasted. However, the disadvantage of using Brownian individuals in these rapidly changing environments is minor, because the results were statistically significantly better in only 18 of the 84 rapidly changing environments that were investigated when Brownian individuals were not used.

This research question investigated appropriate settings for the sub-population size and the proportion of Brownian individuals. The results presented in this section indicate that Mendes and Mohais [2005] were correct in recommending a small sub-population size of six, but that, using a single Brownian individual, rather than two, yields the best results over a wide selection of dynamic environments.

4.6.4 Research Question 2

How do DynDE, RMC, CPE and CDE scale under factors that influence the complexity of a DOP?

This section presents the results of a scalability study on DynDE, CPE, RMC and CDE with respect to dimension, change period, change type, change severity, and underlying function. The standard set (refer to Section 4.6.2) of environments was varied over all combinations of change periods and dimensions given in Table 4.3. The standard set contains 54 environments which, multiplied by the five dimensional settings and the eight change period settings, gives a total of 2 160 experiments that were conducted per algorithm.

The rest of this section is structured as follows: Section 4.6.4.1 gives the results of each algorithm on the standard set. Section 4.6.4.2 describes the scalability of the algorithms with respect to change period, Section 4.6.4.3 the scalability with respect to dimension, Section 4.6.4.4 the scalability with respect to change severity, Section 4.6.4.5 the scalability with respect to underlying function, and Section 4.6.4.6 the scalability with respect to change type. The observed trends are summarised in Section 4.6.4.7.

4.6.4.1 Performance on the Standard Set

The large amount of experimental data (a total of 8 640 experiments) makes it inconvenient to present all results in tabular form. Accordingly, the following sections employ graphs to illustrate the observed trends. The offline errors of each of the algorithms on the standard set is given in Table 4.9 to provide context to the graphs given in the following sections. The MPB conical peak function is denoted by "C", the spherical peak function by "S", the change severity by "C_s", the GDBG functions by "F_1" to "F_6", and the GDBG change types by "T_1" to "T_6".

The first observation that can be made from Table 4.9 is that the different settings of the standard set clearly has a large impact on the offine errors produced by each of the algorithms. For example, observe the increase in offline error of the algorithms on the MPB with the conical peak function (denoted by C) as the change severity, C_s, is increased. Also note that the increase in offline error with respect to change severity is

Table 4.9: Offline error of DynDE, CPE, RMC and CDE on the standard set

Settings		DynDE	CPE	p-val	RMC	p-val	CDE	p-val
C	C_s 1.0	1.05 ± 0.12	**0.95 ± 0.15**	0.155	**0.85 ± 0.06**	*0.021*	**0.80 ± 0.21**	*0.000*
	5.0	4.26 ± 0.22	**2.51 ± 0.25**	*0.000*	**4.22 ± 0.25**	0.741	**2.79 ± 0.22**	*0.000*
	10.0	10.87 ± 0.67	**5.53 ± 0.45**	*0.000*	**10.39 ± 0.56**	0.343	**5.60 ± 0.39**	*0.000*
	20.0	20.33 ± 1.08	**11.50 ± 0.71**	*0.000*	**19.04 ± 1.26**	0.075	**11.98 ± 0.79**	*0.000*
	40.0	20.49 ± 1.01	**20.11 ± 1.18**	0.458	20.59 ± 1.46	0.602	**19.47 ± 0.78**	0.109
	80.0	28.53 ± 1.82	**24.36 ± 1.14**	*0.000*	28.54 ± 1.60	0.752	**24.10 ± 0.81**	*0.000*
S	C_s 1.0	1.06 ± 0.17	1.15 ± 0.15	0.300	**0.80 ± 0.10**	*0.017*	1.07 ± 0.21	0.797
	5.0	4.27 ± 0.13	**2.88 ± 0.15**	*0.000*	**4.03 ± 0.08**	*0.009*	**2.58 ± 0.10**	*0.000*
	10.0	18.16 ± 0.23	**10.42 ± 0.21**	*0.000*	**17.79 ± 0.24**	*0.040*	**9.71 ± 0.22**	*0.000*
	20.0	73.74 ± 1.37	**46.01 ± 1.00**	*0.000*	**72.90 ± 2.05**	0.242	**43.71 ± 0.89**	*0.000*
	40.0	166.96 ± 6.38	**152.81 ± 3.75**	*0.000*	**162.76 ± 6.49**	0.350	**147.32 ± 3.87**	*0.000*
	80.0	242.02 ± 10.48	**234.25 ± 6.86**	0.366	**240.73 ± 7.69**	0.947	**220.62 ± 6.11**	*0.005*
F_{1a}	T_1	9.73 ± 0.20	**6.03 ± 0.29**	*0.000*	9.74 ± 0.20	0.982	**5.99 ± 0.22**	*0.000*
	T_2	18.75 ± 0.45	**10.87 ± 0.30**	*0.000*	**18.47 ± 0.37**	0.248	**10.36 ± 0.29**	*0.000*
	T_3	17.35 ± 1.19	**9.84 ± 0.75**	*0.000*	**16.35 ± 0.95**	0.286	**10.56 ± 0.69**	*0.000*
	T_4	20.24 ± 0.33	**13.54 ± 0.32**	*0.000*	**20.13 ± 0.32**	0.513	**13.08 ± 0.27**	*0.000*
	T_5	22.45 ± 0.35	**13.45 ± 0.42**	*0.000*	22.50 ± 0.32	0.832	**13.63 ± 0.48**	*0.000*
	T_6	26.52 ± 0.45	**17.18 ± 0.38**	*0.000*	**26.43 ± 0.37**	0.752	**17.24 ± 0.36**	*0.000*
F_{1b}	T_1	11.56 ± 0.44	**8.89 ± 0.35**	*0.000*	**11.28 ± 0.37**	0.335	**9.35 ± 0.41**	*0.000*
	T_2	16.91 ± 0.36	**11.35 ± 0.32**	*0.000*	**16.63 ± 0.34**	0.358	**11.35 ± 0.28**	*0.000*
	T_3	16.80 ± 0.57	**11.14 ± 0.51**	*0.000*	**16.64 ± 0.61**	0.741	**12.26 ± 0.63**	*0.000*
	T_4	21.64 ± 0.51	**15.34 ± 0.44**	*0.000*	21.91 ± 0.50	0.314	**15.42 ± 0.57**	*0.000*
	T_5	18.84 ± 0.21	**11.93 ± 0.21**	*0.000*	**18.49 ± 0.25**	0.057	**11.78 ± 0.16**	*0.000*
	T_6	25.13 ± 0.37	**15.87 ± 0.24**	*0.000*	**24.74 ± 0.38**	0.159	**16.12 ± 0.32**	*0.000*
F_2	T_1	59.32 ± 1.16	**52.76 ± 1.56**	*0.000*	59.79 ± 1.29	0.476	**50.98 ± 1.97**	*0.000*
	T_2	154.55 ± 1.43	**142.39 ± 3.00**	*0.000*	155.87 ± 2.31	0.382	**141.45 ± 6.47**	*0.000*
	T_3	144.66 ± 2.19	**142.71 ± 3.67**	0.213	147.59 ± 1.99	0.057	**138.18 ± 3.77**	*0.001*
	T_4	109.16 ± 1.99	**87.51 ± 3.17**	*0.000*	109.39 ± 2.08	0.820	**86.27 ± 3.14**	*0.000*
	T_5	182.75 ± 4.08	210.98 ± 3.01	*0.000*	184.87 ± 3.67	0.254	208.00 ± 2.85	*0.000*
	T_6	157.50 ± 2.02	**132.97 ± 2.94**	*0.000*	160.88 ± 2.57	0.173	**135.86 ± 4.14**	*0.000*
F_3	T_1	603.28 ± 33.91	651.14 ± 21.62	*0.037*	629.03 ± 38.33	0.106	**567.47 ± 63.52**	0.843
	T_2	954.22 ± 8.57	**944.15 ± 8.14**	0.177	**949.64 ± 5.98**	0.440	**948.69 ± 8.33**	0.406
	T_3	957.51 ± 6.51	**950.71 ± 7.31**	0.485	**949.20 ± 7.04**	0.230	**947.46 ± 7.91**	0.068
	T_4	823.01 ± 14.58	**813.42 ± 20.65**	0.912	**807.79 ± 24.27**	0.764	**803.66 ± 18.89**	0.116
	T_5	912.61 ± 16.99	928.88 ± 14.77	*0.033*	928.74 ± 15.26	*0.046*	928.02 ± 19.91	0.182
	T_6	977.84 ± 9.30	**975.48 ± 10.82**	0.832	990.33 ± 9.79	0.077	**972.99 ± 10.47**	0.343
F_4	T_1	104.09 ± 2.71	**75.07 ± 3.00**	*0.000*	106.57 ± 2.93	0.572	**74.63 ± 2.00**	*0.000*
	T_2	317.50 ± 4.86	**200.90 ± 5.37**	*0.000*	**314.00 ± 5.21**	0.390	**202.21 ± 5.31**	*0.000*
	T_3	292.79 ± 5.02	**203.19 ± 5.66**	*0.000*	**288.32 ± 4.59**	0.224	**209.94 ± 6.21**	*0.000*
	T_4	210.72 ± 4.30	**141.80 ± 5.39**	*0.000*	214.69 ± 3.45	0.197	**138.78 ± 4.03**	*0.000*
	T_5	354.22 ± 6.60	**315.53 ± 8.49**	*0.000*	359.61 ± 6.88	0.293	**323.35 ± 7.20**	*0.000*
	T_6	304.35 ± 6.22	**207.34 ± 5.38**	*0.000*	**304.12 ± 6.94**	0.686	**208.45 ± 5.25**	*0.000*
F_5	T_1	184.45 ± 4.56	**140.43 ± 2.70**	*0.000*	**179.84 ± 3.90**	0.112	**138.39 ± 2.74**	*0.000*
	T_2	431.23 ± 3.99	**281.12 ± 2.04**	*0.000*	432.09 ± 5.02	0.686	**283.31 ± 2.59**	*0.000*
	T_3	456.55 ± 8.15	**257.72 ± 4.07**	*0.000*	457.50 ± 5.65	0.582	**254.75 ± 4.22**	*0.000*
	T_4	323.49 ± 5.58	**211.48 ± 2.90**	*0.000*	**321.02 ± 7.11**	0.542	**209.22 ± 3.19**	*0.000*
	T_5	660.93 ± 15.61	**309.31 ± 4.75**	*0.000*	**655.21 ± 17.96**	0.495	**304.52 ± 4.42**	*0.000*
	T_6	530.49 ± 13.27	**274.13 ± 4.30**	*0.000*	**524.56 ± 11.99**	0.350	**272.89 ± 4.76**	*0.000*
F_6	T_1	122.00 ± 4.56	**88.10 ± 2.91**	*0.000*	124.65 ± 3.28	0.279	**89.67 ± 3.25**	*0.000*
	T_2	364.30 ± 12.74	**217.14 ± 20.63**	*0.000*	381.20 ± 12.71	0.051	**231.04 ± 10.06**	*0.000*
	T_3	415.76 ± 14.23	**279.81 ± 16.12**	*0.000*	432.63 ± 15.18	0.100	**305.07 ± 22.58**	*0.000*
	T_4	262.26 ± 18.88	**151.61 ± 11.73**	*0.000*	**254.35 ± 14.22**	0.832	**166.38 ± 14.31**	*0.000*
	T_5	498.69 ± 17.72	**383.18 ± 19.69**	*0.000*	525.48 ± 16.39	0.051	**396.75 ± 21.66**	*0.000*
	T_6	430.24 ± 21.31	**277.79 ± 19.30**	*0.000*	444.89 ± 26.90	0.440	**305.50 ± 18.43**	*0.000*

greater for the spherical peak function (denoted by S) than for the conical function. The substantial effect of different settings is also clear on the GDBG experiments. For example, compare the offline errors on function F_{1a} to errors on function F_3.

The standard set does not include variations of the number of dimensions, n_d, or change period, Cp. Experimental results on different settings of n_d and Cp showed that these values also had a considerable influence on offline errors. For example, consider Figure 4.7 which shows the offline error of DynDE on the MPB with the conical peak function and change severity of 1.0 as n_d and Cp are varied. The large effect of these settings is clear.

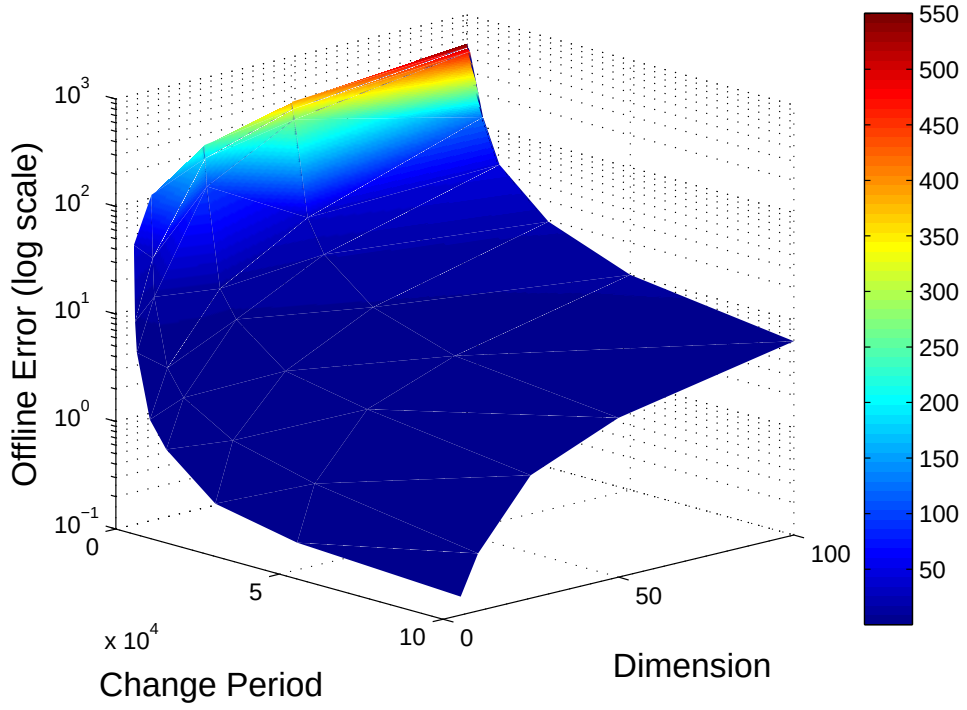

Figure 4.7: Offline error of DynDE on the MPB using the conical peak function with change severity 1.0 for various settings of number of dimensions and change period.

An analysis of the results found that the influence of the various settings is strongly interrelated, for example, a combination of high dimension and low change period results in an extremely high offline error in Figure 4.7. The following sections discuss the trends that were observed for each of the scalability settings, respectively, in the context of the other settings.

4.6.4.2 Trends from Varying the Change Period

Section 2.5.2 pointed out that the frequency of changes (determined by the change period) is one of the defining characteristics of a dynamic environment and argued that the change frequency determines the viability of solving a DOP. Figure 4.7 shows an exponential increase in offline error as the change period is reduced. A similar trend was found for all other experimental environments and algorithms with respect to change period. This trend exists because, as the change period is decreased, the number of available function evaluations between changes in the environment decreases. The algorithms are consequently less likely to locate optima which, in turn, leads to a higher offline error. Conversely, a large change period makes the discovery of optima more likely, which results in a lower offline error. An increase in change period results in diminishing returns, in terms of offline error, as the environment tends towards a static environment. This explains the comparatively low gradient of the curve for large change periods in Figure 4.7.

An analysis of the experimental data found that, in general, DynDE and RMC scaled similarly with respect to change period, but that CPE and CDE exhibited different scaling behaviour. All four algorithms performed similarly in the presence of very frequent changes of 100 function evaluations. This observation is to be expected in the case of CPE since the competition between sub-populations only commences subsequent to the second generation after a change in the environment. A change period of 100 causes a change in the environment before the second generation is completed. Accordingly, the CPE algorithm reduces to DynDE which explains the similar results. The RMC approach is aimed at situations where multiple optima are located within the exclusion radius of each other. RMC aims to prevent the unnecessary reinitialisation of sub-populations when converging to closely located optima. However, in the presence of frequent changes, the algorithm does not have enough time to converge to optima and RMC consequently performs similarly to DynDE.

CPE and CDE were found to be more scalable than DynDE and RMC when the change period was higher than 100, although it was found that the number of dimensions influenced the scalability. The performance of the algorithms on the GDBG function F_{1a}, with change type T_1, was selected as a representative example to illustrate the influence of

the number of dimensions on the scalability trend (refer to Figures 4.8 to 4.11). CPE and CDE scaled better than DynDE and RMC only in the lower change periods (ignoring the change period of 100) in low dimensions, with similar offline errors for the higher change period experiments, as shown in Figure 4.8. The gap between CPE and CDE versus DynDE and RMC widens in the 10 and 25 dimensional cases shown in Figures 4.9 and 4.10 but still yielded similar results for high change periods. However, in 100 dimensions CPE and CDE clearly scale better than DynDE and RMC, even for high change periods.

The trends shown in Figures 4.8 to 4.11 are specific examples, but CPE and CDE tended to scale better than DynDE and RMC in terms of change period on all functions and change types, especially in high dimensions. Only low dimensional experiments yielded situations where the the algorithms performed similarly for high change periods, as shown in Figure 4.9.

4.6.4.3 Trends from Varying the Number of Dimensions

Section 2.5.2 mentioned the number of dimensions as one of the factors that determines the hardness of a fitness landscape. One can generally assume that increasing the number of dimensions increases the hardness of the fitness landscape (refer to page 35) since the size of the search space is increased (there are, however, exceptions to this rule). The experimental results showed that, in general, increasing the number of dimensions resulted in a close-to-linear increase in offline error for all four algorithms. DynDE and RMC showed similar scaling behaviour, while the results of CPE and CDE were similar. Typically, CPE and CDE scaled better in terms of number of dimensions than DynDE and RMC, but it was found that this trend was influenced by the change period.

The performance of the algorithms on the GDBG function F_2 with change type T_4 was selected as a representative example of the scalability of the algorithms in terms of the number of dimensions with respect to change period. These trends are shown in Figures 4.12 to 4.15. The previous section found that all four algorithms performed similarly for low change periods. Figure 4.12 shows similar scaling behaviour over the number of dimensions for DynDE, RMC, CPE and CDE for a change period of 500. As the change period is increased to 5 000 (refer to Figure 4.13) CPE and CDE scales better than DynDE and RMC. A change period of 25 000 function evaluations results in

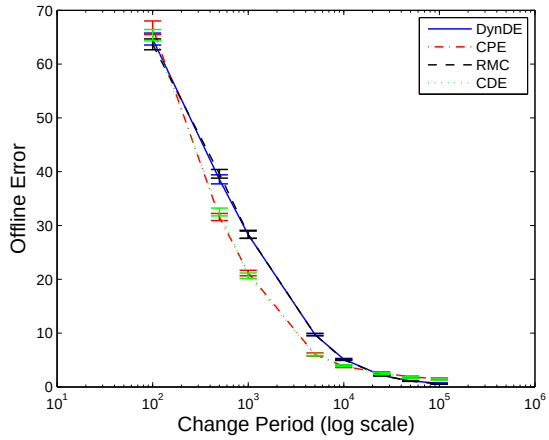

Figure 4.8: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_{1a} and change type T_1 for various settings of change period in 5 dimensions.

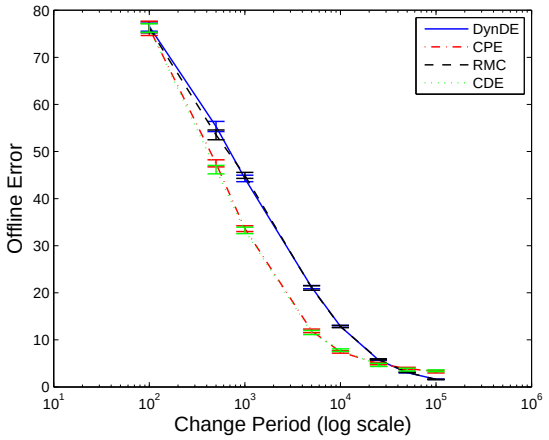

Figure 4.9: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_{1a} and change type T_1 for various settings of change period in 10 dimensions.

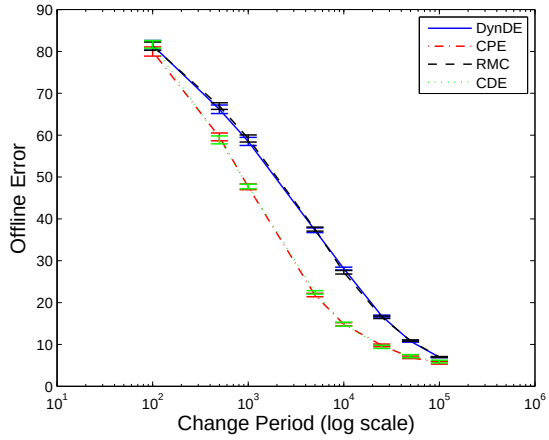

Figure 4.10: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_{1a} and change type T_1 for various settings of change period in 25 dimensions.

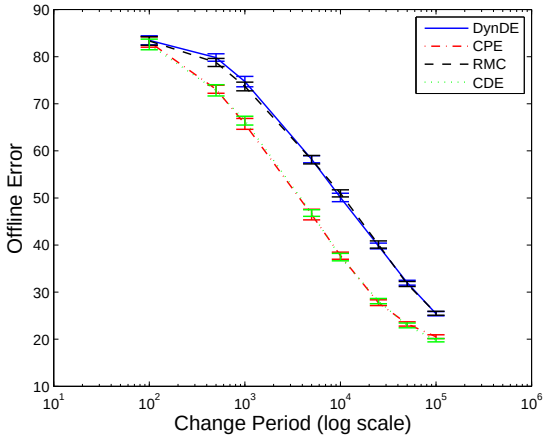

Figure 4.11: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_{1a} and change type T_1 for various settings of change period in 100 dimensions.

similar offline errors for all four algorithms in low dimensions (as shown in Figure 4.14), but with CPE and CDE still scaling better than DynDE and RMC in high dimensions. The diminishing performance of CPE and CDE with respect to DynDE and RMC in low dimensions with an increased change period of 100 000 is apparent in Figure 4.15 where CPE and CDE actually perform worse than DynDE and RMC in low dimensions, although better performance is still observed in high dimensions.

The general trends observed in the examples shown in Figures 4.12 to 4.15 were found in the majority of the benchmark settings that were investigated. CPE and CDE, in general, scale better than DynDE and RMC in terms of number of dimensions. The increase in offine error with the increase in the number of dimensions fits the hypothesis that increasing the number of dimensions makes the DOP harder. The diminished scalability of CPE and CDE in low dimensions with high change periods, in comparison with DynDE and RMC, is explained by the fact that the large change periods allow sufficient function evaluations for all algorithms to locate optima, hence resulting in similar offline errors. As the dimension increases and the problem becomes harder, the more scalable algorithms, CPE and CDE, give lower offline errors.

4.6.4.4 Trends from Varying the Change Severity

The severity of changes in a dynamic environment was identified as a critical factor which determines the ability of an algorithm to solve a DOP (refer to Section 2.5.2). The result of significantly large changes is that the new environment bears no relation to the previous environment. The usefulness of information, gathered about an environment before a change, thus diminishes as the severity of the changes increases. The offline error of an optimisation algorithm is, consequently, expected to increase as the changes become more severe. Experimental results from the experiments on the MPB showed that a considerable increase in offline error for all algorithms does, in fact, result from increasing the change severity.

CPE and CDE, typically, scaled better with respect to change severity than DynDE and RMC. The trends, however, were once again found to be heavily dependent on the change period. Results of the algorithms on the spherical peak function of the MPB in 10 dimensions are used as an example to illustrate the general trend (refer to Figures 4.16 to

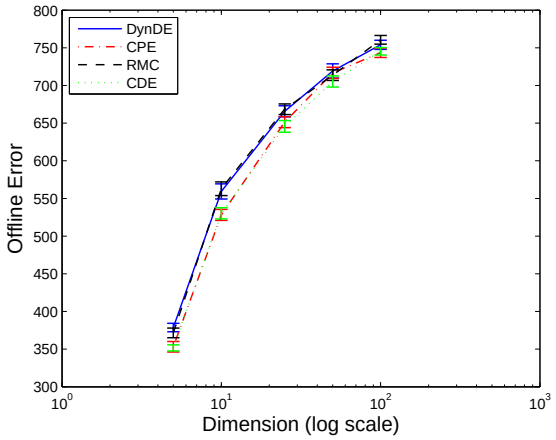

Figure 4.12: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_2 and change type T_4 for various settings of dimension with a change period of 500 function evaluations.

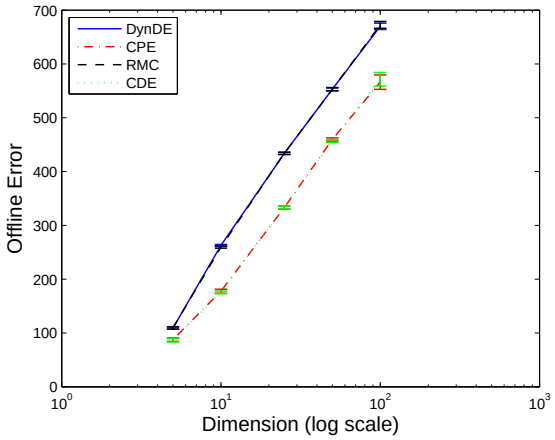

Figure 4.13: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_2 and change type T_4 for various settings of dimension with a change period of 5 000 function evaluations.

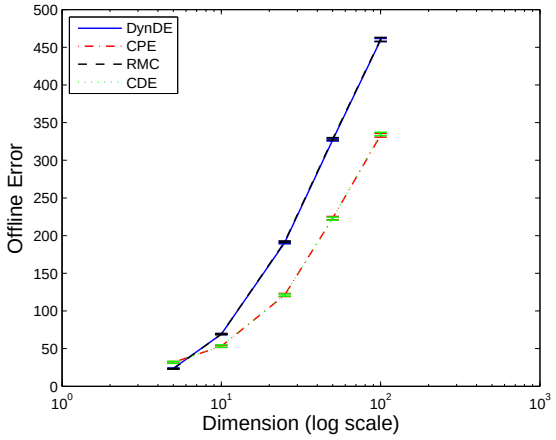

Figure 4.14: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_2 and change type T_4 for various settings of dimension with a change period of 25 000 function evaluations.

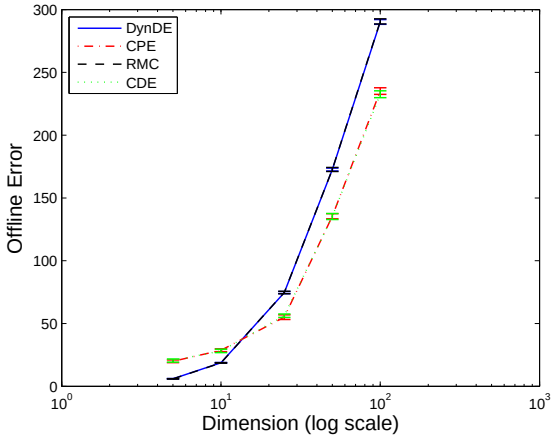

Figure 4.15: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using peak function F_2 and change type T_4 for various settings of dimension with a change period of 100 000 function evaluations.

4.19). Small change periods, as shown in Figure 4.16 resulted in similar scaling behaviour for all four algorithms. A higher change period results in better scaling by CPE and CDE than DynDE and RMC, as shown in Figure 4.17. The gap between the offline errors of CPE and CDE versus DynDE and RMC increases in proportion as the change period is increased to 25 000 in Figure 4.18 and to 100 000 in Figure 4.19.

The superior scalability of CPE and CDE, in terms of change severity, was found to be especially pronounced when using the spherical peak function, but was also present, to a lesser degree, in the conical peak function experiments. The improved scalability of CPE and CDE, with an increasing change period, is in contrast with trends found with respect to scalability in terms of dimension (refer to Section 4.6.4.3) and change period in general (refer to Sections 4.6.4.2). A diminished difference between the scaling behaviour of CPE and CDE versus DynDE and RMC was previously found when increasing the change period.

The superior scaling behaviour exhibited by CPE and CDE is due to large errors resulting from severe changes in the environment, which typically leaves sub-populations located far from the optima. The competing population approach evolves only the best sub-population at any given time and the current error thus decreases in fewer function evaluations than it would when using normal DynDE.

Figure 4.20 shows the offline and current errors of DynDE and CPE on the MPB using the spherical peak function in 10 dimensions with a change period of 5 000, while Figure 4.21 gives the same results with a change period of 100 000. A large change severity value of 80.0 is used. Note that an increase in current error of about 3 000 occurs for both algorithms after changes in the environments, which is roughly equal to the current error at the commencement of the optimisation process. The current error of the CPE algorithm decreases faster than that of DynDE after each change. This allows CPE to achieve a lower current error than DynDE when a low change period is used (note the difference in offline error between DynDE and CPE immediately before changes in Figure 4.20), which lead to a lower offline error.

Large change periods allow both algorithms to attain a current error approaching zero before each change (refer to Figure 4.21), but the current error of CPE reduces faster than that of DynDE. This tendency can be more clearly observed in Figure 4.22, which

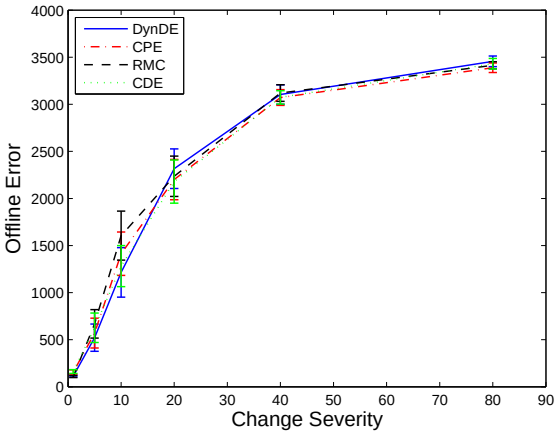

Figure 4.16: Offline errors of DynDE, CPE, RMC, and CDE on the MPB using the spherical peak function in 10 dimensions for various settings of change severity with a change period of 500 function evaluations.

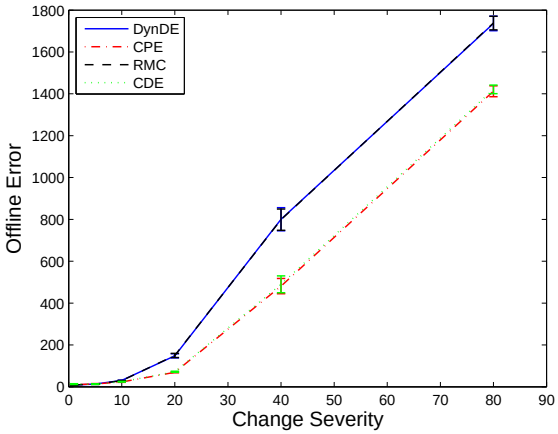

Figure 4.17: Offline errors of DynDE, CPE, RMC, and CDE on the MPB using the spherical peak function in 10 dimensions for various settings of change severity with a change period of 5 000 function evaluations.

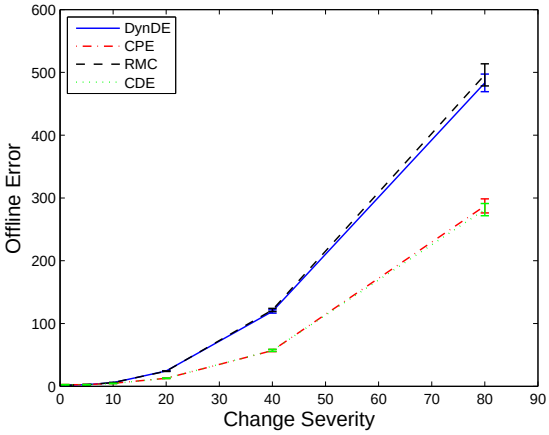

Figure 4.18: Offline errors of DynDE, CPE, RMC, and CDE on the MPB using the spherical peak function in 10 dimensions for various settings of change severity with a change period of 25 000 function evaluations.

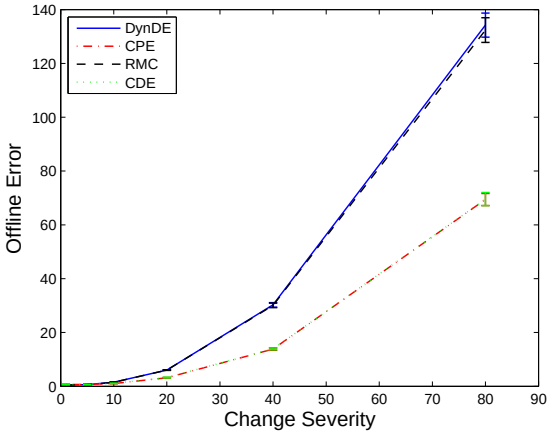

Figure 4.19: Offline errors of DynDE, CPE, RMC, and CDE on the MPB using the spherical peak function in 10 dimensions for various settings of change severity with a change period of 100 000 function evaluations.

shows an enlargement of Figure 4.21 around the first change in the environment. The CPE current error approaches zero in roughly half the function evaluations as does that of DynDE. The faster reduction in error is reflected in the offline error (since it averages over all the best errors found).

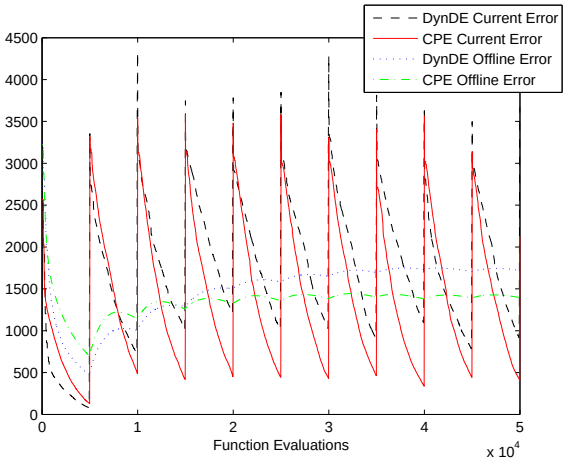

Figure 4.20: Offline and current errors of DynDE and CPE on the MPB spherical function in 10 dimensions, a change period of 5 000, and a change severity of 80.0.

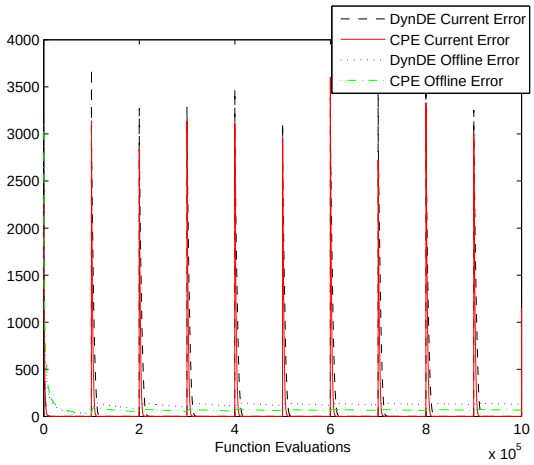

Figure 4.21: Offline and current errors of DynDE and CPE on the MPB spherical function in 10 dimensions, a change period of 100 000, and a change severity of 80.0.

Figure 4.23 shows an enlargement of the current errors of DynDE and CPE in the presence of a change period of 100 000 but with a minor change severity of 1.0. The small change severity of Figure 4.23 (compared to Figure 4.22) results in comparatively small increases in current error after a change in the environment. Both algorithms recover quickly which results in similar behaviour by the two algorithms. The competitive population evaluation approach is thus more useful in the presence of severe changes in the environment.

4.6.4.5 Trends from Various Functions

The underlying function, which determines the topology of the search landscape, was identified in Section 2.5.2 as a factor that influences the intuitive concept of the hardness of the fitness landscape. The underlying function used in a benchmark is consequently expected to influence the performance of a DOP algorithm.

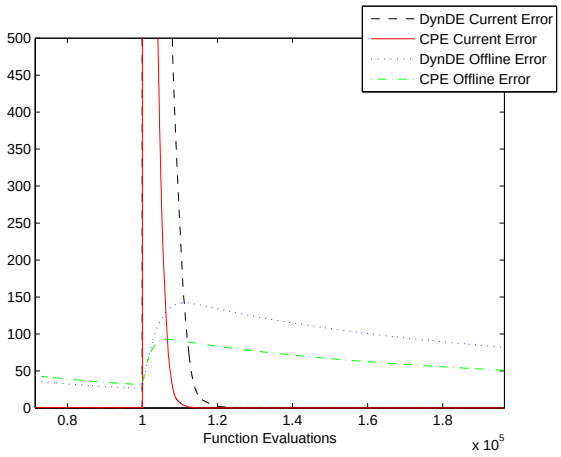

Figure 4.22: Offline and current errors of DynDE and CPE on the MPB spherical function in 10 dimensions, a change period of 100 000, and a change severity of 80.0. The area around the first change is enlarged.

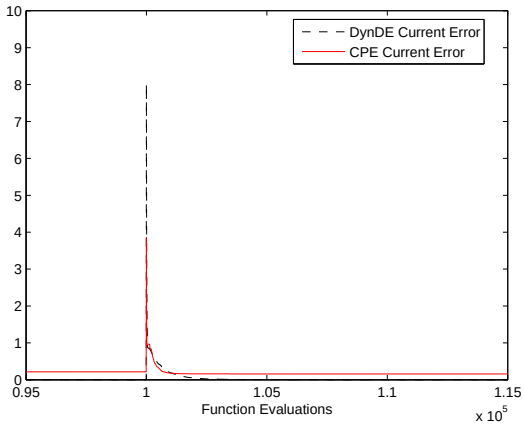

Figure 4.23: Current errors of DynDE and CPE on the MPB spherical function in 10 dimensions, a change period of 100 000, and a change severity of 1.0. The area around the first change is enlarged.

The seven functions of the GDBG listed in Table 4.2 are used in this section to observe the effect of the function on the optimisation process. The number of dimensions was found to have a large impact on the relative performance of DynDE, CPE, RMC and CDE on the seven functions. The performance of the algorithms on environments with a change type T_1 and a change period of 10 000 was selected as a representative example to illustrate the general trends observed over all change types, and the influence of the number of dimensions on the scalability trend (refer to Figures 4.24 to 4.27). Low dimensional experiments typically resulted in the behaviour shown in Figure 4.24. Function F_3, which is a composition of Rastrigin's function (refer to Section 2.5.4.1), yielded a much higher offline error than the other functions. DynDE, CPE, RMC and CDE all performed similarly on F_3, but the performance of DynDE and RMC differed from that of CPE and CDE. Function F_5 shows a more rapid growth in offline error than the other functions, as the number of dimensions is increased (refer to Figure 4.25), while the difference in performance between DynDE and RMC versus CPE and CDE grows.

At high dimensions (refer to Figure 4.26), the offline error on F_5 exceeds that of F_3 while the gap between the performance of DynDE and RMC versus CPE and CDE narrows until their behaviour becomes very similar (refer to Figure 4.27). Note that the

performance of DynDE and RMC versus CPE and CDE still differs for functions F_2, F_4 and F_6 in Figure 4.27.

The general trend that is thus observed is that in low dimensions, function F_3 presents the greatest challenge to the optimisation algorithms, followed by F_5, F_6, F_3, F_2, and finally F_1. High dimensions alter this trend to function F_5 posing the greatest challenge to the algorithms, followed by F_3, F_6, F_4, F_2, and F_1. Function F_3 is comparatively unaffected by increasing the number of dimensions, and causes the most similar scaling behaviour in DynDE, CPE, RMC and CDE. CPE and CDE perform better than DynDE and RMC on all other functions, especially in lower dimensions.

4.6.4.6 Trends from Various Change Types

The six change types of the GDBG listed in Table 4.2 are used in this section to observe the effect of the change type on the optimisation process. The experimental results did not reveal clear, general trends as was the case with the other settings that were varied. The effect of the change type on the optimisation algorithm was found to depend on the underlying function.

Table 4.10 was created by ranking the offline error of DynDE on each change type in ascending order over all experiments for each function. The ranking of each change type was then averaged to give an average ranking per change type, per function. Over all functions, T_1 ranked the highest, which means that, on average, DynDE yielded the lowest offline error on T_1. T_3, T_2, and T_4 received similar rankings, while T_5 and T_6 received lower rankings. T_1 always received the highest ranking while T_6 always ranked the lowest per function, but the rankings per function did differ considerably from the overall ranking in several cases. For example, consider the ranking on function F_3, where T_5 received a relatively high rank and T_2 received a relatively low rank.

Section 4.6.4.2 identified a trend in change period wherein CPE and CDE performed better than DynDE and RMC at low change periods, while DynDE and RMC outperformed CPE and CDE at high change periods. An analysis of the experimental data found that the point where DynDE and RMC start outperforming CPE and CDE depends on the change type for each function. As an example, the offline error per change type of DynDE, CPE, RMC, and CDE on function F_{1a} in 10 dimensions is given in Figures

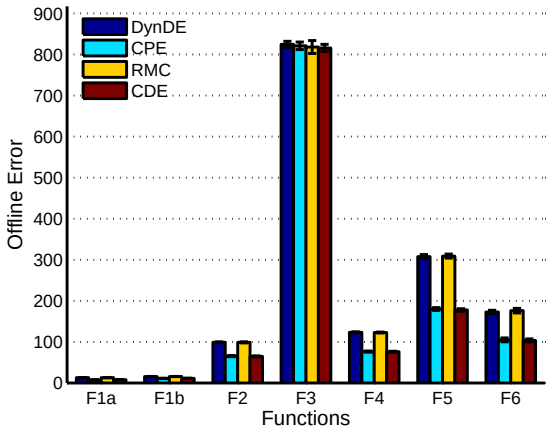

Figure 4.24: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using change type T_1 for various functions in 10 dimensions with a change period of 10 000 function evaluations.

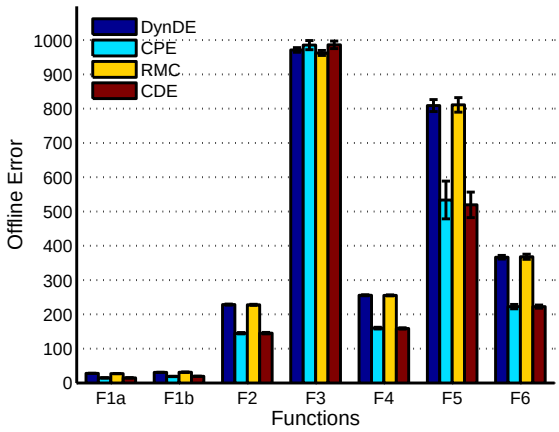

Figure 4.25: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using change type T_1 for various functions in 25 dimensions with a change period of 10 000 function evaluations.

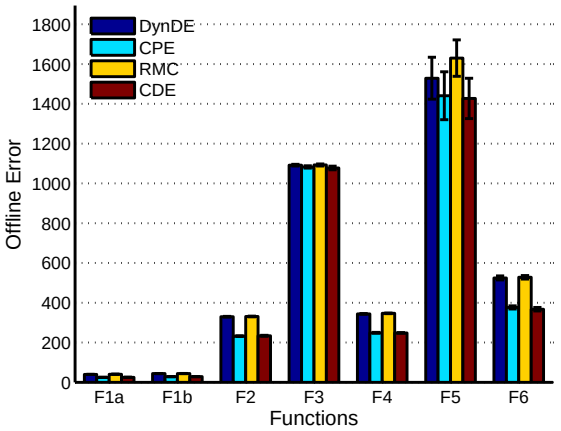

Figure 4.26: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using change type T_1 for various functions in 50 dimensions with a change period of 10 000 function evaluations.

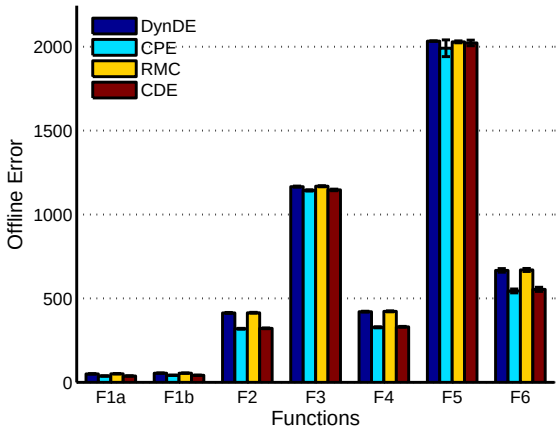

Figure 4.27: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using change type T_1 for various functions in 100 dimensions with a change period of 10 000 function evaluations.

Table 4.10: Average ranking of change type per function

F_{1a}	F_{1b}	F_2	F_3	F_4	F_5	F_6	**All**
T_1 - 1.30	T_1 - 1.80	T_1 - 1.00	T_1 - 1.00	T_1 - 1.00	T_1 - 1.28	T_1 - 1.00	T_1 - 1.20
T_3 - 2.38	T_2 - 2.35	T_4 - 2.95	T_3 - 2.83	T_4 - 2.93	T_4 - 2.80	T_4 - 2.05	T_3 - 3.23
T_2 - 2.58	T_3 - 2.58	T_3 - 3.55	T_5 - 3.38	T_3 - 3.55	T_2 - 3.03	T_2 - 3.23	T_2 - 3.26
T_4 - 4.30	T_5 - 3.60	T_2 - 3.63	T_4 - 3.55	T_2 - 3.80	T_3 - 3.58	T_3 - 4.15	T_4 - 3.36
T_5 - 4.53	T_4 - 4.98	T_5 - 4.73	T_2 - 4.25	T_5 - 4.73	T_5 - 4.60	T_5 - 5.15	T_5 - 4.39
T_6 - 5.93	T_6 - 5.70	T_6 - 5.15	T_6 - 6.00	T_6 - 5.00	T_6 - 5.73	T_6 - 5.43	T_6 - 5.56

4.28 to 4.31.

Very similar offline errors were found for each algorithm in Figure 4.28, which shows the performance with a change period of 100. Figure 4.29, where a change period of 5 000 was used, shows clearly better performance for CPE and CDE than for DynDE and RMC. The diminishing difference in performance between CPE and CDE versus DynDE and RMC is shown in Figure 4.30 (at a change period of 25 000 function evaluations), where T_4 no longer shows clear differences between CPE and CDE versus DynDE and RMC, and the confidence bars of the algorithms on T_3 overlap. Figure 4.31 shows the performance of the algorithms at a change period of 100 000, where DynDE and RMC generally outperformed CPE and CDE. The change type thus has an influence on how early, in terms of the change period setting, the change over point between the performances of the algorithms occurs.

4.6.4.7 Summary for Research Question 2

The trends that emerged from varying the benchmark settings are summarised below for each setting:

Change Period: Increasing the change period resulted in a reduction of offline error. The algorithms have more function evaluations available between changes in the environment which resulted in lower offline errors. CPE and CDE scaled better than DynDE and RMC as changes became more frequent.

Number of Dimensions: Larger numbers of dimensions resulted in larger offline errors. CPE and CDE scaled better than DynDE and RMC as the number of dimensions increased.

Change Severity: More severe changes in the environment resulted in higher offline

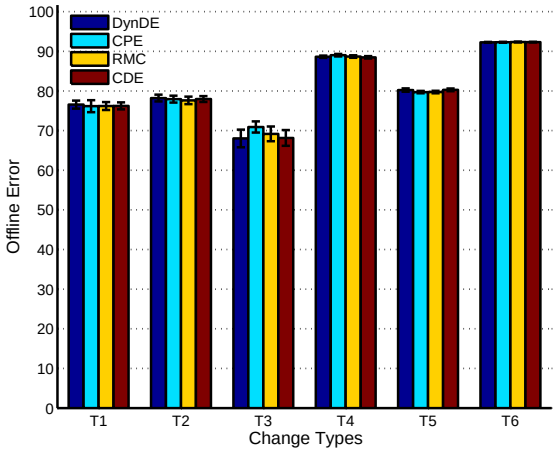

Figure 4.28: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using function F_{1a} for various change types in 10 dimensions with a change period of 100 function evaluations.

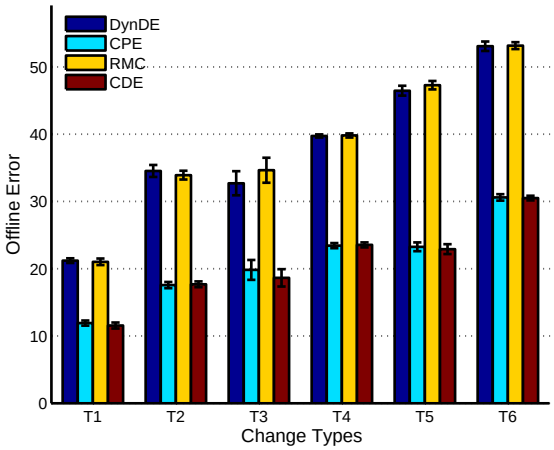

Figure 4.29: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using function F_{1a} for various change types in 10 dimensions with a change period of 5 000 function evaluations.

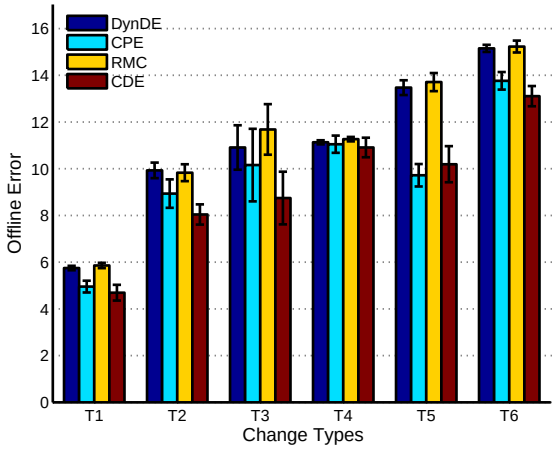

Figure 4.30: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using function F_{1a} for various change types in 10 dimensions with a change period of 25 000 function evaluations.

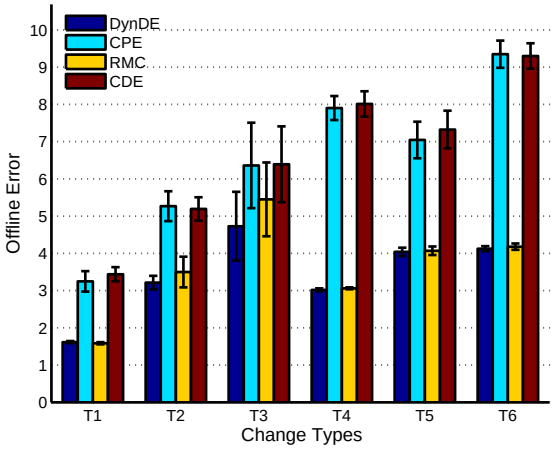

Figure 4.31: Offline errors of DynDE, CPE, RMC, and CDE on the GDBG using function F_{1a} for various change types in 10 dimensions with a change period of 100 000 function evaluations.

errors, as information, gathered before changes, became less relevant. CPE and CDE scaled better than DynDE and RMC as changes became more severe.

Function: The underlying benchmark function was found to have a strong influence on the offline errors of the optimisation algorithms. The underlying function also determines the scalability of each algorithm in terms of dimension. CPE and CDE typically performed better than DynDE and RMC on the various functions.

Change Type: The effect of the change type was found to be related to the function that was being optimised. The change types influence the point where DynDE and RMC starts to outperform CPE and CDE due to an increased change period.

4.6.5 Research Question 3

Are RMC, CPE and CDE more effective dynamic optimisation algorithms than DynDE?

The discussion related to the previous research question identified trends which suggest that CPE and CDE are superior to DynDE and RMC in terms of scalability over the change period, number of dimensions and change severity. The aim of this section is to determine whether RMC, CPE and CDE give significantly lower offline errors than DynDE and are thus more effective algorithms for DOPs.

Table 4.9 gives the offline errors of DynDE, CPE, RMC and CDE on the standard set of experiments which correspond to the benchmark settings in Tables 4.1 and 4.2. The respective offline errors of CPE, RMC and CDE are printed in boldface in shaded cells where the errors are lower than that of DynDE. The outcomes of Mann-Whitney U tests, comparing the results of each algorithm to those of DynDE, are printed in italics if differences in average offline error are statistically significant at a 95% confidence level (i.e. are lower than 0.05). The results in Table 4.9 show that CPE and CDE clearly outperformed DynDE in the majority of the settings of the standard set. RMC, however, yielded a significantly lower offline error in only four cases and was outperformed by DynDE in one case.

The standard set of experiments is not extensive enough to draw meaningful conclusions about the performance of the three algorithms compared to that of DynDE. Experiments were consequently conducted on variations of the standard set, over all com-

binations of settings of change period and number of dimensions given in Table 4.3. The results were analysed by counting the number of times that DynDE was outperformed by each of the algorithms and the number of times that each of the algorithms outperformed DynDE. Only results that were statistically significantly different were considered.

Comparative results for each of the algorithms are summarised in Tables 4.11 to 4.16. Each cell in the tables gives the number of cases in which the relevant algorithm outperformed DynDE (indicated by ↑) and the number of cases in which DynDE outperformed that specific algorithm (indicated by ↓). The results were aggregated by function, change type and change severity for all the values of the change period and number of dimensions and totals given to support the analysis of the data. The value in column **Max** indicates the maximum possible score that can be found in the cells in each row. Note that rows labelled **All**, which gives aggregated results per change period, do not give totals of values in each row as some results are duplicated in the stratifications of the different categories. For example, results for each function are summed over all change types, while the result for each change type is summed over all functions.

The following sections, respectively discuss the performance of CPE, RMC and CDE compared to DynDE. CPE is compared to DynDE in Section 4.6.5.1, RMC is compared to DynDE in Section 4.6.5.2, and CDE is compared to DynDE in Section 4.6.5.3. Concluding remarks on this research question are given in 4.6.5.4.

4.6.5.1 CPE compared to DynDE

Table 4.11 summarises the comparison between the CPE and DynDE results in 5, 10 and 25 dimensions, while Table 4.12 gives the summary for 50 and 100 dimensions, and results summarised over all the dimensions that were investigated. CPE performed significantly better than DynDE in 1 344 experiments out of a total of 2 160 (i.e. 62.2% of all experiments). DynDE performed better than CPE in only 219 experiments (10.1% of all experiments). This clearly indicates that CPE is a more effective algorithm for DOPs than DynDE. The majority of the cases where DynDE performed better than CPE occurred when using a change period 50 000 or higher.

Section 4.3 hypothesised that the competing populations approach could potentially allocate too few function evaluations to weaker sub-populations when changes in the en-

vironment are infrequent, which could lead to inferior performance. This hypothesis has been proved to be correct by the analysis presented in Table 4.11. However, the inferior performance of CPE on high change period problems only occurs in low dimensions (compare the 5 and 10 dimensional results to the 25, 50 and 100 dimensional cases in Tables 4.11 and 4.12). The aggregate analysis over all dimensions, which is given in Table 4.12, shows that, when all dimensions are considered, CPE performed better than DynDE in more cases, even when large change periods were used. These results agree with the trends observed, in terms of change period and number of dimensions, in the scalability study presented in Section 4.6.4.

Experiments, in which a change period of 100 function evaluations were used, resulted in the fewest cases of statistically significant differences between DynDE and CPE. This is because the competing populations process only commences after two generations, following a change in the environment. The two algorithms are identical when a change period of 100 is used, because the population size employed in the experiments results in successive changes in the environment occurring in less than two generations.

The trend that was observed in Section 4.6.4.4 in respect of change severity is confirmed by CPE outperforming DynDE more often when a high change severity is used, in higher than 10 dimensions. CPE outperformed DynDE more often on the MPB's spherical peak function in five dimensions, but in higher dimensions, CPE performed better more often on the conical peak function. The GDBG F_3 function consistently resulted in the fewest number of statistically significantly differing results between CPE and DynDE. Function F_5 resulted in a comparatively large number of cases where CPE outperformed DynDE in low dimensions, but resulted in a comparatively small number of superior results by CPE in high dimensions, thus confirming the trend observed in Section 4.6.4.5.

The analysis described in this section does not give an indication of the magnitude of the improvement of CPE over DynDE. The experimental data was analysed to determine the average percentage improvement (API) in offline error of CPE over DynDE. The API is calculated using:

Table 4.11: CPE vs DynDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*1*	↑1 ↓1	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑3 ↓2**
5	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑12 ↓0**
20	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑6 ↓1**
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑4 ↓1**
C	(6)	↑0 ↓0	**↑1 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑1 ↓0**	↑*1 ↓2*	**↑16 ↓2**
S	(6)	↑*0 ↓1*	**↑2 ↓1**	**↑3 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑28 ↓2**
		GDBG								
F_{1a}	(6)	↑*0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑*0 ↓3*	↑*0 ↓6*	↑*0 ↓6*	**↑24 ↓17**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	↑*0 ↓5*	↑*0 ↓6*	↑*0 ↓6*	**↑23 ↓17**
F_2	(6)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑4 ↓1**	↑*2 ↓4*	↑*0 ↓6*	↑*0 ↓6*	↑*0 ↓6*	↑*15 ↓23*
F_3	(6)	↑*0 ↓1*	**↑1 ↓0**	**↑4 ↓0**	↑*0 ↓2*	↑*0 ↓1*	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑7 ↓4**
F_4	(6)	↑*0 ↓2*	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓1**	↑*0 ↓6*	↑*0 ↓6*	**↑22 ↓15**
F_5	(6)	↑*0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	↑*1 ↓3*	↑*0 ↓6*	**↑29 ↓12**
F_6	(6)	**↑1 ↓0**	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑4 ↓2**	↑*1 ↓2*	**↑31 ↓5**
T_1	(7)	↑*0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑4 ↓1**	↑*0 ↓6*	↑*0 ↓6*	↑*1 ↓6*	**↑23 ↓21**
T_2	(7)	↑*0 ↓1*	**↑4 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑3 ↓2**	↑*1 ↓5*	↑*0 ↓5*	**↑26 ↓14**
T_3	(7)	↑0 ↓0	**↑4 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑3 ↓1**	↑*1 ↓4*	↑*1 ↓5*	**↑26 ↓11**
T_4	(7)	↑*0 ↓1*	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑*0 ↓4*	↑*0 ↓6*	↑*0 ↓6*	**↑25 ↓17**
T_5	(7)	↑*0 ↓2*	**↑3 ↓0**	**↑5 ↓0**	**↑5 ↓2**	**↑5 ↓1**	↑2 ↓2	↑*2 ↓4*	↑*0 ↓5*	**↑22 ↓16**
T_6	(7)	↑1 ↓1	**↑4 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	↑3 ↓3	↑*1 ↓4*	↑*1 ↓5*	**↑29 ↓14**
All	(54)	↑*1 ↓7*	**↑30 ↓1**	**↑45 ↓0**	**↑42 ↓3**	**↑38 ↓5**	**↑19 ↓18**	↑*11 ↓29*	↑*9 ↓34*	**↑195 ↓97**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑0 ↓0	↑1 ↓1	↑*0 ↓1*	↑*0 ↓1*	↑*0 ↓2*	↑*3 ↓7*
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑6 ↓1**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
C	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑35 ↓1**
S	(6)	↑0 ↓0	↑*0 ↓1*	**↑2 ↓1**	**↑4 ↓0**	**↑4 ↓2**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑22 ↓7**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	↑*0 ↓5*	↑*0 ↓6*	**↑27 ↓11**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	↑*0 ↓5*	↑*0 ↓6*	**↑28 ↓11**
F_2	(6)	↑*0 ↓1*	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓1**	↑3 ↓3	↑*0 ↓4*	↑*0 ↓6*	**↑20 ↓15**
F_3	(6)	↑*0 ↓1*	**↑3 ↓0**	**↑5 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑1 ↓1	**↑11 ↓2**
F_4	(6)	↑0 ↓0	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑3 ↓2**	↑*2 ↓4*	↑*0 ↓6*	**↑23 ↓13**
F_5	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	**↑40 ↓2**
F_6	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑39 ↓1**
T_1	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	↑*2 ↓4*	↑*1 ↓6*	**↑33 ↓10**
T_2	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑4 ↓1**	↑*2 ↓4*	↑*2 ↓4*	**↑33 ↓9**
T_3	(7)	↑*0 ↓1*	**↑3 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑4 ↓0**	↑2 ↓2	↑2 ↓2	↑*2 ↓5*	**↑27 ↓10**
T_4	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓2**	↑*1 ↓5*	**↑35 ↓7**
T_5	(7)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓2**	**↑4 ↓2**	↑*2 ↓4*	↑*2 ↓4*	**↑26 ↓12**
T_6	(7)	↑*0 ↓1*	**↑4 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	↑*2 ↓4*	**↑34 ↓7**
All	(54)	↑*0 ↓2*	**↑33 ↓1**	**↑48 ↓1**	**↑46 ↓0**	**↑43 ↓4**	**↑35 ↓6**	**↑22 ↓19**	↑*18 ↓30*	**↑245 ↓63**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑7 ↓1**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑9 ↓0**
10	(2)	↑*0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓1**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
C	(6)	↑*0 ↓1*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑37 ↓1**
S	(6)	↑0 ↓0	↑1 ↓1	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑26 ↓1**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	**↑37 ↓2**
F_{1b}	(6)	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑*1 ↓2*	**↑38 ↓2**
F_2	(6)	↑0 ↓0	**↑5 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑34 ↓0**
F_3	(6)	↑0 ↓0	↑0 ↓0	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓1**	↑*0 ↓2*	↑*0 ↓3*	↑*0 ↓4*	**↑14 ↓10**
F_4	(6)	**↑1 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓1**	**↑34 ↓1**
F_5	(6)	↑0 ↓0	**↑5 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑38 ↓0**
F_6	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
T_1	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑42 ↓4**
T_2	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓0**	↑2 ↓2	**↑38 ↓2**
T_3	(7)	↑0 ↓0	**↑4 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑3 ↓1**	**↑38 ↓2**
T_4	(7)	**↑1 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑4 ↓3**	**↑43 ↓4**
T_5	(7)	↑0 ↓0	**↑4 ↓0**	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑36 ↓1**
T_6	(7)	**↑1 ↓0**	**↑3 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	**↑38 ↓2**
All	(54)	**↑2 ↓1**	**↑34 ↓1**	**↑45 ↓0**	**↑51 ↓0**	**↑50 ↓1**	**↑47 ↓2**	**↑41 ↓3**	**↑28 ↓9**	**↑298 ↓17**

Table 4.12: CPE vs DynDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑12 ↓0**
10	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑10 ↓0**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
C	(6)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑36 ↓0**
S	(6)	↑0 ↓0	**↑1 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑34 ↓0**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
F_{1b}	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑40 ↓0**
F_2	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑38 ↓1**
F_3	(6)	*↑0 ↓1*	**↑1 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	↑0 ↓0	*↑0 ↓3*	*↑0 ↓5*	**↑16 ↓9**
F_4	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑37 ↓1**
F_5	(6)	↑0 ↓0	**↑2 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑3 ↓1**	**↑27 ↓1**
F_6	(6)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑35 ↓0**
T_1	(7)	*↑0 ↓1*	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓2**	**↑37 ↓4**
T_2	(7)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓3**	**↑38 ↓3**
T_3	(7)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑40 ↓2**
T_4	(7)	↑0 ↓0	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑41 ↓2**
T_5	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑41 ↓1**
T_6	(7)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑36 ↓0**
All	(54)	*↑0 ↓1*	**↑30 ↓0**	**↑45 ↓0**	**↑52 ↓0**	**↑52 ↓0**	**↑45 ↓0**	**↑42 ↓3**	**↑37 ↓8**	**↑303 ↓12**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	↑1 ↓1	**↑11 ↓2**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑11 ↓3**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑10 ↓3**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	↑1 ↓1	**↑11 ↓2**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑12 ↓1**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
C	(6)	↑0 ↓0	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓0**
S	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	↑3 ↓3	↑3 ↓3	*↑2 ↓4*	**↑31 ↓11**
		GDBG								
F_{1a}	(6)	*↑0 ↓1*	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓1**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓0**
F_2	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑40 ↓2**
F_3	(6)	↑0 ↓0	*↑0 ↓2*	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓0**	*↑0 ↓2*	*↑0 ↓5*	**↑19 ↓9**
F_4	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑41 ↓2**
F_5	(6)	↑0 ↓0	*↑1 ↓2*	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑20 ↓2**
F_6	(6)	*↑0 ↓1*	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑34 ↓3**
T_1	(7)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑39 ↓3**
T_2	(7)	↑0 ↓0	**↑4 ↓1**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑4 ↓3**	**↑42 ↓5**
T_3	(7)	↑0 ↓0	**↑4 ↓1**	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑39 ↓2**
T_4	(7)	*↑0 ↓2*	**↑5 ↓1**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑45 ↓4**
T_5	(7)	↑0 ↓0	**↑5 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑37 ↓4**
T_6	(7)	*↑0 ↓1*	**↑3 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑33 ↓1**
All	(54)	*↑0 ↓4*	**↑33 ↓4**	**↑46 ↓0**	**↑50 ↓0**	**↑50 ↓1**	**↑46 ↓3**	**↑40 ↓6**	**↑38 ↓12**	**↑303 ↓30**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	*↑0 ↓1*	**↑6 ↓3**	**↑7 ↓1**	**↑5 ↓0**	**↑7 ↓1**	**↑4 ↓2**	**↑5 ↓1**	↑3 ↓3	**↑37 ↓12**
5	(10)	↑0 ↓0	**↑5 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓1**	**↑8 ↓1**	**↑5 ↓1**	**↑3 ↓1**	**↑48 ↓4**
10	(10)	*↑0 ↓1*	**↑4 ↓0**	**↑7 ↓0**	**↑9 ↓0**	**↑9 ↓1**	**↑10 ↓0**	**↑8 ↓1**	**↑6 ↓1**	**↑53 ↓4**
20	(10)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑9 ↓0**	**↑8 ↓1**	**↑59 ↓2**
40	(10)	↑0 ↓0	**↑2 ↓0**	**↑7 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑8 ↓1**	**↑9 ↓1**	**↑53 ↓2**
80	(10)	↑0 ↓0	**↑1 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓1**	**↑52 ↓1**
C	(6)	*↑0 ↓1*	**↑14 ↓0**	**↑27 ↓0**	**↑27 ↓0**	**↑27 ↓0**	**↑26 ↓0**	**↑23 ↓0**	**↑17 ↓3**	**↑161 ↓4**
S	(6)	*↑0 ↓1*	**↑10 ↓3**	**↑19 ↓1**	**↑23 ↓0**	**↑24 ↓3**	**↑23 ↓4**	**↑21 ↓4**	**↑21 ↓5**	**↑141 ↓21**
		GDBG								
F_{1a}	(30)	*↑0 ↓3*	**↑25 ↓0**	**↑27 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑22 ↓3**	**↑18 ↓11**	**↑15 ↓14**	**↑167 ↓31**
F_{1b}	(30)	**↑1 ↓0**	**↑29 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑22 ↓5**	**↑18 ↓11**	*↑12 ↓14*	**↑171 ↓30**
F_2	(30)	*↑0 ↓2*	**↑20 ↓0**	**↑26 ↓0**	**↑28 ↓1**	**↑24 ↓5**	**↑21 ↓9**	**↑15 ↓10**	*↑13 ↓14*	**↑147 ↓41**
F_3	(30)	*↑0 ↓3*	**↑5 ↓2**	**↑23 ↓0**	**↑17 ↓2**	**↑15 ↓2**	**↑4 ↓2**	*↑0 ↓8*	*↑3 ↓15*	**↑67 ↓34**
F_4	(30)	*↑1 ↓3*	**↑19 ↓0**	**↑26 ↓0**	**↑30 ↓0**	**↑28 ↓1**	**↑24 ↓3**	**↑17 ↓10**	*↑12 ↓15*	**↑157 ↓32**
F_5	(30)	*↑0 ↓1*	**↑20 ↓2**	**↑23 ↓0**	**↑26 ↓0**	**↑26 ↓0**	**↑23 ↓2**	**↑20 ↓3**	**↑16 ↓9**	**↑154 ↓17**
F_6	(30)	↑1 ↓1	**↑18 ↓0**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑27 ↓1**	**↑24 ↓3**	**↑21 ↓4**	**↑179 ↓9**
T_1	(35)	*↑0 ↓3*	**↑28 ↓0**	**↑31 ↓0**	**↑30 ↓1**	**↑28 ↓2**	**↑21 ↓7**	**↑18 ↓13**	**↑18 ↓16**	**↑174 ↓42**
T_2	(35)	*↑0 ↓1*	**↑24 ↓1**	**↑32 ↓0**	**↑33 ↓0**	**↑33 ↓1**	**↑26 ↓3**	**↑17 ↓10**	*↑12 ↓17*	**↑177 ↓33**
T_3	(35)	*↑0 ↓1*	**↑18 ↓1**	**↑29 ↓0**	**↑33 ↓0**	**↑30 ↓1**	**↑24 ↓3**	**↑19 ↓8**	**↑17 ↓13**	**↑170 ↓27**
T_4	(35)	*↑1 ↓3*	**↑28 ↓1**	**↑35 ↓0**	**↑33 ↓0**	**↑32 ↓0**	**↑24 ↓4**	**↑20 ↓10**	↑16 ↓16	**↑189 ↓34**
T_5	(35)	*↑0 ↓2*	**↑21 ↓1**	**↑27 ↓0**	**↑31 ↓2**	**↑29 ↓3**	**↑23 ↓5**	**↑17 ↓9**	**↑14 ↓12**	**↑162 ↓34**
T_6	(35)	*↑2 ↓3*	**↑17 ↓0**	**↑29 ↓0**	**↑31 ↓0**	**↑30 ↓1**	**↑25 ↓3**	**↑21 ↓6**	**↑15 ↓11**	**↑170 ↓24**
All	(270)	*↑3 ↓15*	**↑160 ↓7**	**↑229 ↓1**	**↑241 ↓3**	**↑233 ↓11**	**↑192 ↓29**	**↑156 ↓60**	**↑130 ↓93**	**↑1344 ↓219**

$$API = \frac{\sum_{a=1}^{n_{exp}} 100 \times PI_a}{n_{exp}} \tag{4.6}$$

with

$$PI_a = \begin{cases} \frac{H_{OE,a}(\text{DynDE})}{H_{OE,a}(\text{CDE})} - 1 \text{ if } H_{OE,a}(\text{DynDE}) < H_{OE,a}(\text{CDE}) \\ 1 - \frac{H_{OE,a}(\text{CDE})}{H_{OE,a}(\text{DynDE})} \text{ if } H_{OE,a}(\text{DynDE}) > H_{OE,a}(\text{CDE}) \end{cases} \tag{4.7}$$

where n_{exp} is the total number of experimental environments, and $H_{OE,a}$(DynDE) and $H_{OE,a}$(CDE) are the average offline errors of DynDE and CDE on experimental environment a, respectively. API is thus calculated by averaging the percentage improvement or deterioration over all experiments.

The average percentage improvement of CPE over DynDE over all experiments was found to be 10.88%. The APIs per dimension were found to be 3.12%, 10.57%, 15.67%, 13.95% and 11.05% for 5, 10, 25, 50 and 100 dimensions respectively. Larger improvements in offline error were thus found in higher dimensions. The APIs per change period were found to be -0.29%, 4.33%, 11.43%, 22.72%, 22.97%, 17.02%, 8.54% and 0.3% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The maximum improvements were thus found for change periods of 5 000 and 10 000, while small improvements were found for large change periods and a deterioration in offline error was found for the change periods of 100 function evaluations.

4.6.5.2 RMC compared to DynDE

The scalability study presented in Section 4.6.4 did not reveal any obviously different trends between DynDE and RMC. The analysis of the number of times that DynDE and RMC outperformed each other, given in Tables 4.13 and 4.14, shows that the average offline errors of DynDE and RMC differed statistically significantly in only 152 of the 2 160 experiments.

RMC outperformed DynDE in 102 cases while DynDE outperformed RMC in 50 cases. This indicates that there is very little benefit in using RMC. RMC did, however, outperform DynDE in 49 of the five dimensional experiments (i.e. 11.3% of the five dimensional

experiments), while it was only outperformed by DynDE in seven cases. This shows that while, in general, the RMC approach has a very small impact, it is still beneficial in low dimensional problems. The underlying function had a clear impact on when RMC proved useful. For example, on the spherical peak function, RMC outperformed DynDE in 15 (i.e. 31.2%) of the cases but was inferior to DynDE in only one case.

An experiment was conducted to investigate why RMC delivered improvements only in low dimensions, and why RMC is more effective when using the spherical peak function. The experiment used 100 000 random moving peak function fitness landscapes which were created for each of the numbers of dimensions from 2 to 40. For each case, the number of pairs of peaks, located within an Euclidean distance of less than the exclusion threshold of each other, was counted. For each pair of these peaks, a midpoint check was performed to determine whether the RMC algorithm would have detected multiple peaks correctly. The results of these experiments are depicted in Figures 4.32 and 4.33 for simulations on the conical and spherical peak functions respectively. Observe that the number of peaks, located within the exclusion threshold of each other, drops sharply from over 300 000 to zero between 2 and 16 dimensions. This explains why the RMC approach is only effective in low dimensional cases. The situations where RMC would have been useful did not occur in high dimensional problems. The midpoint check approach was more effective at identifying pairs of peaks on the spherical function than on the conical function in low dimensions (i.e. more cases as depicted in Figure 4.6, scenarios B and C, occur with the conical peak function). This explains why RMC yielded larger improvements in offline error on the spherical peak function than on the conical peak function.

The average percentage improvement of RMC over DynDE, over all experiments, was found to be 0.2%. The APIs per dimension were found to be 1.42%, -0.445%, 0.06%, -0.19% and 0.13% for 5, 10, 25, 50 and 100 dimensions respectively. The largest improvement was thus found in 5 dimensions. The APIs per change period were found to be -0.36%, 0.20%, 0.38%, 0.20%, 0.37%, 0.17%, 0.31% and 0.29% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The general improvement of RMC over CDE is thus minor and localised to 5 dimensions. Although RMC yielded low improvements on average, specific cases where large improvements were found to exist; for example, the experiment using the MPB's spherical peak function in five

Table 4.13: RMC vs DynDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑7 ↓0**
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑5 ↓0**
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑1 ↓1	**↑3 ↓1**
20	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑3 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
80	(2)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**
C	(6)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑1 ↓1	**↑5 ↓1**
S	(6)	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑5 ↓0**	**↑15 ↓1**
		GDBG								
F_{1a}	(6)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑4 ↓0**	**↑11 ↓0**
F_{1b}	(6)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑9 ↓0**
F_2	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**
F_3	(6)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1
F_4	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
F_5	(6)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑2 ↓0**	**↑5 ↓1**
F_6	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓3*
T_1	(7)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑6 ↓0**
T_2	(7)	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑2 ↓0**	**↑6 ↓2**
T_3	(7)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	**↑3 ↓1**
T_4	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓0**
T_5	(7)	**↑3 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑10 ↓1**
T_6	(7)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**
All	(54)	**↑3 ↓2**	**↑4 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑6 ↓1**	**↑7 ↓0**	**↑6 ↓1**	**↑13 ↓2**	**↑49 ↓7**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*
20	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
C	(6)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑2 ↓1**
S	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑1 ↓3*
F_{1b}	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
F_2	(6)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	**↑4 ↓1**
F_3	(6)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓2*	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓4*
F_4	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*
F_5	(6)	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑3 ↓1**
F_6	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*
T_1	(7)	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1
T_2	(7)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
T_3	(7)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑1 ↓1
T_4	(7)	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓4*
T_5	(7)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓1**
T_6	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓3*	*↑0 ↓1*	↑0 ↓0	*↑0 ↓4*
All	(54)	**↑1 ↓0**	**↑2 ↓1**	*↑2 ↓3*	**↑3 ↓0**	↑0 ↓0	*↑2 ↓4*	*↑1 ↓3*	*↑0 ↓1*	*↑11 ↓12*
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
20	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**
40	(2)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
C	(6)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
S	(6)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑3 ↓1**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**
F_{1b}	(6)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
F_2	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**
F_3	(6)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	*↑1 ↓3*
F_4	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑1 ↓1
F_5	(6)	↑1 ↓1	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
F_6	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	↑0 ↓0	↑0 ↓0	↑2 ↓2
T_1	(7)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**
T_2	(7)	*↑0 ↓1*	**↑2 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑3 ↓1**
T_3	(7)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	**↑1 ↓0**	↑0 ↓0	**↑3 ↓2**
T_4	(7)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑1 ↓1
T_5	(7)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	↑0 ↓0	↑0 ↓0	*↑0 ↓3*
T_6	(7)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**
All	(54)	↑2 ↓2	**↑3 ↓1**	**↑2 ↓0**	**↑3 ↓0**	**↑2 ↓1**	*↑0 ↓4*	**↑2 ↓1**	↑0 ↓0	**↑14 ↓9**

Table 4.14: RMC vs DynDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
10	(2)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
20	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*
40	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
C	(6)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓3*
S	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑1 ↓1
		GDBG								
F_{1a}	(6)	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓4*
F_{1b}	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑4 ↓0**
F_2	(6)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
F_3	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	**↑2 ↓1**
F_4	(6)	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	*↑2 ↓3*
F_5	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑1 ↓1
F_6	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
T_1	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓3*
T_2	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓3*
T_3	(7)	↑1 ↓1	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑3 ↓2**
T_4	(7)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	*↑0 ↓2*	↑0 ↓0	**↑4 ↓2**
T_5	(7)	↑0 ↓0	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑3 ↓1**
T_6	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
All	(54)	↑1 ↓1	↑2 ↓2	↑1 ↓1	↑2 ↓2	*↑1 ↓3*	**↑3 ↓0**	*↑1 ↓3*	*↑1 ↓3*	*↑12 ↓15*
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
5	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
10	(2)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑2 ↓2
20	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
40	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
C	(6)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑1 ↓2*
S	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	**↑2 ↓1**
		GDBG								
F_{1a}	(6)	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
F_{1b}	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑5 ↓0**
F_2	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**
F_3	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
F_4	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
F_5	(6)	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1
F_6	(6)	*↑0 ↓1*	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑1 ↓2*
T_1	(7)	↑0 ↓0	↑1 ↓1	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
T_2	(7)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**
T_3	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**
T_4	(7)	↑1 ↓1	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑2 ↓2
T_5	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑3 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑5 ↓0**
T_6	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
All	(54)	↑1 ↓1	*↑2 ↓4*	**↑2 ↓1**	↑0 ↓0	**↑2 ↓0**	**↑5 ↓0**	**↑2 ↓0**	**↑2 ↓1**	**↑16 ↓7**

Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑1 ↓0**	*↑0 ↓1*	**↑2 ↓0**	**↑3 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓1**
5	(10)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑6 ↓1**
10	(10)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓1*	*↑1 ↓2*	↑5 ↓5
20	(10)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**	**↑5 ↓1**
40	(10)	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
80	(10)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑1 ↓2*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	*↑2 ↓3*
C	(6)	↑0 ↓0	*↑0 ↓3*	**↑3 ↓1**	**↑2 ↓1**	**↑2 ↓1**	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	↑8 ↓8
S	(6)	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑2 ↓0**	**↑3 ↓0**	**↑5 ↓2**	**↑22 ↓4**
		GDBG								
F_{1a}	(30)	**↑2 ↓1**	**↑4 ↓2**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑3 ↓1**	*↑1 ↓2*	**↑4 ↓2**	**↑16 ↓8**
F_{1b}	(30)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑3 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑19 ↓0**
F_2	(30)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑3 ↓0**	*↑0 ↓1*	**↑2 ↓1**	**↑3 ↓0**	**↑1 ↓0**	**↑12 ↓2**
F_3	(30)	*↑0 ↓1*	**↑2 ↓1**	*↑1 ↓2*	↑1 ↓1	↑0 ↓0	*↑1 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑5 ↓9*
F_4	(30)	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	**↑2 ↓1**	*↑0 ↓2*	↑0 ↓0	↑5 ↓5
F_5	(30)	*↑1 ↓2*	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑3 ↓0**	**↑2 ↓1**	**↑2 ↓0**	**↑12 ↓5**
F_6	(30)	*↑0 ↓1*	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓9*
T_1	(35)	↑0 ↓0	**↑4 ↓2**	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	**↑1 ↓0**	**↑2 ↓1**	**↑2 ↓1**	**↑11 ↓5**
T_2	(35)	*↑0 ↓2*	**↑3 ↓1**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓1**	**↑1 ↓0**	*↑0 ↓1*	**↑3 ↓1**	**↑12 ↓7**
T_3	(35)	**↑2 ↓1**	↑0 ↓0	*↑1 ↓2*	**↑1 ↓0**	↑0 ↓0	↑2 ↓2	**↑3 ↓0**	**↑2 ↓1**	**↑11 ↓6**
T_4	(35)	↑1 ↓1	**↑2 ↓1**	**↑3 ↓1**	**↑1 ↓0**	**↑1 ↓0**	**↑4 ↓1**	*↑0 ↓4*	*↑0 ↓1*	**↑12 ↓9**
T_5	(35)	**↑3 ↓1**	**↑3 ↓1**	*↑0 ↓1*	**↑2 ↓1**	**↑1 ↓0**	**↑5 ↓2**	**↑4 ↓0**	**↑2 ↓0**	**↑20 ↓6**
T_6	(35)	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	*↑2 ↓3*	*↑0 ↓1*	**↑1 ↓0**	**↑6 ↓5**
All	(270)	**↑8 ↓6**	**↑13 ↓8**	**↑13 ↓5**	**↑12 ↓3**	**↑11 ↓5**	**↑17 ↓8**	**↑12 ↓8**	**↑16 ↓7**	**↑102 ↓50**

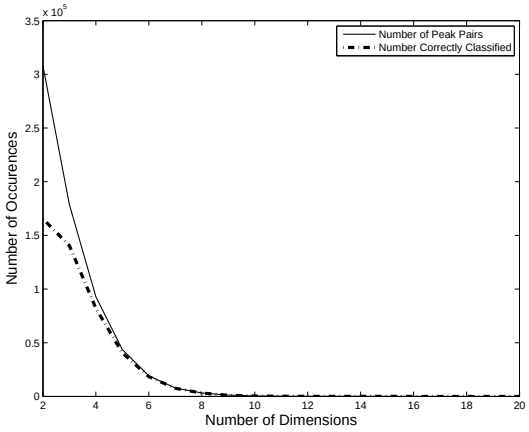

Figure 4.32: Number of peak pairs that fall within the exclusion threshold and number of correct classifications by RMC per dimension on the conical peak function.

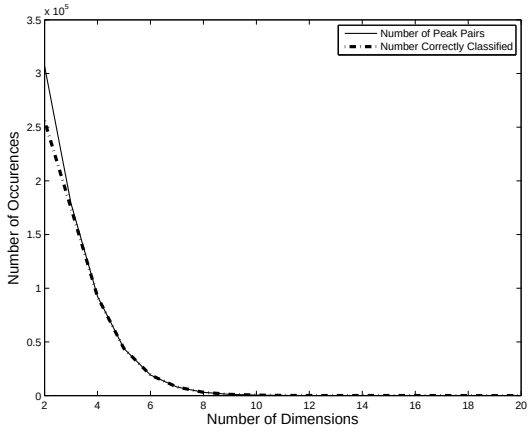

Figure 4.33: Number of peak pairs that fall within the exclusion threshold and number of correct classifications by RMC per dimension on the spherical peak function.

dimensions with a change period of 50 000 yielded a 54.61% improvement over DynDE.

The benefits of using RMC are less pronounced and wide-ranging than that of CPE. However, RMC does constitute an improvement over DynDE in a specific sub-set of dynamic environments.

4.6.5.3 CDE compared to DynDE

The analysis of the CDE results, in comparison with DynDE, is given in Tables 4.15 and 4.16. CDE performed significantly better than DynDE in 1 351 experiments and worse than DynDE in 208 cases. CDE was thus better than DynDE in 62.5% of all experiments and DynDE was better than CDE in only 9.6% of the experiments. This performance is considerably better than that of RMC, and slightly better than that of CPE. The most noticeable difference between CPE and CDE is in five dimensional experiments, in which the number of cases where CDE is better than DynDE is 17 more than those of CPE, and the number of cases where CDE is worse than DynDE is 10 less than CPE. The higher dimensional experiments yielded roughly equivalent performance in comparison to DynDE for CPE and CDE. RMC's positive contribution was shown to be localised to low dimensions, and the benefits of incorporating RMC into CPE are also limited to the five dimensional experiments.

CPE was shown to be more effective when using a low change period than when using a high change period. This trend is continued in CDE, where 86% of the cases where DynDE performed the best, occurred when a change period of 25 000 or higher was used.

The average percentage improvement of CDE over DynDE over all experiments was found to be 11.09%. The APIs per dimension were found to be 4.30%, 10.69%, 15.51%, 13.96% and 10.97% for 5, 10, 25, 50 and 100 dimensions respectively. Larger improvements in offline error were thus found in higher dimensions, as was the case with CPE. The APIs per change period, were found to be -0.22%, 4.56%, 11.51%, 22.66%, 22.68%, 17.45%, 9.02% and 1.04% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The maximum improvements were thus found for change periods of 5 000 and 10 000, while small improvements were found for large change periods and a deterioration in offline error was found for the change period of 100.

4.6.5.4 Summary for Research Question 3

CPE, RMC and CDE were all found to outperform DynDE more often than being outperformed by DynDE. RMC differed from DynDE in only a small number of instances, but was shown to be effective on isolated functions. CPE generally outperformed DynDE, with cases where it was inferior to DynDE being concentrated in the low dimensions with a high change period.

CDE was better than DynDE in more cases than both CPE and RMC. The scalability study conducted in Section 4.6.4 found that CDE scaled very similar to CPE, and is especially superior to DynDE in high dimensional, low change period and severely changing environments. Section 2.5.2 argued that the set of feasible DOPs for any particular function is bounded by values of the change severity and change period. CDE thus increases the set of feasible DOP by being more effective on high dimensional environments in which changes occur frequently.

An analysis was performed to determine whether CDE results in statistically significant improvements over its sub-components, CPE and RMC. Full results of these comparisons are given in Appendix B. CDE performed better than CPE in 86 of the 2160 experiments and performed worse in 47. The majority of the cases where CDE outperformed CPE (45 cases) was in five dimensional experiments due to the influence of the RMC component.

Table 4.15: CDE vs DynDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑1 ↓0	↑2 ↓0	↑1 ↓0	↑0 ↓0	↑0 ↓0	↑7 ↓0
5	(2)	↑0 ↓0	↑0 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑12 ↓0
10	(2)	↑0 ↓0	↑0 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑1 ↓0	↑11 ↓0
20	(2)	↑0 ↓0	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑1 ↓0	↑10 ↓0
40	(2)	↑0 ↓0	↑0 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓1	↑1 ↓1	↑6 ↓2
80	(2)	↑0 ↓0	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓1	↑7 ↓1
C	(6)	↑0 ↓0	↑1 ↓0	↑3 ↓0	↑5 ↓0	↑4 ↓0	↑4 ↓0	↑3 ↓1	↑1 ↓2	↑21 ↓3
S	(6)	↑0 ↓0	↑0 ↓0	↑6 ↓0	↑5 ↓0	↑6 ↓0	↑5 ↓0	↑5 ↓0	↑5 ↓0	↑32 ↓0
		GDBG								
F_{1a}	(6)	↑0 ↓1	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑0 ↓3	↑0 ↓6	↑0 ↓6	↑24 ↓16
F_{1b}	(6)	↑1 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑0 ↓4	↑0 ↓6	↑0 ↓6	↑24 ↓16
F_2	(6)	↑0 ↓0	↑4 ↓0	↑6 ↓0	↑5 ↓1	↑3 ↓3	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑18 ↓22
F_3	(6)	↑0 ↓0	↑3 ↓0	↑4 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓1	↑2 ↓0	↑2 ↓0	↑11 ↓1
F_4	(6)	↑0 ↓0	↑2 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓1	↑0 ↓6	↑0 ↓6	↑24 ↓13
F_5	(6)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓2	↑1 ↓4	↑0 ↓5	↑29 ↓11
F_6	(6)	↑0 ↓0	↑3 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓1	↑4 ↓2	↑0 ↓2	↑29 ↓5
T_1	(7)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓1	↑0 ↓6	↑1 ↓6	↑1 ↓6	↑24 ↓19
T_2	(7)	↑0 ↓1	↑5 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑3 ↓1	↑1 ↓5	↑0 ↓5	↑28 ↓12
T_3	(7)	↑0 ↓0	↑5 ↓0	↑7 ↓0	↑6 ↓0	↑5 ↓1	↑3 ↓2	↑1 ↓4	↑0 ↓5	↑27 ↓12
T_4	(7)	↑0 ↓0	↑6 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑0 ↓4	↑1 ↓6	↑0 ↓6	↑26 ↓16
T_5	(7)	↑1 ↓0	↑3 ↓0	↑6 ↓0	↑5 ↓1	↑5 ↓1	↑3 ↓2	↑2 ↓4	↑0 ↓4	↑25 ↓12
T_6	(7)	↑0 ↓0	↑5 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑3 ↓3	↑1 ↓5	↑1 ↓5	↑29 ↓13
All	(54)	↑1 ↓1	↑31 ↓0	↑49 ↓0	↑45 ↓1	↑42 ↓3	↑21 ↓18	↑15 ↓31	↑8 ↓33	↑212 ↓87
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓0	↑0 ↓2	↑3 ↓7
5	(2)	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑0 ↓0	↑7 ↓0
10	(2)	↑0 ↓0	↑1 ↓0	↑1 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑12 ↓0
20	(2)	↑0 ↓0	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑11 ↓0
40	(2)	↑0 ↓0	↑0 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑12 ↓0
80	(2)	↑0 ↓0	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑11 ↓0
C	(6)	↑0 ↓0	↑3 ↓0	↑5 ↓0	↑6 ↓0	↑5 ↓0	↑5 ↓0	↑5 ↓0	↑4 ↓1	↑33 ↓1
S	(6)	↑0 ↓0	↑0 ↓1	↑3 ↓1	↑4 ↓1	↑4 ↓1	↑4 ↓1	↑4 ↓0	↑4 ↓1	↑23 ↓6
		GDBG								
F_{1a}	(6)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑0 ↓4	↑0 ↓6	↑29 ↓10
F_{1b}	(6)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑3 ↓0	↑0 ↓5	↑0 ↓5	↑27 ↓10
F_2	(6)	↑1 ↓1	↑3 ↓0	↑5 ↓0	↑6 ↓0	↑3 ↓1	↑3 ↓3	↑0 ↓4	↑0 ↓6	↑21 ↓15
F_3	(6)	↑0 ↓0	↑3 ↓0	↑5 ↓0	↑3 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1	↑12 ↓1
F_4	(6)	↑0 ↓0	↑2 ↓0	↑5 ↓0	↑6 ↓0	↑4 ↓1	↑3 ↓2	↑2 ↓4	↑0 ↓6	↑22 ↓13
F_5	(6)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓2	↑40 ↓2
F_6	(6)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓1	↑40 ↓1
T_1	(7)	↑0 ↓0	↑6 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑2 ↓4	↑0 ↓6	↑32 ↓10
T_2	(7)	↑0 ↓0	↑6 ↓0	↑7 ↓0	↑7 ↓0	↑4 ↓0	↑4 ↓1	↑2 ↓4	↑2 ↓4	↑32 ↓9
T_3	(7)	↑0 ↓1	↑5 ↓0	↑7 ↓0	↑7 ↓0	↑5 ↓0	↑3 ↓2	↑2 ↓2	↑2 ↓3	↑31 ↓8
T_4	(7)	↑0 ↓0	↑7 ↓0	↑7 ↓0	↑7 ↓0	↑6 ↓0	↑4 ↓0	↑3 ↓2	↑0 ↓5	↑34 ↓7
T_5	(7)	↑1 ↓0	↑4 ↓0	↑4 ↓0	↑6 ↓0	↑4 ↓2	↑4 ↓2	↑2 ↓3	↑3 ↓4	↑28 ↓11
T_6	(7)	↑0 ↓0	↑4 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑3 ↓2	↑2 ↓5	↑34 ↓7
All	(54)	↑1 ↓1	↑35 ↓1	↑47 ↓1	↑49 ↓1	↑40 ↓3	↑35 ↓6	↑23 ↓17	↑17 ↓29	↑247 ↓59
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑0 ↓0	↑0 ↓0	↑5 ↓0
5	(2)	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑1 ↓0	↑1 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑11 ↓0
10	(2)	↑0 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑1 ↓0	↑2 ↓0	↑1 ↓0	↑1 ↓0	↑8 ↓0
20	(2)	↑0 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑14 ↓0
40	(2)	↑0 ↓0	↑1 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑13 ↓0
80	(2)	↑0 ↓0	↑0 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑2 ↓0	↑12 ↓0
C	(6)	↑0 ↓0	↑4 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑4 ↓0	↑37 ↓0
S	(6)	↑0 ↓0	↑2 ↓0	↑4 ↓0	↑3 ↓0	↑3 ↓0	↑5 ↓0	↑4 ↓0	↑5 ↓0	↑26 ↓0
		GDBG								
F_{1a}	(6)	↑0 ↓0	↑5 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑2 ↓2	↑37 ↓2
F_{1b}	(6)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑1 ↓2	↑36 ↓2
F_2	(6)	↑0 ↓0	↑4 ↓0	↑4 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓0	↑3 ↓1	↑33 ↓1
F_3	(6)	↑0 ↓0	↑0 ↓0	↑6 ↓0	↑5 ↓0	↑2 ↓1	↑0 ↓1	↑0 ↓5	↑0 ↓4	↑13 ↓11
F_4	(6)	↑1 ↓1	↑3 ↓0	↑2 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑4 ↓0	↑3 ↓1	↑31 ↓2
F_5	(6)	↑0 ↓1	↑5 ↓0	↑5 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑39 ↓1
F_6	(6)	↑0 ↓0	↑4 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑6 ↓0	↑39 ↓0
T_1	(7)	↑0 ↓2	↑6 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓1	↑6 ↓1	↑6 ↓1	↑5 ↓1	↑42 ↓6
T_2	(7)	↑0 ↓0	↑5 ↓0	↑6 ↓0	↑7 ↓0	↑7 ↓0	↑6 ↓0	↑4 ↓1	↑2 ↓3	↑37 ↓4
T_3	(7)	↑0 ↓0	↑3 ↓0	↑4 ↓0	↑7 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓1	↑2 ↓1	↑35 ↓2
T_4	(7)	↑0 ↓0	↑6 ↓0	↑7 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑6 ↓1	↑4 ↓3	↑42 ↓4
T_5	(7)	↑0 ↓0	↑4 ↓0	↑5 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑3 ↓1	↑4 ↓0	↑35 ↓1
T_6	(7)	↑1 ↓0	↑3 ↓0	↑6 ↓0	↑7 ↓0	↑6 ↓0	↑6 ↓0	↑5 ↓0	↑3 ↓2	↑37 ↓2
All	(54)	↑1 ↓2	↑33 ↓0	↑45 ↓0	↑50 ↓0	↑47 ↓1	↑47 ↓1	↑39 ↓5	↑29 ↓10	↑291 ↓19

Table 4.16: CDE vs DynDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑12 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
20	(2)	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓1**
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
80	(2)	**↑1 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
C	(6)	**↑1 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓0**	**↑37 ↓0**
S	(6)	*↑0 ↓1*	**↑2 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓2**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑38 ↓0**
F_{1b}	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑40 ↓1**
F_2	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑39 ↓2**
F_3	(6)	*↑0 ↓1*	↑1 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓0**	↑0 ↓0	*↑0 ↓4*	*↑0 ↓4*	**↑16 ↓9**
F_4	(6)	↑0 ↓0	**↑5 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑35 ↓1**
F_5	(6)	**↑1 ↓0**	↑1 ↓0	**↑2 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑24 ↓0**
F_6	(6)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑35 ↓0**
T_1	(7)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑38 ↓2**
T_2	(7)	**↑1 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓3**	**↑38 ↓3**
T_3	(7)	↑0 ↓0	**↑2 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑5 ↓1**	**↑38 ↓2**
T_4	(7)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑41 ↓3**
T_5	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑38 ↓2**
T_6	(7)	*↑0 ↓1*	**↑3 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑34 ↓1**
All	(54)	*↑2 ↓4*	**↑31 ↓1**	**↑43 ↓0**	**↑52 ↓0**	**↑51 ↓0**	**↑46 ↓0**	**↑40 ↓4**	**↑36 ↓6**	**↑301 ↓15**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑10 ↓5**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	↑1 ↓1	*↑0 ↓1*	**↑9 ↓3**
10	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑2 ↓0**	**↑10 ↓3**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑12 ↓1**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
C	(6)	↑0 ↓0	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑36 ↓0**
S	(6)	↑1 ↓1	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	↑3 ↓3	**↑4 ↓2**	*↑2 ↓4*	**↑4 ↓2**	**↑31 ↓13**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
F_{1b}	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
F_2	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑39 ↓1**
F_3	(6)	↑0 ↓0	*↑0 ↓2*	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	*↑0 ↓1*	*↑0 ↓5*	**↑19 ↓8**
F_4	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑41 ↓2**
F_5	(6)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑20 ↓0**
F_6	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑34 ↓4**
T_1	(7)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑39 ↓2**
T_2	(7)	↑0 ↓0	**↑6 ↓1**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓3**	**↑44 ↓4**
T_3	(7)	↑0 ↓0	**↑3 ↓1**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓2**	**↑38 ↓3**
T_4	(7)	*↑0 ↓2*	**↑4 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑43 ↓3**
T_5	(7)	**↑1 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑5 ↓2**	**↑37 ↓3**
T_6	(7)	↑0 ↓0	**↑2 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑32 ↓0**
All	(54)	*↑2 ↓3*	**↑32 ↓2**	**↑45 ↓0**	**↑50 ↓1**	**↑47 ↓3**	**↑46 ↓2**	**↑39 ↓6**	**↑39 ↓11**	**↑300 ↓28**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑1 ↓0**	**↑7 ↓1**	**↑8 ↓1**	**↑6 ↓2**	**↑6 ↓2**	**↑5 ↓2**	**↑2 ↓1**	*↑2 ↓3*	**↑37 ↓12**
5	(10)	↑0 ↓0	**↑4 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑7 ↓1**	**↑9 ↓0**	**↑8 ↓1**	**↑5 ↓1**	**↑51 ↓3**
10	(10)	*↑0 ↓1*	**↑5 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑7 ↓0**	**↑57 ↓1**
20	(10)	*↑0 ↓1*	**↑3 ↓0**	**↑8 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓0**	**↑57 ↓4**
40	(10)	↑0 ↓0	**↑2 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑8 ↓2**	**↑9 ↓1**	**↑54 ↓3**
80	(10)	**↑1 ↓0**	**↑2 ↓1**	**↑8 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓1**	**↑57 ↓2**
C	(6)	**↑1 ↓0**	**↑13 ↓0**	**↑26 ↓0**	**↑29 ↓0**	**↑27 ↓0**	**↑27 ↓0**	**↑24 ↓1**	**↑17 ↓3**	**↑164 ↓4**
S	(6)	*↑1 ↓2*	**↑10 ↓2**	**↑24 ↓1**	**↑23 ↓2**	**↑22 ↓4**	**↑24 ↓3**	**↑21 ↓4**	**↑24 ↓3**	**↑149 ↓21**
		GDBG								
F_{1a}	(30)	*↑0 ↓1*	**↑26 ↓0**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑23 ↓3**	**↑18 ↓10**	*↑13 ↓14*	**↑168 ↓28**
F_{1b}	(30)	↑1 ↓1	**↑27 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑21 ↓4**	**↑17 ↓11**	*↑12 ↓13*	**↑167 ↓29**
F_2	(30)	*↑1 ↓2*	**↑21 ↓0**	**↑26 ↓0**	**↑29 ↓1**	**↑24 ↓4**	**↑21 ↓9**	**↑15 ↓10**	*↑13 ↓15*	**↑150 ↓41**
F_3	(30)	*↑0 ↓1*	**↑7 ↓2**	**↑24 ↓0**	**↑20 ↓0**	**↑11 ↓1**	**↑4 ↓2**	*↑2 ↓10*	*↑3 ↓14*	**↑71 ↓30**
F_4	(30)	*↑1 ↓2*	**↑18 ↓0**	**↑23 ↓0**	**↑30 ↓0**	**↑28 ↓1**	**↑25 ↓3**	**↑16 ↓10**	*↑12 ↓15*	**↑153 ↓31**
F_5	(30)	**↑2 ↓1**	**↑19 ↓0**	**↑21 ↓0**	**↑25 ↓0**	**↑26 ↓0**	**↑24 ↓2**	**↑19 ↓4**	**↑16 ↓7**	**↑152 ↓14**
F_6	(30)	*↑0 ↓1*	**↑21 ↓0**	**↑27 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑26 ↓1**	**↑24 ↓3**	**↑19 ↓5**	**↑177 ↓10**
T_1	(35)	*↑0 ↓2*	**↑28 ↓0**	**↑32 ↓0**	**↑30 ↓0**	**↑28 ↓2**	**↑22 ↓7**	**↑19 ↓13**	**↑16 ↓15**	**↑175 ↓39**
T_2	(35)	↑1 ↓1	**↑26 ↓1**	**↑32 ↓0**	**↑34 ↓0**	**↑31 ↓0**	**↑26 ↓2**	**↑17 ↓10**	*↑12 ↓18*	**↑179 ↓32**
T_3	(35)	*↑0 ↓1*	**↑18 ↓1**	**↑28 ↓0**	**↑34 ↓0**	**↑31 ↓1**	**↑24 ↓4**	**↑20 ↓8**	**↑14 ↓12**	**↑169 ↓27**
T_4	(35)	*↑0 ↓3*	**↑28 ↓0**	**↑33 ↓0**	**↑34 ↓0**	**↑32 ↓0**	**↑23 ↓4**	**↑21 ↓10**	*↑15 ↓16*	**↑186 ↓33**
T_5	(35)	**↑3 ↓1**	**↑22 ↓0**	**↑26 ↓0**	**↑31 ↓1**	**↑27 ↓3**	**↑23 ↓4**	**↑15 ↓10**	**↑16 ↓10**	**↑163 ↓29**
T_6	(35)	↑1 ↓1	**↑17 ↓0**	**↑28 ↓0**	**↑31 ↓0**	**↑29 ↓0**	**↑26 ↓3**	**↑19 ↓7**	**↑15 ↓12**	**↑166 ↓23**
All	(270)	*↑7 ↓11*	**↑162 ↓4**	**↑229 ↓1**	**↑246 ↓3**	**↑227 ↓10**	**↑195 ↓27**	**↑156 ↓63**	**↑129 ↓89**	**↑1351 ↓208**

The results indicate that CDE is thus a slightly better algorithm than CPE. CDE performed better than RMC in 1 342 cases and performed worse in only 209 experiments. CDE is thus clearly a better algorithm than RMC.

4.6.6 Research Question 4

How does the convergence behaviour of CDE differ from that of DynDE?

The analysis under the previous research question found that CDE is a more effective DOP algorithm than DynDE. This research question investigates the difference between DynDE and CDE in terms of diversity and current error with the aim of explaining trends observed in the previous sections.

Section 3.4.1.5 showed that one of the reasons why DynDE is more effective than normal DE on DOPs is that DynDE maintains higher diversity during the optimisation process. The average diversity (calculated using equation (2.7)) of CDE was measured on the conical peak function of the MPB in five dimensions with a change period of 5 000 function evaluations. The optimisation process was allowed to continue for 500 000 function evaluations and was repeated 30 times. The average diversity per generation over all repeats was found to be 0.2624 for CDE, compared to the values of 0.2661 for DynDE, and 0.0017 for DE. The diversity of CDE is thus virtually identical to that of DynDE.

The diversity measure given in equation (2.7) gives the diversity of all individuals used by the algorithm. This measure will appear artificially high because sub-populations converge to optima that are uniformly distributed around the fitness landscape. The measure also gives no information regarding the diversity within sub-populations. Equation (2.7) can be adapted to give the average diversity per sub-population, D_{AP}, as shown in equation (4.8).

$$D_{AP} = \frac{\sum_{i=1}^{n_k} \frac{\sum_{i=1}^{n_{I,k}} \|\vec{d} - \vec{x}_{i,k}\|_2}{n_{I,k}}}{n_k L} \tag{4.8}$$

D_{AP} can be used to study the diversity within sub-populations. Figure 4.34 gives the offline error, current error, diversity and average diversity per population for DynDE and CDE on the conical peak function of the MPB in five dimensions with a change period of 5 000 function evaluations. The first 10 changes in the environment are depicted.

The figure shows that, while DynDE and CDE had very similar normal diversity profiles, the average diversity per sub-population differed drastically between DynDE and CDE. DynDE's value for D_{AP} decreased gradually over many function evaluations. This means that DynDE's sub-populations slowly converged to optima in the fitness landscape.

CDE's value for D_{AP}, in contrast to DynDE, rapidly decreased to a relatively low value. CDE thus converged to optima faster than DynDE, due to the competitive population evaluation approach which allows sub-populations to evolve in sequence and thus discover optima faster. The RMC component of CDE also reduced the average sub-population diversity as it prevents the unnecessary reinitialisation of sub-populations which would have dispersed individuals randomly over the search space. Figure 4.34 shows that CDE's current error reduced faster than that of DynDE after changes in the environment. This is due to CDE's sub-populations converging to optima earlier than DynDE's sub-populations.

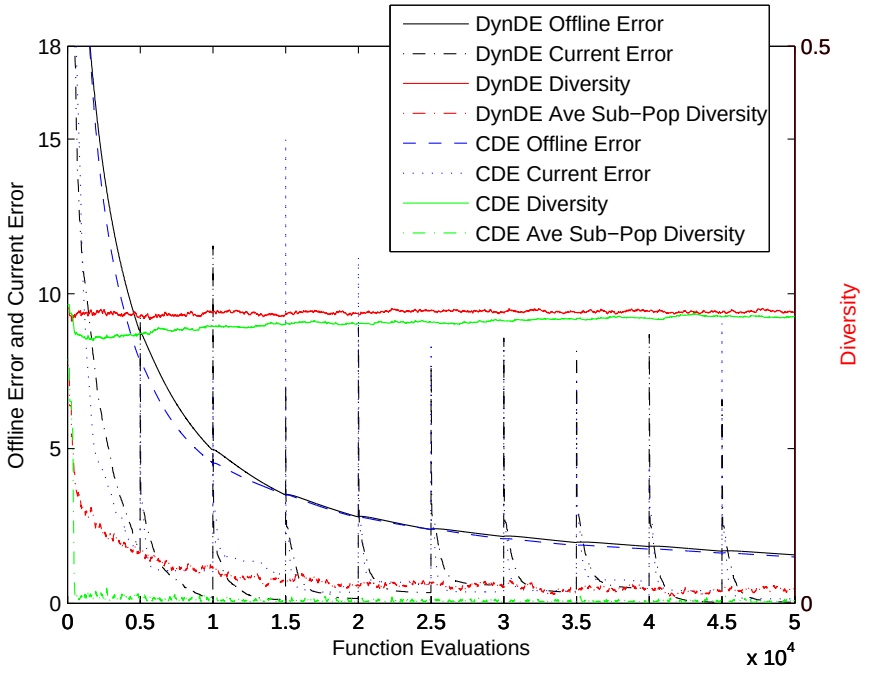

Figure 4.34: Diversity, current error, offline error and average sub-population diversity of DynDE and CDE on the MPB, Scenario 2

The previous sections found that CDE is more effective than DynDE, especially when a low change period and high dimensions are present in the environment. The convergence behaviour of DynDE and CDE is investigated here on environments with low change

period - low dimensions, low change period - high dimensions, high change period - low dimensions, and high change period - high dimensions.

Figure 4.35 gives the offline error, current error, diversity and average diversity per population for DynDE and CDE on the conical peak function of the MPB in five dimensions with a change period of 1 000 function evaluations. The figure confirms that the average diversity per sub-population of CDE droped noticeably faster than that of DynDE. This resulted in faster reductions in CDE current error after changes in the environment, and current errors that were consistently lower than those of DynDE. CDE thus not only found optima faster, but also achieved lower current errors than DynDE in the presence of frequent changes.

Figure 4.36 gives the offline error, current error, diversity and average diversity per population for DynDE and CDE on the conical peak function of the MPB in 100 dimensions with a change period of 1 000 function evaluations. The average diversity per sub-population of CDE was once again lower than that of DynDE. The high number of dimensions resulted in considerably larger errors than were found in the five dimensional cases. The magnitude of the difference between the current errors of DynDE and CDE was greater than what it was on the low dimensional experiment (compare Figure 4.36 to Figure 4.35). The allocation of function evaluations based on performance was thus especially beneficial in high dimensions, which explains why CDE generally achieved better offline errors than DynDE in high dimensional DOPs.

The offline error, current error, diversity and average diversity per population for DynDE and CDE on the conical peak function of the MPB in five dimensions with a change period of 100 000 function evaluations, is given in Figure 4.37. The average diversity per sub-population of CDE reduced faster than that of DynDE. However, the large number of function evaluations between changes allowed DynDE's sub-populations more function evaluations to converge. The result was that the average diversity per sub-population of DynDE became much closer to that of CDE than what it was in Figure 4.35. The high change period allowed both algorithms enough function evaluations to achieve low offline errors.

Figure 4.38 gives an enlargement of the area around the third change in the environment. CDE's current error decreased slightly faster than that of DynDE after the change,

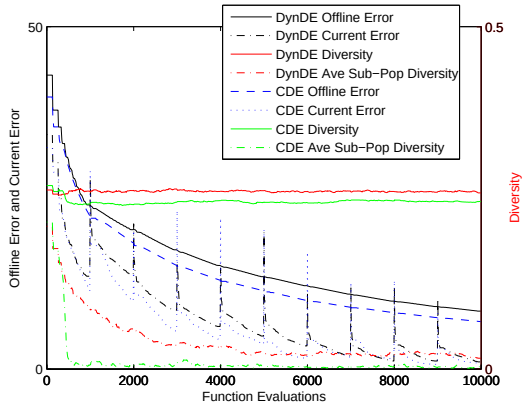

Figure 4.35: DynDE and CDE on the MPB with a conical peak function in five dimensions using a change period of 1 000

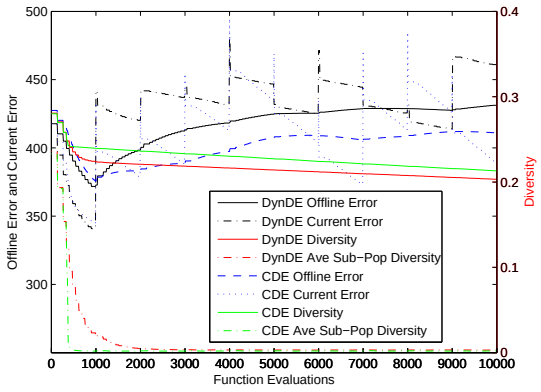

Figure 4.36: DynDE and CDE on the MPB with a conical peak function in 100 dimensions using a change period of 1 000

but DynDE's current error eventually reached a lower value than that of CDE. The large number of function evaluations between changes made the initial fast decrease in current error of CDE negligible, and DynDE thus yielded a lower offline error on this environment.

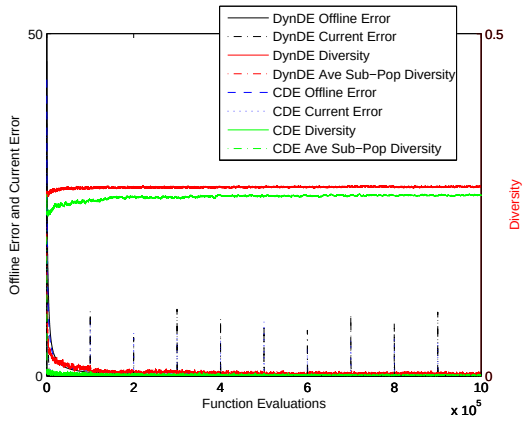

Figure 4.37: DynDE and CDE on the MPB with a conical peak function in five dimensions using a change period of 100 000

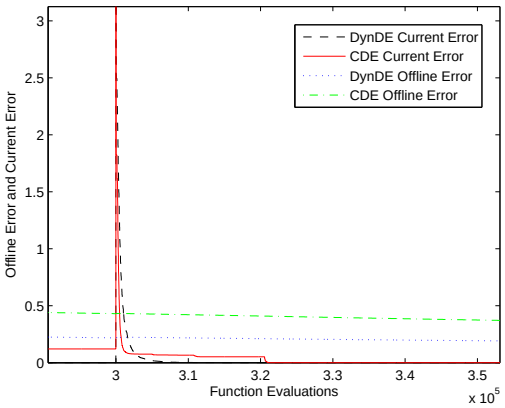

Figure 4.38: DynDE and CDE on the MPB with a conical peak function in five dimensions using a change period of 100 000, enlarged

Figure 4.39 gives the offline error, current error, diversity and average diversity per population for DynDE and CDE on the conical peak function of the MPB in 100 dimensions with a change period of 100 000 function evaluations. The average diversity per sub-

population of DynDE and CDE was, once again, very similar due to the large number of function evaluations available between changes in the environment. Note that the normal diversity profile of both algorithms initially decreased as the sub-populations frequently converged to the same optima. The normal diversity and the average sub-population diversity then increased at about 60 000 function evaluations as sub-populations were reinitialised due to exclusion. Note that much lower current and offline errors were achieved by both algorithms than were found when using a low change period in conjunction with a high number of dimensions (compare Figure 4.39 to Figure 4.36).

Figure 4.40 gives an enlargement of the area around the second change in the environment. DynDE's current error typically took a large number of function evaluations to reach its lowest value, while CDE's current error decreased faster after a change in the environment. CDE consequently yielded lower offline errors on high dimensional problems with a high change period.

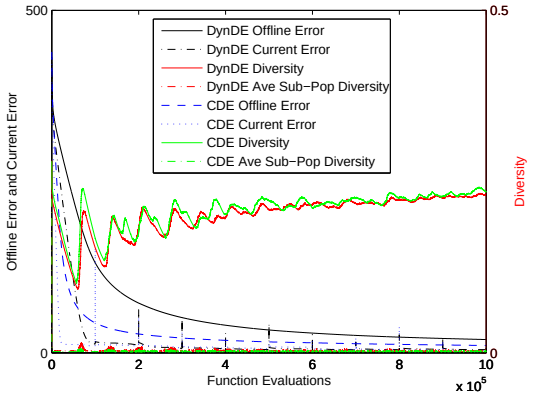

Figure 4.39: DynDE and CDE on the MPB with a conical peak function in 100 dimensions using a change period of 100 000

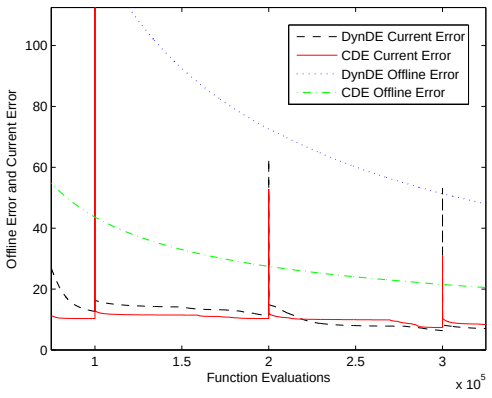

Figure 4.40: DynDE and CDE on the MPB with a conical peak function in 100 dimensions using a change period of 100 000, enlarged

4.6.7 Research Question 5

How does the average lowest error found just before changes in the environment differ between DynDE and CDE?

The components used to form CDE have been shown to produce a lower offline error than normal DynDE. However, CDE could, potentially negatively, affect the lowest error

found immediately before changes in the environment occur. A reduction in offline error does not necessarily imply that the lowest error, found before a change in the environment occurs, is also reduced. For example, the CPE component could allocate too many function evaluations to a sub-population that is positioned on a sub-optimal peak, to the detriment of the sub-population that is positioned on the global optimum. This research question investigates the possibility that CDE merely exploits the offline error performance measure and is not effective when the errors immediately before changes are considered.

The second performance measurement discussed in Section 2.5.5, H_{AEBI}, calculates the average error of the best individuals found just before changes in the environment. This performance measure was not used as the main performance measure in this study, because it only considers the performance immediately before changes in the environment, and ignores the performance during periods between changes. This performance measure is, however, ideal for answering this research question.

The same set of experiments, used to compare the performance of DynDE and CDE in Section 4.6.5, was repeated using the H_{AEBI} performance measure. A total of 2 160 experiments were thus conducted for each algorithm. The same analysis that was performed in Section 4.6.5, i.e. a count of the number of instances that each algorithm was statistically significantly better than the other, was performed. The results of this analysis are given in Tables 4.17 and 4.18.

CDE performed significantly better than DynDE in just under half of the experiments (a total of 1 068 cases). DynDE performed better than CDE in 470 cases. The general trend that can be observed is that DynDE performed better than CDE when a high change period was used. The analysis of the five dimensional experiments shows that CDE performed better, more often, than DynDE only in the experiments that used a change period of 500 or 1 000. As the number of dimensions was increased, CDE performed comparatively better then DynDE. CDE performed better more often than DynDE in 100 dimensions for all settings of change period except in the experiments that used a change period of 100 000.

These results are consistent with the comparisons done in the previous section in terms of the current errors of CDE and DynDE. The low dimensional - low change period experiment, shown in Figure 4.35, found that CDE's current error was typically lower

than that of DynDE just before changes occur in the environment. H_{AEBI} only considers the error just before a change, and CDE consequently performed better than DynDE in terms of H_{AEBI} for this case. The low dimensional - high change period experiment, shown in Figure 4.37, found that CDE, typically, had a higher current error than DynDE immediately before a change. The H_{AEBI} performance measure thus rates CDE lower than DynDE when low dimensions are used in conjunction with a high change period.

The analysis presented in this section suggests that CDE is inferior to DynDE in terms of the H_{AEBI} performance measure in the presence of large change periods. The competitive evaluation approach is aimed at discovering optima early and is consequently not beneficial in problems where only the error immediately before infrequent changes is important. However, CDE did perform better than DynDE more than twice as often as DynDE performed better than CDE, over all experiments. The cases where CDE was better than DynDE where mostly in low change period - high dimensional problems. The algorithms presented in this chapter thus improved the offline error of DynDE without completely deteriorating the error immediately before changes in the environment.

4.7 Comparison to Other Approaches

This chapter showed that using RMC and CPE yields better results than DynDE alone. This section compares the combined CDE algorithm to other algorithms in the literature. Section 4.7.1 compares the performance of CDE to that of *jDE* (refer to Section 3.4.2). Section 4.7 compares CDE to the published results from other algorithms.

4.7.1 CDE Compared to jDE

jDE, developed by Brest *et al.* [2009], was the winning algorithm of the CEC2009 competition on dynamic optimisation, and is currently one of the state-of-the-art algorithms for optimisation in dynamic environments. *jDE* was selected for a detailed comparison to CDE because of its good performance, and because it is also based on DE.

The *jDE* algorithm was implemented by the author in order to compare its performance to the algorithms presented in this thesis. The *jDE* algorithm was executed on the standard set of experiments for all the combinations of change period and number

Table 4.17: CDE vs DynDE H_{AEBI} performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	*↑0 ↓1*	↑0 ↓0	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑6 ↓3**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑8 ↓1**
10	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	*↑0 ↓2*	↑1 ↓1	**↑6 ↓4**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	**↑7 ↓4**
40	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	**↑6 ↓4**
80	(2)	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	↑5 ↓5
C	(6)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑3 ↓1**	*↑0 ↓3*	*↑0 ↓4*	*↑1 ↓4*	*↑0 ↓4*	*↑12 ↓16*
S	(6)	*↑0 ↓1*	*↑0 ↓1*	**↑5 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑3 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑26 ↓5**
		GDBG								
F_{1a}	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑12 ↓29*
F_{1b}	(6)	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑13 ↓30*
F_2	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑11 ↓30*
F_3	(6)	↑0 ↓0	**↑5 ↓0**	**↑3 ↓0**	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓2**
F_4	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓1**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑16 ↓24*
F_5	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑16 ↓27*
F_6	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑2 ↓1**	*↑0 ↓1*	*↑0 ↓2*	*↑1 ↓3*	*↑0 ↓3*	**↑15 ↓10**
T_1	(7)	↑0 ↓0	**↑7 ↓0**	**↑6 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑15 ↓30*
T_2	(7)	*↑0 ↓1*	**↑7 ↓0**	**↑7 ↓0**	**↑3 ↓2**	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑17 ↓25*
T_3	(7)	*↑0 ↓1*	**↑7 ↓0**	**↑7 ↓0**	↑2 ↓2	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑16 ↓24*
T_4	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	*↑1 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑1 ↓6*	*↑0 ↓6*	*↑14 ↓27*
T_5	(7)	**↑1 ↓0**	**↑5 ↓0**	**↑6 ↓0**	*↑2 ↓4*	*↑0 ↓4*	*↑0 ↓5*	*↑1 ↓5*	*↑0 ↓5*	*↑15 ↓23*
T_6	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	↑3 ↓3	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑1 ↓5*	*↑18 ↓23*
All	(54)	*↑1 ↓3*	**↑42 ↓1**	**↑49 ↓1**	*↑19 ↓22*	*↑5 ↓33*	*↑3 ↓38*	*↑8 ↓38*	*↑6 ↓37*	*↑133 ↓173*
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓7*
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	*↑3 ↓4*
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑3 ↓2**
20	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	**↑4 ↓3**
40	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	**↑6 ↓3**
80	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	**↑6 ↓3**
C	(6)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑2 ↓0**	*↑0 ↓3*	*↑1 ↓3*	*↑0 ↓2*	**↑16 ↓8**
S	(6)	↑0 ↓0	*↑0 ↓1*	**↑3 ↓1**	**↑3 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑8 ↓14*
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑18 ↓22*
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑18 ↓23*
F_2	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	*↑2 ↓3*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑17 ↓24*
F_3	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑12 ↓1**
F_4	(6)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑3 ↓1**	**↑3 ↓2**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑15 ↓21*
F_5	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑1 ↓3*	*↑0 ↓4*	*↑0 ↓5*	**↑25 ↓12**
F_6	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑5 ↓1**	**↑37 ↓4**
T_1	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑3 ↓2**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	**↑23 ↓20**
T_2	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑4 ↓2**	*↑2 ↓3*	*↑1 ↓5*	*↑1 ↓5*	*↑1 ↓5*	**↑22 ↓20**
T_3	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑7 ↓0**	**↑4 ↓0**	↑2 ↓2	*↑1 ↓4*	*↑1 ↓5*	*↑1 ↓5*	**↑22 ↓17**
T_4	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓2**	*↑0 ↓5*	*↑0 ↓6*	*↑1 ↓5*	**↑25 ↓18**
T_5	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑4 ↓2**	*↑2 ↓3*	*↑2 ↓4*	*↑1 ↓4*	*↑2 ↓5*	**↑24 ↓18**
T_6	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓2**	*↑1 ↓4*	*↑1 ↓4*	*↑1 ↓4*	**↑26 ↓14**
All	(54)	*↑0 ↓1*	**↑41 ↓1**	**↑51 ↓1**	**↑37 ↓4**	**↑21 ↓14**	*↑5 ↓31*	*↑5 ↓39*	*↑6 ↓38*	**↑166 ↓129**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑7 ↓0**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑8 ↓0**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑5 ↓2**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑8 ↓0**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑8 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑8 ↓0**
C	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑22 ↓2**
S	(6)	↑0 ↓0	**↑2 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑22 ↓0**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑2 ↓2	*↑0 ↓3*	*↑0 ↓5*	**↑26 ↓10**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑1 ↓3*	*↑0 ↓6*	*↑0 ↓6*	**↑25 ↓15**
F_2	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓2**	*↑0 ↓4*	**↑28 ↓7**
F_3	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑2 ↓1**	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	*↑13 ↓19*
F_4	(6)	*↑1 ↓2*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓1**	**↑3 ↓0**	*↑0 ↓3*	**↑29 ↓6**
F_5	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑2 ↓0**	*↑0 ↓1*	**↑31 ↓2**
F_6	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑41 ↓0**
T_1	(7)	*↑0 ↓2*	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑3 ↓2**	*↑1 ↓5*	**↑35 ↓12**
T_2	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	*↑2 ↓3*	*↑1 ↓4*	*↑1 ↓6*	**↑30 ↓13**
T_3	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑3 ↓1**	↑2 ↓2	*↑1 ↓2*	**↑32 ↓6**
T_4	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	↑3 ↓3	↑3 ↓3	*↑1 ↓5*	**↑33 ↓12**
T_5	(7)	*↑0 ↓2*	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓0**	↑4 ↓0	**↑4 ↓1**	*↑1 ↓3*	*↑1 ↓2*	**↑29 ↓8**
T_6	(7)	**↑1 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑4 ↓2**	**↑3 ↓2**	*↑1 ↓3*	**↑34 ↓8**
All	(54)	*↑1 ↓4*	**↑45 ↓0**	**↑52 ↓0**	**↑48 ↓1**	**↑46 ↓4**	**↑25 ↓11**	*↑13 ↓17*	*↑7 ↓24*	**↑237 ↓61**

Table 4.18: CDE vs DynDE H_{AEBI} performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	**↑10 ↓3**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	**↑9 ↓2**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	**↑10 ↓3**
20	(2)	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	**↑8 ↓4**
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓1*	**↑7 ↓1**
80	(2)	↑0 ↓0	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑10 ↓2**
C	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑1 ↓4*	*↑0 ↓4*	**↑29 ↓8**
S	(6)	*↑0 ↓1*	**↑2 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑0 ↓0	*↑0 ↓5*	**↑25 ↓7**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	↑2 ↓2	*↑1 ↓3*	**↑31 ↓5**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑2 ↓4*	*↑1 ↓3*	**↑32 ↓7**
F_2	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑3 ↓1**	**↑36 ↓4**
F_3	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	↑0 ↓0	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	↑16 ↓16
F_4	(6)	↑0 ↓0	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑4 ↓1**	**↑3 ↓1**	**↑35 ↓3**
F_5	(6)	**↑1 ↓0**	**↑2 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑2 ↓1**	**↑25 ↓2**
F_6	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑37 ↓0**
T_1	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓2**	**↑5 ↓1**	**↑4 ↓2**	**↑37 ↓5**
T_2	(7)	**↑1 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓3**	*↑2 ↓4*	*↑2 ↓5*	**↑34 ↓12**
T_3	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑3 ↓1**	**↑3 ↓2**	**↑2 ↓1**	**↑33 ↓4**
T_4	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	↑3 ↓3	↑3 ↓3	**↑37 ↓7**
T_5	(7)	*↑0 ↓1*	**↑7 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑4 ↓1**	**↑2 ↓1**	**↑37 ↓4**
T_6	(7)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑4 ↓2**	↑2 ↓2	**↑34 ↓5**
All	(54)	*↑1 ↓2*	**↑42 ↓1**	**↑47 ↓0**	**↑52 ↓0**	**↑47 ↓0**	**↑40 ↓9**	**↑22 ↓17**	*↑15 ↓23*	**↑266 ↓52**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓2*	**↑9 ↓6**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	**↑8 ↓4**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	**↑9 ↓3**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	**↑7 ↓5**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	*↑0 ↓2*	**↑10 ↓3**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	*↑0 ↓2*	**↑10 ↓3**
C	(6)	↑0 ↓0	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	*↑0 ↓6*	**↑28 ↓6**
S	(6)	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	↑3 ↓3	↑3 ↓3	*↑1 ↓5*	*↑0 ↓6*	**↑25 ↓18**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	↑2 ↓2	**↑35 ↓2**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	*↑1 ↓3*	**↑34 ↓3**
F_2	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑2 ↓1**	**↑36 ↓2**
F_3	(6)	↑0 ↓0	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓6*	**↑19 ↓14**
F_4	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	↑2 ↓2	**↑37 ↓3**
F_5	(6)	**↑1 ↓0**	**↑1 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑21 ↓0**
F_6	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑2 ↓2	↑2 ↓2	**↑4 ↓2**	**↑31 ↓7**
T_1	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓1**	**↑38 ↓2**
T_2	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓0**	*↑2 ↓3*	*↑2 ↓4*	**↑36 ↓7**
T_3	(7)	↑0 ↓0	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓2**	**↑4 ↓2**	*↑1 ↓3*	**↑35 ↓7**
T_4	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑4 ↓3**	**↑41 ↓5**
T_5	(7)	**↑1 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	**↑3 ↓2**	**↑3 ↓2**	**↑34 ↓6**
T_6	(7)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓1**	*↑1 ↓3*	**↑29 ↓4**
All	(54)	**↑2 ↓1**	**↑38 ↓0**	**↑51 ↓0**	**↑50 ↓1**	**↑47 ↓3**	**↑38 ↓7**	**↑25 ↓15**	*↑15 ↓28*	**↑266 ↓55**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑1 ↓0**	**↑7 ↓2**	**↑6 ↓2**	**↑5 ↓1**	**↑6 ↓2**	**↑5 ↓2**	*↑3 ↓4*	*↑1 ↓6*	**↑34 ↓19**
5	(10)	↑0 ↓0	**↑5 ↓0**	**↑10 ↓0**	**↑7 ↓1**	**↑6 ↓1**	**↑6 ↓0**	*↑1 ↓4*	*↑1 ↓5*	**↑36 ↓11**
10	(10)	↑0 ↓0	**↑5 ↓0**	**↑8 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓1**	*↑0 ↓6*	*↑1 ↓7*	**↑33 ↓14**
20	(10)	*↑0 ↓1*	**↑4 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑5 ↓2**	↑3 ↓3	*↑2 ↓4*	*↑1 ↓6*	**↑34 ↓16**
40	(10)	↑0 ↓0	**↑2 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑5 ↓2**	*↑2 ↓3*	*↑0 ↓5*	**↑37 ↓11**
80	(10)	*↑0 ↓1*	**↑2 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑4 ↓3**	↑3 ↓3	*↑1 ↓5*	**↑39 ↓13**
C	(6)	↑0 ↓0	**↑15 ↓0**	**↑29 ↓0**	**↑25 ↓1**	**↑20 ↓3**	**↑12 ↓7**	*↑6 ↓12*	*↑0 ↓17*	**↑107 ↓40**
S	(6)	*↑1 ↓2*	**↑10 ↓3**	**↑23 ↓2**	**↑24 ↓1**	**↑22 ↓3**	**↑16 ↓4**	*↑5 ↓12*	*↑5 ↓17*	**↑106 ↓44**
		GDBG								
F_{1a}	(30)	*↑0 ↓1*	**↑28 ↓0**	**↑30 ↓0**	**↑24 ↓4**	**↑18 ↓10**	*↑13 ↓14*	*↑6 ↓17*	*↑3 ↓22*	**↑122 ↓68**
F_{1b}	(30)	**↑1 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑24 ↓6**	**↑18 ↓11**	*↑12 ↓15*	*↑5 ↓22*	*↑2 ↓24*	**↑122 ↓78**
F_2	(30)	*↑0 ↓3*	**↑28 ↓0**	**↑29 ↓0**	**↑20 ↓8**	**↑19 ↓9**	**↑15 ↓13**	*↑12 ↓16*	*↑5 ↓18*	**↑128 ↓67**
F_3	(30)	↑0 ↓0	**↑22 ↓0**	**↑26 ↓0**	**↑14 ↓3**	**↑5 ↓4**	*↑0 ↓14*	*↑2 ↓16*	*↑3 ↓15*	**↑72 ↓52**
F_4	(30)	*↑1 ↓2*	**↑25 ↓0**	**↑29 ↓0**	**↑26 ↓2**	**↑20 ↓7**	↑14 ↓14	*↑12 ↓14*	*↑5 ↓18*	**↑132 ↓57**
F_5	(30)	↑2 ↓2	**↑21 ↓0**	**↑24 ↓0**	**↑23 ↓2**	**↑20 ↓6**	**↑13 ↓10**	*↑9 ↓10*	*↑6 ↓13*	**↑118 ↓43**
F_6	(30)	*↑0 ↓1*	**↑29 ↓0**	**↑30 ↓0**	**↑26 ↓1**	**↑24 ↓1**	**↑16 ↓5**	**↑16 ↓7**	**↑20 ↓6**	**↑161 ↓21**
T_1	(35)	*↑0 ↓2*	**↑33 ↓0**	**↑32 ↓0**	**↑24 ↓7**	**↑20 ↓9**	↑15 ↓15	*↑14 ↓16*	*↑10 ↓20*	**↑148 ↓69**
T_2	(35)	↑1 ↓1	**↑32 ↓0**	**↑33 ↓0**	**↑28 ↓4**	**↑21 ↓8**	*↑12 ↓16*	*↑6 ↓22*	*↑6 ↓26*	**↑139 ↓77**
T_3	(35)	*↑0 ↓2*	**↑29 ↓0**	**↑35 ↓0**	**↑26 ↓2**	**↑21 ↓8**	*↑12 ↓14*	*↑10 ↓16*	*↑5 ↓16*	**↑138 ↓58**
T_4	(35)	*↑0 ↓1*	**↑32 ↓0**	**↑34 ↓0**	**↑27 ↓4**	**↑23 ↓8**	*↑14 ↓15*	*↑11 ↓19*	*↑9 ↓22*	**↑150 ↓69**
T_5	(35)	*↑2 ↓3*	**↑30 ↓0**	**↑32 ↓0**	**↑24 ↓6**	**↑18 ↓7**	**↑15 ↓13**	*↑10 ↓15*	*↑8 ↓15*	**↑139 ↓59**
T_6	(35)	**↑1 ↓0**	**↑27 ↓0**	**↑32 ↓0**	**↑28 ↓3**	**↑21 ↓8**	**↑15 ↓12**	*↑11 ↓14*	*↑6 ↓17*	**↑141 ↓54**
All	(270)	*↑5 ↓11*	**↑208 ↓3**	**↑250 ↓2**	**↑206 ↓28**	**↑166 ↓54**	**↑111 ↓96**	*↑73 ↓126*	*↑49 ↓150*	**↑1068 ↓470**

of dimensions listed in Table 4.3. This is the same set of experiments used to compare CPE, RMC and CDE to DynDE. *jDE* was thus tested on 2 160 different dynamic environments. The offline errors resulting from the *jDE* experiments were analysed with respect to the offline errors produced by CDE. The number of times that each algorithm was statistically significantly better than the other (according to Mann-Whitney U tests), was counted. The analysis is summarised in Tables 4.19 and 4.20.

CDE performed significantly better than *jDE* in 1 427 cases and worse in 472 cases. The analysis over all dimensions (refer to Table 4.20) showed that *jDE* performed better than CDE more often when a change period of 100 function evaluations was used. At this change period CDE reduced to DynDE as the competitive population evaluation component only functions after the second generation. *jDE* also performed better than CDE on the GDBG function F_3. The scalability study presented in Section 4.6.4 found that CDE and DynDE performed similarly on function F_3, and always resulted in a large offline error. A possible explanation for the comparatively good performance of *jDE* on function F_3 is that F_3 has a large number of local optima (refer to Figure 2.6). The ageing component of *jDE*, which continuously reinitialises individuals, may be useful on this function to reduce the likelihood of convergence to local optima.

jDE performed better more often than CDE in low dimensions when a change period of 100 000 was used. This result is to be expected as previous sections found that CDE's performance deteriorated with respect to DynDE's when the change period was increased (refer to Section 4.6.5.3). *jDE* also performed better than CDE on the MPB functions when a change period of 500 was used as insufficient function evaluations are available to the competitive population evaluation to start locating optima early.

The *jDE* algorithm maintains a history archive of previously found good solutions. The purpose of this archive is to make the algorithms more effective on environments which are cyclic (refer to the definition in Section 2.5.2). *jDE* did not, however, perform better than CDE more often on the two recurrent change types of the GDBG (T_5 and T_6, described in Section 2.5.4.2). This result is unexpected, as CDE does not make use of a history archive.

The analysis in this section shows that, although *jDE* performs better than CDE in isolated cases, CDE generally performs better than *jDE*.

Table 4.19: CDE vs *jDE* performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
5	(2)	↑*0* ↓*1*	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**12** ↓**2**
10	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**11** ↓**4**
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**5**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*6* ↓*10*
80	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*6* ↓*9*
C	(6)	↑*0* ↓*3*	↑*1* ↓*4*	↑**3** ↓**2**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**34** ↓**9**
S	(6)	↑*0* ↓*6*	↑*1* ↓*5*	↑*2* ↓*4*	↑**6** ↓**0**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**2**	↑**25** ↓**23**
		GDBG								
F_{1a}	(6)	↑**1** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**2** ↓**1**	↑*2* ↓*4*	↑**35** ↓**5**
F_{1b}	(6)	↑**2** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑*2* ↓*3*	↑*1* ↓*4*	↑**33** ↓**9**
F_2	(6)	↑1 ↓1	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑*0* ↓*6*	↑*0* ↓*6*	↑**28** ↓**14**
F_3	(6)	↑**1** ↓**0**	↑*1* ↓*4*	↑**4** ↓**0**	↑**5** ↓**0**	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*11* ↓*28*
F_4	(6)	↑1 ↓1	↑**2** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑*1* ↓*3*	↑**34** ↓**5**
F_5	(6)	↑2 ↓2	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑*1* ↓*3*	↑**39** ↓**5**
F_6	(6)	↑**1** ↓**0**	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑*1* ↓*4*	↑*0* ↓*6*	↑**28** ↓**10**
T_1	(7)	↑**4** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**4** ↓**2**	↑*2* ↓*3*	↑*0* ↓*7*	↑**37** ↓**13**
T_2	(7)	↑**1** ↓**0**	↑**5** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑*1* ↓*3*	↑**38** ↓**8**
T_3	(7)	↑*0* ↓*1*	↑**5** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**4** ↓**3**	↑*2* ↓*3*	↑**37** ↓**10**
T_4	(7)	↑**4** ↓**0**	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑*2* ↓*5*	↑*0* ↓*7*	↑**33** ↓**15**
T_5	(7)	↑*0* ↓*3*	↑**3** ↓**2**	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**4** ↓**2**	↑*2* ↓*3*	↑*2* ↓*5*	↑**30** ↓**16**
T_6	(7)	↑*0* ↓*1*	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑*2* ↓*4*	↑*0* ↓*7*	↑**33** ↓**14**
All	(54)	↑*9* ↓*14*	↑**32** ↓**14**	↑**45** ↓**6**	↑**53** ↓**0**	↑**46** ↓**8**	↑**40** ↓**10**	↑**27** ↓**22**	↑*15* ↓*34*	↑**267** ↓**108**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**9** ↓**4**
5	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**11** ↓**3**
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**11** ↓**5**
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**6**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**9** ↓**7**
80	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*5* ↓*10*
C	(6)	↑*0* ↓*6*	↑*0* ↓*4*	↑3 ↓3	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**33** ↓**13**
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑**3** ↓**2**	↑**4** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑22 ↓22
		GDBG								
F_{1a}	(6)	↑**2** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑*2* ↓*3*	↑**40** ↓**3**
F_{1b}	(6)	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**2**	↑*2* ↓*3*	↑**39** ↓**5**
F_2	(6)	↑**2** ↓**1**	↑**3** ↓**1**	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**39** ↓**4**
F_3	(6)	↑**2** ↓**0**	↑*0* ↓*5*	↑*0* ↓*1*	↑**6** ↓**0**	↑**4** ↓**0**	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*12* ↓*24*
F_4	(6)	↑**2** ↓**1**	↑2 ↓2	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**39** ↓**4**
F_5	(6)	↑**3** ↓**2**	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**3**
F_6	(6)	↑**2** ↓**0**	↑**2** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**3** ↓**2**	↑3 ↓3	↑**34** ↓**5**
T_1	(7)	↑**2** ↓**0**	↑**6** ↓**1**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**2**	↑*3* ↓*4*	↑**42** ↓**8**
T_2	(7)	↑*2* ↓*3*	↑**3** ↓**1**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**43** ↓**7**
T_3	(7)	↑*0* ↓*1*	↑2 ↓2	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**38** ↓**8**
T_4	(7)	↑**5** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**5** ↓**2**	↑**4** ↓**3**	↑**45** ↓**8**
T_5	(7)	↑**2** ↓**0**	↑*2* ↓*4*	↑**4** ↓**2**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**5** ↓**2**	↑*3* ↓*4*	↑**35** ↓**13**
T_6	(7)	↑**5** ↓**0**	↑**4** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑**3** ↓**2**	↑**43** ↓**4**
All	(54)	↑16 ↓16	↑**23** ↓**19**	↑**37** ↓**11**	↑**51** ↓**2**	↑**50** ↓**0**	↑**47** ↓**7**	↑**42** ↓**11**	↑**35** ↓**17**	↑**301** ↓**83**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*6* ↓*9*
5	(2)	↑*0* ↓*1*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑6 ↓6
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**9** ↓**5**
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**5**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**5**
80	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**9** ↓**5**
C	(6)	↑*0* ↓*5*	↑*0* ↓*4*	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**33** ↓**9**
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑2 ↓2	↑**3** ↓**2**	↑**4** ↓**2**	↑**4** ↓**1**	↑**4** ↓**1**	↑*17* ↓*26*
		GDBG								
F_{1a}	(6)	↑**3** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**43** ↓**0**
F_{1b}	(6)	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑**44** ↓**1**
F_2	(6)	↑**3** ↓**1**	↑**3** ↓**1**	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**40** ↓**3**
F_3	(6)	↑2 ↓2	↑*0* ↓*5*	↑*0* ↓*5*	↑**3** ↓**0**	↑**4** ↓**0**	↑2 ↓2	↑*0* ↓*6*	↑*0* ↓*6*	↑*11* ↓*26*
F_4	(6)	↑**2** ↓**0**	↑**3** ↓**1**	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**39** ↓**2**
F_5	(6)	↑**2** ↓**0**	↑**3** ↓**1**	↑**3** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**33** ↓**2**
F_6	(6)	↑**3** ↓**2**	↑**2** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**3** ↓**1**	↑**36** ↓**4**
T_1	(7)	↑0 ↓0	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**41** ↓**5**
T_2	(7)	↑*1* ↓*3*	↑**5** ↓**1**	↑**6** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**45** ↓**7**
T_3	(7)	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑**35** ↓**6**
T_4	(7)	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**4** ↓**2**	↑**47** ↓**6**
T_5	(7)	↑**5** ↓**0**	↑*2* ↓*4*	↑3 ↓3	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**5** ↓**2**	↑**4** ↓**2**	↑**38** ↓**11**
T_6	(7)	↑**6** ↓**0**	↑**2** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**4** ↓**2**	↑**40** ↓**3**
All	(54)	↑**19** ↓**16**	↑**22** ↓**18**	↑**31** ↓**13**	↑**46** ↓**2**	↑**48** ↓**2**	↑**48** ↓**4**	↑**44** ↓**8**	↑**38** ↓**10**	↑**296** ↓**73**

Table 4.20: CDE vs *jDE* performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*6* ↓*9*
5	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*6* ↓*9*
10	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑6 ↓4**
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓5**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓5**
80	(2)	↑*0* ↓*1*	↑*0* ↓*1*	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓3**
C	(6)	↑*0* ↓*5*	↑*0* ↓*2*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑35 ↓7**
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	**↑3 ↓2**	**↑3 ↓2**	**↑3 ↓2**	**↑3 ↓2**	**↑3 ↓2**	↑*15* ↓*28*
		GDBG								
F_{1a}	(6)	**↑2 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓0**
F_{1b}	(6)	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑47 ↓0**
F_2	(6)	↑*2* ↓*3*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑44 ↓3**
F_3	(6)	↑*2* ↓*3*	↑*0* ↓*6*	↑*0* ↓*5*	↑*1* ↓*3*	**↑3 ↓1**	**↑3 ↓2**	↑*0* ↓*5*	↑*0* ↓*6*	↑*9* ↓*31*
F_4	(6)	↑3 ↓3	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑44 ↓3**
F_5	(6)	↑1 ↓1	↑2 ↓2	↑*2* ↓*3*	↑*2* ↓*3*	**↑2 ↓1**	**↑4 ↓1**	**↑5 ↓1**	**↑3 ↓1**	**↑21 ↓13**
F_6	(6)	↑*2* ↓*3*	↑*1* ↓*4*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓1**	**↑34 ↓8**
T_1	(7)	↑*1* ↓*5*	**↑6 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑42 ↓11**
T_2	(7)	↑*1* ↓*4*	**↑5 ↓2**	**↑6 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑44 ↓9**
T_3	(7)	↑*1* ↓*4*	**↑3 ↓2**	**↑4 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑5 ↓1**	**↑38 ↓10**
T_4	(7)	**↑5 ↓0**	**↑3 ↓2**	**↑5 ↓2**	**↑5 ↓1**	**↑5 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑40 ↓8**
T_5	(7)	**↑3 ↓0**	**↑4 ↓3**	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓1**	**↑5 ↓0**	**↑6 ↓1**	**↑4 ↓2**	**↑37 ↓11**
T_6	(7)	**↑6 ↓0**	**↑4 ↓2**	**↑4 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑4 ↓2**	**↑40 ↓9**
All	(54)	↑*17* ↓*24*	**↑25 ↓20**	**↑34 ↓14**	**↑42 ↓8**	**↑44 ↓4**	**↑46 ↓5**	**↑44 ↓8**	**↑39 ↓10**	**↑291 ↓93**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*1*	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑6 ↓3**
5	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑*6* ↓*8*
10	(2)	↑*0* ↓*2*	↑*0* ↓*1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑10 ↓5**
20	(2)	↑*0* ↓*2*	↑*0* ↓*1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑11 ↓4**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓4**
80	(2)	↑*0* ↓*2*	↑*0* ↓*2*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓4**
C	(6)	↑*0* ↓*6*	↑*0* ↓*3*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑36 ↓9**
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	**↑2 ↓1**	**↑4 ↓0**	**↑4 ↓1**	**↑4 ↓1**	↑2 ↓2	**↑3 ↓2**	↑19 ↓19
		GDBG								
F_{1a}	(6)	**↑3 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓1**
F_{1b}	(6)	**↑3 ↓1**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓1**
F_2	(6)	↑*2* ↓*3*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑44 ↓3**
F_3	(6)	↑*2* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*4*	**↑3 ↓1**	**↑5 ↓0**	↑*0* ↓*3*	↑*0* ↓*6*	↑*10* ↓*29*
F_4	(6)	↑*2* ↓*3*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑44 ↓3**
F_5	(6)	↑*0* ↓*2*	↑*0* ↓*6*	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*5*	↑*1* ↓*5*	↑*1* ↓*38*
F_6	(6)	↑*2* ↓*3*	↑*0* ↓*4*	**↑3 ↓2**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑3 ↓3	**↑32 ↓12**
T_1	(7)	↑*0* ↓*5*	**↑4 ↓2**	**↑5 ↓2**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑36 ↓16**
T_2	(7)	↑*2* ↓*4*	**↑4 ↓3**	**↑5 ↓1**	**↑5 ↓2**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓1**	**↑39 ↓11**
T_3	(7)	↑*0* ↓*5*	↑*2* ↓*3*	**↑3 ↓2**	**↑5 ↓2**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑4 ↓3**	**↑31 ↓18**
T_4	(7)	**↑4 ↓2**	**↑4 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑39 ↓14**
T_5	(7)	**↑2 ↓0**	**↑4 ↓3**	**↑4 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓1**	**↑5 ↓2**	**↑4 ↓3**	**↑34 ↓15**
T_6	(7)	**↑6 ↓0**	↑3 ↓3	**↑4 ↓3**	**↑5 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑4 ↓3**	**↑38 ↓13**
All	(54)	↑*14* ↓*28*	↑*21* ↓*25*	**↑34 ↓13**	**↑40 ↓10**	**↑43 ↓7**	**↑45 ↓6**	**↑38 ↓10**	**↑37 ↓16**	**↑272 ↓115**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑*0* ↓*10*	↑*2* ↓*4*	**↑6 ↓3**	**↑6 ↓2**	**↑6 ↓2**	**↑7 ↓2**	**↑7 ↓2**	**↑7 ↓2**	**↑41 ↓27**
5	(10)	↑*0* ↓*8*	↑*0* ↓*6*	**↑6 ↓3**	**↑7 ↓2**	**↑7 ↓3**	**↑7 ↓3**	**↑7 ↓1**	**↑7 ↓2**	**↑41 ↓28**
10	(10)	↑*0* ↓*9*	↑*0* ↓*8*	**↑6 ↓4**	**↑8 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑8 ↓1**	**↑8 ↓1**	**↑47 ↓23**
20	(10)	↑*0* ↓*10*	↑*0* ↓*9*	↑*3* ↓*5*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑10 ↓0**	**↑52 ↓25**
40	(10)	↑*0* ↓*10*	↑*0* ↓*10*	↑*1* ↓*6*	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑46 ↓31**
80	(10)	↑*0* ↓*8*	↑*0* ↓*9*	↑*2* ↓*6*	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑42 ↓31**
C	(6)	↑*0* ↓*25*	↑*1* ↓*17*	**↑20 ↓5**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑171 ↓47**
S	(6)	↑*0* ↓*30*	↑*1* ↓*29*	↑*4* ↓*22*	**↑18 ↓6**	**↑18 ↓7**	**↑20 ↓8**	**↑18 ↓8**	**↑19 ↓8**	↑*98* ↓*118*
		GDBG								
F_{1a}	(30)	**↑11 ↓1**	**↑27 ↓0**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑26 ↓1**	**↑21 ↓7**	**↑203 ↓9**
F_{1b}	(30)	**↑17 ↓2**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑28 ↓1**	**↑24 ↓5**	**↑19 ↓8**	**↑206 ↓16**
F_2	(30)	**↑10 ↓9**	**↑23 ↓2**	**↑27 ↓2**	**↑30 ↓0**	**↑30 ↓0**	**↑28 ↓1**	**↑24 ↓6**	**↑23 ↓7**	**↑195 ↓27**
F_3	(30)	**↑9 ↓8**	↑*1* ↓*26*	↑*4* ↓*17*	**↑15 ↓7**	**↑14 ↓8**	↑*10* ↓*16*	↑*0* ↓*26*	↑*0* ↓*30*	↑*53* ↓*138*
F_4	(30)	**↑10 ↓8**	**↑18 ↓4**	**↑27 ↓2**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑25 ↓3**	**↑200 ↓17**
F_5	(30)	**↑8 ↓7**	**↑15 ↓10**	**↑17 ↓7**	**↑19 ↓9**	**↑19 ↓6**	**↑22 ↓6**	**↑22 ↓6**	**↑15 ↓10**	**↑137 ↓61**
F_6	(30)	**↑10 ↓8**	**↑9 ↓8**	**↑24 ↓2**	**↑30 ↓0**	**↑30 ↓0**	**↑28 ↓0**	**↑21 ↓7**	↑*12* ↓*14*	**↑164 ↓39**
T_1	(35)	↑7 ↓*10*	**↑29 ↓5**	**↑29 ↓4**	**↑31 ↓2**	**↑31 ↓2**	**↑28 ↓6**	**↑24 ↓9**	**↑19 ↓15**	**↑198 ↓53**
T_2	(35)	↑7 ↓*14*	**↑22 ↓8**	**↑30 ↓3**	**↑32 ↓2**	**↑33 ↓1**	**↑32 ↓2**	**↑28 ↓5**	**↑25 ↓7**	**↑209 ↓42**
T_3	(35)	↑*1* ↓*13*	**↑13 ↓9**	**↑22 ↓4**	**↑32 ↓3**	**↑32 ↓2**	**↑32 ↓3**	**↑26 ↓8**	**↑21 ↓10**	**↑179 ↓52**
T_4	(35)	**↑25 ↓2**	**↑23 ↓7**	**↑28 ↓6**	**↑29 ↓3**	**↑28 ↓2**	**↑29 ↓5**	**↑24 ↓11**	**↑18 ↓15**	**↑204 ↓51**
T_5	(35)	**↑12 ↓3**	↑*15* ↓*16*	**↑22 ↓9**	**↑30 ↓4**	**↑29 ↓4**	**↑26 ↓4**	**↑23 ↓10**	**↑17 ↓16**	**↑174 ↓66**
T_6	(35)	**↑23 ↓1**	**↑19 ↓5**	**↑26 ↓4**	**↑30 ↓2**	**↑30 ↓3**	**↑29 ↓4**	**↑22 ↓8**	↑*15* ↓*16*	**↑194 ↓43**
All	(270)	↑*75* ↓*98*	**↑123 ↓96**	**↑181 ↓57**	**↑232 ↓22**	**↑231 ↓21**	**↑226 ↓32**	**↑195 ↓59**	**↑164 ↓87**	**↑1427 ↓472**

4.7.2 CDE Compared to Other Algorithms

This section compares the performance CDE to the results of other algorithms published in the literature. Algorithms are generally required to detect changes in the environment, which makes a direct comparison with the results presented in this chapter inappropriate, as CDE used an automatic detection strategy which allowed the algorithm to check for changes in the environment without using function evaluations. A fair comparison to the published results of other algorithms necessitates the incorporation of a detection strategy into CDE.

The algorithms used for comparison in this section all use one of two detection strategies. The first strategy is to re-evaluate the performance of the best individual that has been found since the last change in the environment. A change is detected if the fitness of the individual changed since it was last evaluated. The second detection strategy, commonly used, is to re-evaluate the fitness of the best individual in each of the sub-populations. A change is detected if the fitness of any the best individuals has changed since it was last evaluated. The first strategy has the advantage of using only one function evaluation per generation to detect changes, but the disadvantage that sampling only a single point in the fitness landscape could lead to the strategy's not detecting a change when it occurs. The second strategy has the advantage of sampling multiple points in the fitness landscape, but unfortunately uses more function evaluations to detect the changes than the first strategy does.

Any two algorithms should ideally use the same change detection strategy when being compared. However, using the same detection strategy does not guarantee that the algorithms will use the same number of function evaluations to detect changes, or that changes will be detected equally effectively. The functioning of each algorithm influences the performance of a change detection strategy. For example, the number of sub-populations and the sub-population size used by an algorithm determines the number of function evaluations between generations, and consequently how frequently the change detection strategy is performed. Algorithms that employ a large sub-population size or use a large number of sub-populations, perform change detections less frequently than algorithms that uses a small number of sub-populations or a small sub-population size. The problem

is further complicated by the fact that not all algorithms maintain a constant number of sub-populations during the optimisation process. Furthermore, CDE only evolves one sub-population per generation, which leads to fewer function evaluations between executions of the change detection strategy.

This section presents CDE results, using several change detection strategies on variations of the MPB Scenario 2 which are commonly used by researchers to evaluate their algorithms. The results of each of the algorithms used for comparison can then be compared to CDE's results using the closest matching detection strategy. Four detection strategies were selected for incorporation into CDE. The first, Det_{best}, re-evaluates the best individual over all sub-populations, after each generation. Det_{best} thus uses a single function evaluation per generation. The second change detection strategy, Det_{local}, re-evaluates the best individual in each sub-population after each generation. Det_{local} thus uses n_k function evaluations per generation, where n_k is the number of sub-populations. The Det_{best} and Det_{local} change detection strategies function differently on CDE than on standard multi-population algorithms (like DynDE) where all sub-populations are evolved per generation. The third and fourth change detection strategies are variations of Det_{best} and Det_{local} where the detection is not performed in each generation, but once every n_k generations. These strategies are denoted by Det_{n_k-best} and $Det_{n_k-local}$. Detecting changes once every n_k generations ensures that the number of function evaluations used by Det_{n_k-best} and $Det_{n_k-local}$ in CDE would, respectively, be similar to the number of function evaluations used by Det_{best} and Det_{local} in standard multi-population algorithms.

The offline errors and 95% confidence intervals of CDE, using the four change detection strategies on variations of the MPB Scenario 2, are given in Table 4.21. Outcomes of Mann-Whitney U tests comparing the results of CDE using each of the detection strategies and CDE using the automatic detection strategy are included in Table 4.21. Results that are statistically significantly worse than when the automatic detection strategy is used are printed in italics. The results show that Det_{best} did not statistically significantly affect the performance of CDE. The Det_{local} detection strategy resulted in inferior performance by CDE in almost all cases. This bad performance when using Det_{local} is due to too many function evaluations being wasted to detect changes. The CDE algorithm uses 10 sub-populations. During the period when only one sub-population is evolved per generation,

the algorithm, on average, uses more than half of the function evaluations per generation to detect changes using Det_{local}. The benefits of detecting changes correctly are thus outweighed by the cost in function evaluations. The performance of CDE deteriorated significantly in only two cases when using Det_{n_k-best} and four cases when using $Det_{n_k-local}$. Changes can thus be effectively detected without a large deterioration in performance on the MBP, if an appropriate detection strategy is used.

Table 4.21: CDE using various detection strategies

Settings	**Automatic**	Det_{best}	**p-val**	Det_{local}	**p-val**	Det_{n_k-best}	**p-val**	$Det_{n_k-local}$	**p-val**
C_s 1	0.79 ± 0.13	0.79 ± 0.15	0.686	0.9 ± 0.09	0.061	0.71 ± 0.12	0.415	0.71 ± 0.1	0.467
C_s 2	1.1 ± 0.08	1.21 ± 0.13	0.39	*1.62 ± 0.16*	0.000	*1.29 ± 0.12*	0.026	1.11 ± 0.1	0.947
C_s 4	2.17 ± 0.12	2.17 ± 0.13	0.901	*2.96 ± 0.22*	0.000	2.19 ± 0.15	0.901	2.41 ± 0.19	0.075
C_s 6	3.22 ± 0.19	3.46 ± 0.25	0.138	*4.61 ± 0.26*	0.000	*3.45 ± 0.15*	0.034	*3.72 ± 0.21*	0.001
Cp 500	4.26 ± 0.12	4.19 ± 0.12	0.293	*5.47 ± 0.28*	0.000	4.36 ± 0.17	0.44	*4.52 ± 0.18*	0.046
Cp 1000	2.43 ± 0.11	2.36 ± 0.09	0.504	*3.05 ± 0.12*	0.000	2.55 ± 0.11	0.146	*2.60 ± 0.12*	0.036
Cp 2500	1.16 ± 0.1	1.16 ± 0.1	0.82	*1.49 ± 0.11*	0.000	1.13 ± 0.12	0.307	1.13 ± 0.09	0.592
Cp 10000	0.52 ± 0.15	0.45 ± 0.11	0.467	0.58 ± 0.14	0.127	0.49 ± 0.11	0.994	0.43 ± 0.08	0.592
n_d 10	1.9 ± 0.24	2.03 ± 0.24	0.382	*2.66 ± 0.29*	0.000	2.02 ± 0.3	0.476	2.09 ± 0.32	0.532
n_d 50	12.4 ± 1.13	12.77 ± 0.8	0.134	*18.80 ± 1.24*	0.000	11.43 ± 0.67	0.523	12.74 ± 0.89	0.219
n_d 100	25.81 ± 1.71	26.27 ± 1.91	0.901	*40.09 ± 3.95*	0.000	24.42 ± 1.57	0.314	26.51 ± 2.2	0.843

Table 4.22 lists the published results for variations of the MBP Scenario 2 for 12 of the most seminal and modern algorithms aimed at the MPB. An overview of these algorithms was given in Section 3.3. The 95% confidence intervals were calculated from the reported standard errors or standard deviations in cases where the confidence interval was not reported. Results that are better than the corresponding CDE results are printed in italics while results that are worse than CDE's are printed in boldface in shaded cells. Results with overlapping confidence intervals are considered to be similar and are consequently printed in the normal font.

Table 4.22: Reported offline errors of various algorithms

	MMEO	**HJEO**	**LSEO**	**CESO**	**ESCA**	**MPSO**
C_s 1	0.66 ± 0.39	*0.25 ± 0.20*	*0.25 ± 0.16*	**1.38 ± 0.04**	**1.53 ± 0.02**	**1.75 ± 0.12**
C_s 2	0.86 ± 0.41	*0.52 ± 0.27*	*0.47 ± 0.24*	**1.78 ± 0.04**	**1.57 ± 0.02**	**2.4 ± 0.12**
C_s 4	*0.97 ± 0.41*	*0.64 ± 0.31*	*0.53 ± 0.25*	2.23 ± 0.10	*1.72 ± 0.06*	**3.59 ± 0.20**
C_s 6	*1.09 ± 0.43*	*0.9 ± 0.33*	*0.77 ± 0.47*	*2.74 ± 0.20*	*1.79 ± 0.06*	**4.79 ± 0.20**
n_d 10	2.44 ± 1.51	2.17 ± 1.57	2.25 ± 1.67	**2.51 ± 0.08**	N/A	**4.17 ± 0.29**
n_d 50	**206.3 ± 69.97**	*5.79 ± 2.74*	*6.22 ± 3.14*	*6.81 ± 0.14*	N/A	N/A
n_d 100	**480.5 ± 137.39**	16.5 ± 10.86	17.8 ± 13.52	24.6 ± 0.49	N/A	N/A
	CPSO	**MPSOD**	**HMSO**	**MSO**	**Cellular DE**	**Cellular PSO**
C_s 1	**1.06 ± 0.07**	**1.06 ± 0.03**	**1.42 ± 0.04**	**1.51 ± 0.04**	**1.64 ± 0.03**	**1.14 ± 0.13**
C_s 2	1.17 ± 0.06	**1.51 ± 0.04**	N/A	N/A	N/A	N/A
C_s 4	*1.38 ± 0.08*	N/A	N/A	N/A	N/A	N/A
C_s 6	*1.53 ± 0.08*	N/A	N/A	N/A	N/A	N/A
Cp 500	N/A	N/A	**7.56 ± 0.27**	**5.95 ± 0.09**	N/A	*1.44 ± 0.13*
Cp 1000	N/A	**3.58 ± 0.05**	**4.61 ± 0.07**	**3.94 ± 0.08**	**3.98 ± 0.03**	*1.33 ± 0.11*
Cp 2500	N/A	N/A	**2.39 ± 0.16**	N/A	**2.42 ± 0.02**	1.08 ± 0.09
Cp 10000	0.63 ± N/A	N/A	**0.94 ± 0.09**	**0.97 ± 0.04**	N/A	**1.1 ± 0.18**

The comparison of CDE with each of the tabulated algorithms are briefly discussed here.

MMEO [Moser and Hendtlass, 2007] MMEO is an algorithm based on extremal optimisation and searches for optima in parallel. The algorithm detects changes by re-evaluating all the best solutions in the search space and is thus comparable to the Det_{local} and $Det_{n_k-local}$ detection strategies. MMEO yielded a lower offline error than CDE on experiments with change severities 1 and 2, but confidence intervals of the two algorithms overlap and MMEO, accordingly, cannot conclusively be considered superior to CDE. Change severities of 4 and 6 yielded MMEO results that are clearly superior to CDE's results. Overlapping confidence intervals were found in 10 dimensions, but CDE clearly performed better than MMEO in 50 and 100 dimensions.

HJEO [Moser, 2007] HJEO is an extension of MMEO which incorporates a Hooke-Jeeves local search. HJEO performed better than CDE on all reported cases, except 10 and 100 dimensions where the confidence intervals overlapped.

LSEO [Moser and Chiong, 2010] The local search component of MMEO was further improved in the LSEO algorithm. LSEO performed better than CDE on all reported cases, except in 10 and 100 dimensions, where the confidence intervals overlapped.

CESO [Lung and Dumitrescu, 2007] CESO uses a single DE population and a single PSO population to solve DOPs. Changes are detected by re-evaluating the best individual in the DE population and is thus comparable to the Det_{best} detection strategy. CDE performed better than CESO on experiments with a change severity of 1 and 2. Overlapping confidence intervals were found when using a change severity of 4, but CESO performed better than CDE when using a change severity of 6. CDE performed better than CESO in 10 dimensions but worse in 50 dimensions. Overlapping confidence intervals were found in 100 dimensions.

ESCA [Lung and Dumitrescu, 2010] ESCA is an adaptation of CESO which employs two DE populations in combination with the PSO population. CDE performed

better than ESCA on change severities of 1 and 2, but worse on change severities of 4 and 6.

MPSO [Blackwell and Branke, 2006] MPSO is a multi-swarm algorithm based on PSO. MPSO uses the Det_{local} detection strategy, and is consequently comparable to CDE using the $Det_{n_k-local}$ detection strategy. CDE performed better than MPSO in all reported cases. Overlapping confidence intervals were found on the change severity of 6 experiment when using Det_{local} detection strategy.

CPSO [Yang and Li, 2010] CPSO clusters particles of an PSO algorithm into sub-swarms to track optima in a dynamic environment. CPSO detects changes using the Det_{best} detection strategy and is thus roughly comparable to CDE using the Det_{n_k-best} strategy. CPE performed better than CPSO in experiments with a change severity of 1, but overlapping confidence intervals were found with a change severity of 2. CPSO performed better than CPE with change severities of 4 and 6. CPE performed better than CPSO when a change period of 10 000 was used, but as Yang and Li [2010] did not report the confidence interval for this experiment, CPE's superiority cannot be confirmed for this experiment.

MPSOD [Novoa-Hernández *et al.*, 2011] MPSOD is an extension of MPSO. The authors do not explicitly state which change detection strategy is used, but it is assumed that MPSOD employs Det_{local} which is used in MPSO. MPSOD is consequently compared to CDE using the $Det_{n_k-local}$ detection strategy. CDE performed better than MPSOD in all reported cases.

HMSO [Kamosi *et al.*, 2010a] HMSO is an extension of MPSO which hibernates unproductive sub-swarms. HMSO uses the Det_{best} change detection strategy, and, as not all sub-populations are evolved per generation, it can be compared to CDE using Det_{best}. CDE performed better than HMSO in all reported cases.

MSO [Kamosi *et al.*, 2010b] MSO is a PSO-based algorithm that utilises a dynamic number of swarms to locate optima in a dynamic environment. Kamosi *et al.* [2010b] do not specify the change detection strategy that is used in MSO, but it is assumed

that the Det_{best} strategy that was used in HMSO (which was developed by the same authors) is also used in MSO. CDE performed better than MSO in all reported cases.

Cellular DE [Noroozi *et al.*, 2011] Cellular DE creates sub-populations by dividing the search space into equally sized cells. Cellular DE employs the Det_{local} change detecting strategy and is thus comparable to CDE using the $Det_{n_k-local}$ strategy. CDE performed better than Cellular DE in all reported cases.

Cellular PSO [Hashemi and Meybodi, 2009a] Cellular PSO creates sub-swarms by dividing the search space into equally sized cells. Cellular PSO uses the Det_{local} change detecting strategy and is thus comparable to CDE using the $Det_{n_k-local}$ strategy. Cellular PSO performed better than CDE when using change periods of 500 and 1 000, but overlapping confidence intervals were found when using a change period of 2 500. CDE performed better than Cellular PSO in when using a change period of 5 000 and 10 000.

This section compared CDE results with the reported results of other algorithms. Moser and Chiong [2010] currently report the lowest offline errors in the literature on the MPB with the LSEO algorithm. The offline errors of LSEO and HJEO are considerably lower than the reported results of any other algorithm, and CDE is no exception. CDE does, however, outperform each of the other algorithms to which it was compared on at least one of the tested environments. The results presented in this section indicate that CDE compares favourably with the state-of-the-art approaches in the literature.

4.8 General Applicability

DynDE [Mendes and Mohais, 2005] was used as base algorithm for the CPE and RMC adaptations in this chapter. However, while DynDE is well suited for adaptation, the novel adaptations presented here can, in theory, be incorporated into any multi-population optimisation algorithm. This section investigates the general applicability of the competitive population evaluation and reinitialisation midpoint check approaches by incorporating these components into *jDE* of Brest *et al.* [2009].

jDE was adapted to incorporate the competitive population evaluation component of CDE to allocate function evaluations to sub-populations based on performance. The exclusion component of *jDE* was replaced with the reinitialisation midpoint check component and exclusion threshold of CDE. The self-adaptive control parameters, aging, local exclusion and custom reinitialisation components of *jDE* were left intact. The combination of *jDE* and CDE is referred to as CjDE.

CjDE was evaluated on the standard set for all combinations of dimension and change period listed in Table 4.3. The results were compared to those of *jDE* and the number of times that each algorithm performed statistically significantly better than the other were counted. The results of this analysis are given in Tables 4.23 and 4.24. The new components improved the *jDE* algorithm in 1 382 cases and deteriorated *jDE*'s performance in only 346 cases. The benefit of the new components are clear at lower change periods, but *jDE* did outperform CjDE at change periods of over 25 000 in low dimensions. This confirms the trends observed earlier in this chapter that the competitive population evaluation approach is only useful in problems with a low change period, and is more useful at high dimensions.

The results of the CDE algorithm were compared to CjDE on the same experimental set as described above. An analysis of the results showed that CDE performed better than CjDE in 1 273 experiments and worse in 631 experiments (full results of this analysis are given in Appendix B). CDE outperformed CjDE in the majority of experiments, but considering that *jDE* outperformed CDE in only 472 experiments and was inferior to CDE in 1 427 cases, it can be concluded that CjDE performed better with respect to CDE than *jDE*.

CPE and RMC outperformed *jDE* in the majority of environments that were investigated. These results prove that the approaches suggested in this chapter can be successfully incorporated into other multi-population algorithms aimed at DOPs.

4.9 Conclusions

This chapter evaluated DynDE on a wide range of environments. An investigation into the appropriate sub-population size and number of Brownian individuals confirmed the

Table 4.23: CjDE vs *jDE* performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓1	↑**2** ↓**0**	↑**1** ↓**0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑3 ↓11
5	(2)	↑0 ↓0	↑**1** ↓**0**	↑**1** ↓**0**	↑0 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑2 ↓8
10	(2)	↑**1** ↓**0**	↑**1** ↓**0**	↑0 ↓1	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑3 ↓9
20	(2)	↑0 ↓0	↑**1** ↓**0**	↑1 ↓1	↑**2** ↓**0**	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑4 ↓8
40	(2)	↑**1** ↓**0**	↑0 ↓1	↑0 ↓1	↑**2** ↓**0**	↑**1** ↓**0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑4 ↓8
80	(2)	↑0 ↓0	↑0 ↓1	↑0 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑0 ↓0	↑0 ↓2	↑0 ↓2	↑4 ↓6
C	(6)	↑**1** ↓**0**	↑**4** ↓**0**	↑**3** ↓**0**	↑**4** ↓**1**	↑1 ↓2	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑13 ↓20
S	(6)	↑1 ↓1	↑1 ↓2	↑0 ↓4	↑3 ↓3	↑2 ↓3	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑7 ↓30
		GDBG								
F_{1a}	(6)	↑0 ↓1	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑1 ↓3	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑17 ↓22
F_{1b}	(6)	↑**2** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑1 ↓3	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑19 ↓21
F_2	(6)	↑0 ↓1	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑2 ↓3	↑0 ↓6	↑0 ↓6	↑**23** ↓**16**
F_3	(6)	↑**3** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**2** ↓**1**	↑0 ↓5	↑0 ↓6	↑**20** ↓**12**
F_4	(6)	↑0 ↓0	↑**4** ↓**1**	↑**3** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑1 ↓1	↑0 ↓6	↑0 ↓6	↑**20** ↓**15**
F_5	(6)	↑**2** ↓**1**	↑**3** ↓**2**	↑**3** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**3** ↓**2**	↑0 ↓6	↑**28** ↓**13**
F_6	(6)	↑**2** ↓**1**	↑**3** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑1 ↓1	↑0 ↓6	↑0 ↓6	↑**23** ↓**14**
T_1	(7)	↑**1** ↓**0**	↑0 ↓2	↑**2** ↓**1**	↑**7** ↓**0**	↑**4** ↓**2**	↑0 ↓6	↑0 ↓7	↑0 ↓7	↑14 ↓25
T_2	(7)	↑**2** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**7** ↓**0**	↑**5** ↓**0**	↑2 ↓3	↑1 ↓6	↑0 ↓7	↑**27** ↓**18**
T_3	(7)	↑**2** ↓**0**	↑**3** ↓**0**	↑**4** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑2 ↓2	↑1 ↓6	↑0 ↓7	↑**24** ↓**15**
T_4	(7)	↑1 ↓1	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**5** ↓**2**	↑3 ↓3	↑0 ↓7	↑0 ↓7	↑**30** ↓**20**
T_5	(7)	↑0 ↓3	↑**5** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**3** ↓**2**	↑1 ↓5	↑0 ↓7	↑**29** ↓**17**
T_6	(7)	↑**3** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**7** ↓**0**	↑**5** ↓**2**	↑1 ↓3	↑0 ↓6	↑0 ↓7	↑**26** ↓**18**
All	(54)	↑**11** ↓**5**	↑**30** ↓**5**	↑**32** ↓**6**	↑**49** ↓**4**	↑**34** ↓**11**	↑11 ↓29	↑3 ↓49	↑0 ↓54	↑**170** ↓**163**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑**2** ↓**0**	↑**1** ↓**0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑3 ↓10
5	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑5 ↓8
10	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑5 ↓7
20	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑**8** ↓**6**
40	(2)	↑0 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑**7** ↓**6**
80	(2)	↑0 ↓0	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑0 ↓0	↑0 ↓2	↑0 ↓2	↑**6** ↓**4**
C	(6)	↑1 ↓1	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**3** ↓**1**	↑0 ↓4	↑0 ↓6	↑0 ↓6	↑19 ↓19
S	(6)	↑0 ↓0	↑**5** ↓**0**	↑**6** ↓**0**	↑1 ↓2	↑3 ↓3	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑15 ↓22
		GDBG								
F_{1a}	(6)	↑**1** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑1 ↓4	↑0 ↓4	↑0 ↓6	↑**25** ↓**14**
F_{1b}	(6)	↑**1** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑1 ↓4	↑0 ↓5	↑0 ↓6	↑**26** ↓**15**
F_2	(6)	↑1 ↓1	↑**3** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑1 ↓1	↑0 ↓6	↑**28** ↓**8**
F_3	(6)	↑0 ↓0	↑**4** ↓**0**	↑**2** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**3** ↓**0**	↑**33** ↓**1**
F_4	(6)	↑**2** ↓**1**	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**1** ↓**0**	↑0 ↓4	↑0 ↓6	↑**24** ↓**11**
F_5	(6)	↑**1** ↓**0**	↑**6** ↓**0**	↑**4** ↓**2**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**38** ↓**3**
F_6	(6)	↑**1** ↓**0**	↑**3** ↓**1**	↑**3** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑1 ↓3	↑**31** ↓**5**
T_1	(7)	↑0 ↓0	↑**7** ↓**0**	↑**5** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**4** ↓**2**	↑2 ↓4	↑1 ↓5	↑**33** ↓**12**
T_2	(7)	↑2 ↓2	↑**4** ↓**0**	↑**4** ↓**2**	↑**7** ↓**0**	↑**7** ↓**0**	↑**4** ↓**2**	↑2 ↓2	↑1 ↓5	↑**31** ↓**13**
T_3	(7)	↑0 ↓0	↑**3** ↓**1**	↑**4** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**4** ↓**0**	↑**3** ↓**2**	↑1 ↓5	↑**29** ↓**9**
T_4	(7)	↑**2** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**5** ↓**2**	↑**3** ↓**2**	↑1 ↓4	↑**39** ↓**8**
T_5	(7)	↑0 ↓0	↑**4** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**3** ↓**1**	↑2 ↓5	↑**36** ↓**6**
T_6	(7)	↑**3** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**4** ↓**2**	↑**4** ↓**3**	↑2 ↓4	↑**37** ↓**9**
All	(54)	↑**8** ↓**3**	↑**40** ↓**1**	↑**43** ↓**4**	↑**48** ↓**3**	↑**48** ↓**4**	↑**27** ↓**17**	↑17 ↓26	↑8 ↓40	↑**239** ↓**98**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑6 ↓6
5	(2)	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑0 ↓2	↑0 ↓2	↑**10** ↓**6**
10	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑0 ↓1	↑0 ↓2	↑**10** ↓**3**
20	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑0 ↓2	↑**11** ↓**2**
40	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑**13** ↓**1**
80	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**13** ↓**0**
C	(6)	↑**1** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**4** ↓**2**	↑2 ↓3	↑0 ↓5	↑**29** ↓**11**
S	(6)	↑0 ↓1	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**3** ↓**2**	↑2 ↓4	↑**34** ↓**7**
		GDBG								
F_{1a}	(6)	↑0 ↓1	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**2**	↑1 ↓4	↑**32** ↓**7**
F_{1b}	(6)	↑0 ↓0	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**3** ↓**2**	↑1 ↓4	↑**34** ↓**6**
F_2	(6)	↑0 ↓0	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**39** ↓**0**
F_3	(6)	↑0 ↓0	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**0**
F_4	(6)	↑0 ↓0	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**39** ↓**0**
F_5	(6)	↑0 ↓0	↑**6** ↓**0**	↑**5** ↓**0**	↑0 ↓4	↑**2** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**30** ↓**5**
F_6	(6)	↑0 ↓0	↑**5** ↓**0**	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**39** ↓**0**
T_1	(7)	↑0 ↓0	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**5** ↓**2**	↑**5** ↓**2**	↑**43** ↓**5**
T_2	(7)	↑0 ↓1	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**3** ↓**2**	↑**41** ↓**4**
T_3	(7)	↑0 ↓0	↑**4** ↓**0**	↑**4** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**39** ↓**2**
T_4	(7)	↑0 ↓0	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**5** ↓**2**	↑**5** ↓**2**	↑**43** ↓**4**
T_5	(7)	↑0 ↓0	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**46** ↓**0**
T_6	(7)	↑0 ↓0	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**5** ↓**2**	↑**43** ↓**3**
All	(54)	↑1 ↓2	↑**50** ↓**0**	↑**49** ↓**0**	↑**47** ↓**4**	↑**49** ↓**2**	↑**50** ↓**2**	↑**41** ↓**9**	↑**31** ↓**17**	↑**318** ↓**36**

Table 4.24: CjDE vs *jDE* performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑10 ↓3**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑12 ↓1**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑13 ↓1**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑11 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
80	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
C	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓1**	*↑2 ↓3*	**↑34 ↓5**
S	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓0**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑37 ↓1**
F_{1b}	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑37 ↓1**
F_2	(6)	↑0 ↓0	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓0**
F_3	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
F_4	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓0**
F_5	(6)	**↑1 ↓0**	**↑4 ↓0**	**↑5 ↓0**	*↑0 ↓4*	*↑0 ↓6*	*↑1 ↓3*	↑2 ↓2	**↑4 ↓1**	**↑17 ↓16**
F_6	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓1**
T_1	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑39 ↓3**
T_2	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑45 ↓2**
T_3	(7)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑40 ↓3**
T_4	(7)	**↑1 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓2**	**↑43 ↓3**
T_5	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑44 ↓5**
T_6	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑42 ↓3**
All	(54)	↑1 ↓1	**↑47 ↓0**	**↑49 ↓0**	**↑43 ↓4**	**↑47 ↓6**	**↑48 ↓4**	**↑47 ↓3**	**↑44 ↓6**	**↑326 ↓24**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑12 ↓1**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
80	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
C	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑39 ↓1**
S	(6)	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓0**
		GDBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑39 ↓0**
F_{1b}	(6)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑35 ↓0**
F_2	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓0**
F_3	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓0**
F_4	(6)	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓0**
F_5	(6)	↑0 ↓0	**↑4 ↓0**	**↑3 ↓0**	*↑1 ↓4*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓5*	*↑8 ↓24*
F_6	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
T_1	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑41 ↓4**
T_2	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓2**
T_3	(7)	**↑1 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓1**	**↑40 ↓4**
T_4	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑4 ↓1**	**↑39 ↓5**
T_5	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑44 ↓5**
T_6	(7)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑42 ↓4**
All	(54)	**↑2 ↓0**	**↑48 ↓0**	**↑49 ↓0**	**↑46 ↓4**	**↑46 ↓6**	**↑48 ↓5**	**↑46 ↓4**	**↑44 ↓6**	**↑329 ↓25**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	*↑0 ↓1*	**↑10 ↓0**	**↑8 ↓0**	**↑5 ↓4**	*↑4 ↓5*	*↑3 ↓6*	*↑2 ↓7*	*↑2 ↓8*	**↑34 ↓31**
5	(10)	↑1 ↓1	**↑9 ↓0**	**↑9 ↓0**	**↑7 ↓2**	**↑6 ↓2**	↑5 ↓5	*↑3 ↓6*	*↑2 ↓7*	**↑42 ↓23**
10	(10)	**↑1 ↓0**	**↑9 ↓0**	**↑8 ↓1**	**↑8 ↓1**	**↑6 ↓2**	**↑6 ↓4**	*↑4 ↓5*	*↑3 ↓7*	**↑45 ↓20**
20	(10)	**↑1 ↓0**	**↑8 ↓0**	**↑9 ↓1**	**↑8 ↓0**	**↑8 ↓1**	**↑6 ↓4**	**↑5 ↓4**	*↑3 ↓6*	**↑48 ↓16**
40	(10)	↑1 ↓1	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓0**	**↑8 ↓0**	**↑6 ↓3**	**↑6 ↓4**	↑5 ↓5	**↑50 ↓15**
80	(10)	**↑1 ↓0**	**↑6 ↓1**	**↑8 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑6 ↓0**	**↑6 ↓4**	**↑5 ↓4**	**↑52 ↓10**
C	(6)	**↑3 ↓1**	**↑27 ↓0**	**↑26 ↓0**	**↑26 ↓2**	**↑20 ↓4**	**↑15 ↓12**	*↑11 ↓16*	*↑6 ↓21*	**↑134 ↓56**
S	(6)	↑2 ↓2	**↑23 ↓2**	**↑24 ↓4**	**↑20 ↓5**	**↑22 ↓6**	**↑17 ↓10**	**↑15 ↓14**	*↑14 ↓16*	**↑137 ↓59**
		GDBG								
F_{1a}	(30)	*↑1 ↓2*	**↑23 ↓0**	**↑27 ↓0**	**↑30 ↓0**	**↑25 ↓3**	**↑18 ↓10**	**↑16 ↓12**	*↑10 ↓17*	**↑150 ↓44**
F_{1b}	(30)	**↑3 ↓0**	**↑25 ↓0**	**↑27 ↓0**	**↑30 ↓0**	**↑24 ↓3**	**↑19 ↓10**	↑13 ↓13	*↑10 ↓17*	**↑151 ↓43**
F_2	(30)	*↑1 ↓2*	**↑25 ↓0**	**↑27 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑26 ↓3**	**↑18 ↓7**	**↑17 ↓12**	**↑173 ↓24**
F_3	(30)	**↑3 ↓0**	**↑23 ↓0**	**↑22 ↓1**	**↑27 ↓0**	**↑30 ↓0**	**↑26 ↓1**	**↑24 ↓5**	**↑21 ↓6**	**↑176 ↓13**
F_4	(30)	**↑3 ↓1**	**↑23 ↓1**	**↑26 ↓1**	**↑30 ↓0**	**↑30 ↓0**	**↑20 ↓1**	**↑18 ↓10**	**↑17 ↓12**	**↑167 ↓26**
F_5	(30)	**↑4 ↓1**	**↑23 ↓2**	**↑20 ↓3**	**↑13 ↓12**	**↑14 ↓13**	**↑18 ↓9**	**↑16 ↓8**	↑13 ↓13	**↑121 ↓61**
F_6	(30)	**↑3 ↓2**	**↑23 ↓1**	**↑23 ↓1**	**↑27 ↓0**	**↑30 ↓0**	**↑25 ↓1**	**↑23 ↓6**	**↑19 ↓9**	**↑173 ↓20**
T_1	(35)	↑1 ↓1	**↑26 ↓2**	**↑24 ↓2**	**↑30 ↓2**	**↑30 ↓4**	**↑23 ↓9**	**↑19 ↓14**	**↑17 ↓15**	**↑170 ↓49**
T_2	(35)	**↑4 ↓3**	**↑27 ↓1**	**↑28 ↓3**	**↑32 ↓3**	**↑30 ↓2**	**↑26 ↓5**	**↑22 ↓8**	**↑17 ↓14**	**↑186 ↓39**
T_3	(35)	**↑3 ↓0**	**↑21 ↓1**	**↑21 ↓1**	**↑30 ↓3**	**↑30 ↓3**	**↑25 ↓4**	**↑23 ↓8**	**↑19 ↓13**	**↑172 ↓33**
T_4	(35)	**↑4 ↓1**	**↑34 ↓0**	**↑35 ↓0**	**↑31 ↓1**	**↑30 ↓4**	**↑26 ↓6**	**↑19 ↓12**	*↑15 ↓16*	**↑194 ↓40**
T_5	(35)	*↑0 ↓3*	**↑30 ↓0**	**↑34 ↓0**	**↑32 ↓2**	**↑32 ↓2**	**↑28 ↓4**	**↑23 ↓8**	**↑20 ↓14**	**↑199 ↓33**
T_6	(35)	**↑6 ↓0**	**↑27 ↓0**	**↑30 ↓0**	**↑32 ↓1**	**↑30 ↓4**	**↑24 ↓7**	**↑22 ↓11**	**↑19 ↓14**	**↑190 ↓37**
All	(270)	**↑23 ↓11**	**↑215 ↓6**	**↑222 ↓10**	**↑233 ↓19**	**↑224 ↓29**	**↑184 ↓57**	**↑154 ↓91**	**↑127 ↓123**	**↑1382 ↓346**

result of Mendes and Mohais [2005] that using a small sub-population size of six is an effective parameter setting, but concluded that only one Brownian individual should be used, rather than two.

Three DynDE-based algorithms were introduced and evaluated in this chapter. The first algorithm, competitive population evaluation (CPE), allows sub-populations to compete for function evaluations based on performance. The second algorithm, reinitialisation midpoint check (RMC), is aimed at improving the exclusion process of DynDE. The third algorithm, competitive differential evolution (CDE) is a combination of CPE and RMC.

A scalability study was conducted on DynDE, CPE, RMC and CDE to determine how the algorithms scale with respect to the number of dimensions, change period, severity of changes, change types, and underlying function in dynamic environments. The study found that DynDE and RMC exhibit similar scaling behaviour, while CPE and CDE exhibit similar scaling behaviour. An increase in the number of dimensions resulted in an increase in offline error for all the algorithms, with CPE and CDE yielding lower offline errors as the number of dimensions is increased.

CPE and CDE scaled better than DynDE and RMC as the period between changes in the environment is decreased, with the exception of extremely low change periods where the algorithms performed similarly. The offline errors of all the algorithms were found to decrease as the change period is increased, since more function evaluations are available between changes. At high change periods DynDE and RMC were found to perform better than CPE and CDE.

An increase in change severity resulted in lower offline errors for all algorithms. CPE and CDE were found to scale better than DynDE and RMC as the severity of changes in the environment increases. Experimental results showed that the improved performance of CPE and CDE vs DynDE and RMC on high change severity problems is due to CPE and CDE locating optima faster, after changes in the environment.

The underlying function used in the benchmark was shown to have a strong influence on the performance of the optimisation algorithm. The underlying function generally determines the magnitude of the difference in offline error of CPE and CDE vs DynDE and RMC. The influence of the type of changes in the environment was found to be strongly linked to the underlying function. Generally, the change type influences the point at which

DynDE and RMC start outperforming CPE and CDE due to an increased change period.

An analysis was conducted to determine whether CPE, RMC and CDE yielded statistically significantly better results than DynDE. CPE was found to perform significantly better than DynDE in the vast majority of experiments. The improvements were found to be especially pronounced in the lower change period experiments. RMC was found to perform better than DynDE in a small number of experiments localised in the low dimensional environments. CDE performed significantly better than DynDE in the majority of experiments and was shown to perform better than both RMC and CPE.

Experimental evidence was presented to show that CDE yielded a lower offline error than DynDE, because CDE's error values decreased faster after a change in the environment than those of DynDE. The diversity of the individuals in the search space differed very little between CDE and DynDE, but the average diversity per sub-population of CDE decreased faster than that of DynDE.

CDE and DynDE were compared in terms of the average error immediately before changes in the environment to investigate the possibility that CDE merely exploits the offline error performance measure. An analysis of the experimental results showed that, while improvements in terms of the average error immediately before changes were less pronounced than when using the offline error, CDE still performed significantly better than DynDE in the majority of the experiments.

An extensive comparison to *jDE*, a state-of-the-art optimisation algorithm for DOPs, found that CDE performed significantly better than *jDE* in the majority of investigated environments, especially at lower change periods. A comparison of CDE to the reported results of other algorithms found in the literature showed that CDE compares favourably with state-of-the-art algorithms.

The general applicability of the approaches suggested in this chapter was investigated by incorporating the novel components of CDE into the *jDE* algorithm. The experimental results proved that *jDE* was improved by the incorporation of the new approaches, but that the resultant algorithm was still inferior to CDE.

The evaluations in this chapter did not focus on the effect of the number of optima in the environment. The following chapter focuses on this important consideration.

Chapter 5

UNKNOWN NUMBER OF OPTIMA

The previous chapter presented CDE, an extension of DynDE, consisting of two algorithms: CPE and RMC. This chapter proposes a novel extension to CDE, called DynPopDE. DynPopDE dynamically spawns and removes sub-populations to adapt the number of sub-populations to an appropriate value. DynPopDE is aimed at dynamic optimisation problems where the number of optima is unknown or fluctuating. DynPopDE is evaluated and compared to DynDE and CDE on problems where the number of optima is constant but unknown, and on problems where the number of optima fluctuates over time.

5.1 Introduction

Section 2.5.2 identified the number of optima as a factor that contributes to the hardness of an optimisation problem. The presence of local optima makes it possible that the optimisation algorithm may become trapped, which could lead to sub-optimal results. The locations of local optima are generally not of interest when optimising a static environment, and algorithms typically avoid wasting function evaluations on local optima.

The literature study in Chapter 3 found that several optimisation algorithms for dynamic environments aim to track local optima, as these may become global optima after a change in the environment [Blackwell and Branke, 2006], [Mendes and Mohais, 2005].

The use of multiple populations has become a common strategy for tracking multiple optima. However, the number of optima in a fitness landscape is typically not known, which means the parameter controlling the number of sub-populations must be manually tuned for specific problems. The problem of finding the appropriate number of sub-populations was ignored in the previous chapter, and is consequently the focus of this chapter.

This chapter investigates two types of environments. The first is when the number of optima in the environment is unknown, but constant. The second is when the number of optima is unknown and changes over time. The majority of current research on solving DOPs focuses on problems where the number of optima is unknown but does not change over time. This is the case with the popular moving peaks benchmark (MPB) and the generalised dynamic benchmark generator (GDBG). Environments in which the number of optima explicitly changes over time have only recently received attention by the optimisation community [du Plessis and Engelbrecht, 2012a].

The phrase "varying number of optima" is often used by researchers, but this generally refers only to the fact that some optima are obscured by others for a brief period during the algorithm's execution, and not to the number of optima fluctuating during the optimisation process. An environment in which the number of optima fluctuates over time has been included in the benchmark set of the IEEE WCCI-2012 competition on evolutionary computation for dynamic optimisation problems [Li *et al.*, 2011]. The extension applied to the MPB to allow fluctuating numbers of peaks, which was discussed in Section 2.5.3.1, is used in this chapter to simulate dynamic environments in which the number of optima fluctuates.

This chapter presents a new extension to CDE, called DynPopDE, which is discussed in Section 5.2. This algorithm maintains a dynamic set of sub-populations that can shrink or expand to suit the environment. DynPopDE spawns sub-populations when it finds that limited improvement in fitness is being made by the current sub-populations (Section 5.2.1). A sub-population is removed when it converges to an optimum that is already occupied by another sub-population (Section 5.2.2). DynPopDE thus avoids the problem of tuning the number of sub-populations, by adapting the parameter during the optimisation process. It is interesting to note that several earlier algorithms aimed at dynamic optimisation supported the adaptation of the number of sub-populations (e.g. SOS [Branke *et al.*,

2000], [Branke, 2002] and MGA [Ursem, 2000]). Later, more effective algorithms made use of a constant number of sub-populations (e.g. MPSO [Blackwell and Branke, 2006], DynDE [Mendes and Mohais, 2005] and *jDE* [Brest *et al.*, 2009]). DynPopDE revisits the domain of using a dynamic number of sub-populations by building on the competitive population evaluation process of CDE.

The remainder of the this chapter is structured as follows: Section 5.2 presents the DynPopDE algorithm. The research questions of this chapter are outlined and experimentally investigated in Section 5.3. These investigations include a scalability study of DynDE, CDE and DynPopDE on environments with various numbers of optima and environments where the number of optima fluctuates over time. DynPopDE is compared to other algorithms found in the literature in Section 5.4 and conclusions are drawn and presented in Section 5.5.

5.2 Dynamic Population Differential Evolution

This section introduces a novel approach specifically designed to address problems with unknown or fluctuating numbers of optima. Dynamic population differential evolution (DynPopDE) is an extension of CDE consisting of two components, namely spawning new populations (refer to Section 5.2.1) and removing populations (refer to Section 5.2.2). The complete DynPopDE algorithm is given in Section 5.2.3.

5.2.1 Spawning Populations

Section 5.3 will show that, even if the number of peaks is known, using a number of sub-populations equal to the number of peaks is not always an effective strategy. When the number of peaks is unknown, choosing the number of sub-populations to use would be, at best, an educated guess. It is therefore suggested that the number of sub-populations should not be a parameter of the algorithm, but that sub-populations should be spawned as needed. The question that must be answered is: When should new populations be spawned?

CDE is based on allocating processing time and function evaluations to populations based on a performance value, $\mathcal{P}_k(t)$ (refer to equation (4.2) in Section 4.3). Sub-populations

are evolved in sequence until all sub-populations converge to their respective optima in the fitness landscape. It is proposed here that an appropriate time to introduce an additional sub-population is when little further improvement in fitness is found for all the current sub-populations. Introducing new sub-populations earlier would be contrary to the competing population approach of CDE, where inferior sub-populations are deliberately excluded from the evolution process so that optima can be discovered earlier.

A detection scheme is suggested to indicate when evolution has reached a point of little or no improvement in fitness of current sub-populations. This point is referred to as stagnation in the context of this thesis. DynPopDE introduces a new population of randomly created individuals when stagnation is detected, so that previously undiscovered optima can be located. CDE calculates the performance value of a sub-population, P_k, as the product of its current relative fitness, $R_k(t)$, and the improvement that was made in fitness during the previous generation, $\Delta F(\vec{x}_{best,k}, t)$ (see equation (4.2)). The suggestion made in this section is that a meaningful indicator of stagnation is when all the current sub-populations receive a zero improvement of fitness after their last respective function evaluations. Let $n_k(t)$ be the number of current sub-populations. Define a function, $S(t)$, that is true if stagnation has occurred, as follows:

$$S(t) = \begin{cases} true \text{ if} (\Delta F(\vec{x}_{best,k}, t) = 0) \ \forall \ k \in \{1, \ldots, n_k(t)\} \\ false \text{ otherwise} \end{cases} \tag{5.1}$$

$\Delta F(\vec{x}_{best,k}, t)$ is as defined in equation (4.3). Note that equation (5.1) does not guarantee that stagnation of all populations has permanently occurred, since the fitness of some of the sub-populations may improve in subsequent generations. However, when $S(t)$ is true, it does mean that none of the sub-populations has improved its fitness in the previous generation. Since function evaluations are not effectively used by the current sub-populations, a new sub-population should be created which may lead to locating more optima in the problem space.

After each generation, DynPopDE determines the value of $S(t)$. If $S(t) = true$ then a new randomly generated sub-population is added to the set of sub-populations. The sub-population spawning approach allows DynPopDE to commence with only a single sub-population and adapt to an appropriate number of sub-populations. The number of

sub-populations is thus removed as a parameter from the algorithm.

5.2.2 Removing Populations

The previous section explained how new populations are spawned when necessary. However, it is possible that equation (5.1) detects stagnation incorrectly since it cannot guarantee that stagnation for all sub-populations has occurred indefinitely. More sub-populations than necessary may consequently be created. Furthermore, in problems where the number of optima fluctuate, it would be desirable to remove superfluous sub-populations when the number of optima decreases. The algorithms should thus be able to detect and remove redundant sub-populations.

DynDE and CDE reinitialises a sub-population through exclusion when the spatial difference between the sub-population and a more fit sub-population drops below the exclusion threshold. A reasonable assumption that can be made is that, when redundant populations are present (i.e. more sub-populations exist than optima), these sub-populations will perpetually be reinitialised and will not converge to specific optima, since exclusion prevents the convergence of more than one sub-population to the same optimum. Consequently, redundant sub-populations can be detected by finding sub-populations that are successively reinitialised through exclusion without reaching a point of apparent stagnation (i.e. there are no more optima available for the sub-population to occupy).

A sub-population, P_k, will be discarded when it is flagged for reinitialisation due to exclusion and if

$$\Delta F(\vec{x}_{best,k}, t) \neq 0 \tag{5.2}$$

The exclusion process, as given in Algorithm 13, is thus further adapted to remove sub-populations as detailed in Algorithm 14 (assuming a function minimisation problem).

This approach may, at times, incorrectly classify populations as redundant in that it does not guarantee the removal of populations only when the number of populations outnumber the number of optima. A sub-population may be discarded for converging to an optimum occupied by another sub-population even when undiscovered optima still exist in the problem space. However, no optimum is ever left unguarded through the discarding process since a sub-population is discarded only when it converges to an optimum already

Algorithm 14: DynPopDE Exclusion

```
for k_1 = 1, ..., n_k(t) do
    for k_2 = 1, ..., n_k(t) do
        if ||x⃗_{best,k_1} − x⃗_{best,k_2}||_2 < r_excl and k_1 ≠ k_2 then
            Let x⃗_mid = (x⃗_{best,k_1} + x⃗_{best,k_2})/2;
            if F(x⃗_mid) > F(x⃗_{best,k_1}) and F(x⃗_mid) > F(x⃗_{best,k_2}) then
                if F(x⃗_{best,k_1}) < F(x⃗_{best,k_2}) then
                    if ΔF(x⃗_{best,k_2}, t) = 0 then
                        Reinitialise population P_{k_2}
                    else
                        Discard population P_{k_2}
                    end
                else
                    if ΔF(x⃗_{best,k_1}, t) = 0 then
                        Reinitialise population P_{k_1}
                    else
                        Discard population P_{k_1}
                    end
                end
            end
        end
    end
end
```

occupied by another sub-population. No information about the fitness landscape is thus lost through discarding a sub-population. If all sub-populations have stagnated, a new sub-population will be created through the spawning process. The spawning and discarding components of DynPopDE thus reach a point of equilibrium, where sub-populations are created when function evaluations are not being used effectively by the current sub-populations, and where sub-populations are removed when converging to optima that are already guarded by other sub-populations.

5.2.3 DynPopDE Algorithm

The previous sections described the two components of DynPopDE. These components are incorporated into CDE to form the complete DynPopDE algorithm given in Algorithm 15.

Algorithm 15: DynPopDE Algorithm

while *termination criterion not met* **do**
  Allow the standard CDE algorithm to run for two generations;
  repeat
    for $k = 1, \ldots, n_k$ **do**
      Calculate the performance value, $\mathcal{P}_k(t)$
    end
    Select sub-population P_a such that $\mathcal{P}_a(t) = \min_{k=1,\ldots,n_k} \{\mathcal{P}_k(t)\}$;
    Evolve only sub-population P_a;
    $t = t + 1$;
    Perform Exclusion according to Algorithm 14;
    if $S(t) = true$ **then**
      Introduce new randomly created sub-population;
    end
  until *a change occurs in the environment*;
end

5.3 Experimental Results

This section details experiments conducted on DynDE, CDE and DynPopDE. Two main problem classes are investigated. The first is dynamic environments in which the number of optima is constant but unknown. The second class is dynamic optimisation problems in which the number of optima is fluctuating over time and unknown. Experiments were conducted to answer the following research questions:

1. *How many sub-populations should be used when the number of optima is known?* Before investigating problems with unknown numbers of optima, experiments are conducted on DynDE and CDE to determine which is the best strategy to follow when the number of optima is known. Two strategies are investigated, using the same number of sub-populations as the number of optima, and using a constant number of optima.

2. *How do DynDE, CDE and DynPopDE scale with respect to unknown numbers of optima, number of dimensions, and the change period?* A thorough scalability study was conducted on DynDE and CDE in the previous chapter with respect to the number of dimensions and change period. Experiments are conducted to determine if and how the number of optima affects the scalability behaviour of DynDE, CDE and DynPopDE.

3. *Does DynPopDE perform better than CDE and DynDE on problems where the number of optima is constant, but unknown?* The results of DynDE, CDE and DynPopDE are compared to determine whether DynPopDE is a more effective algorithm.

4. *How do DynDE, CDE and DynPopDE scale in terms of the maximum number of optima and percentage change in the environment on dynamic optimisation problems where the number of optima is unknown and fluctuates over time?* The scalability of the algorithms with regard to the maximum number of optima and percentage change in the number of optima in the problem space, is investigated. The effect of changing the number of dimensions and change period is also investigated.

5. *Is DynPopDE more effective than DynDE and CDE on dynamic problems where the*

number of optima is unknown and fluctuates over time? The results of DynDE, CDE and DynPopDE are compared to determine whether DynPopDE is a more effective algorithm.

6. *What is the convergence profile of DynPopDE?* The DynPopDE algorithm commences with a single sub-population. Fewer individuals are expected to affect diversity negatively, as a smaller sample of the fitness landscape is used by the optimisation algorithm. The convergence behaviours of DynDE, CDE and DynPopDE are compared in terms of diversity, current error, and resulting offline error to assist in explaining the trends observed in the analyses of research questions 2, 3, 4 and 5. The numbers of sub-populations employed by DynPopDE for unknown and fluctuating numbers of optima are investigated.

7. *Is the process of spawning and removing sub-populations as used in DynPopDE more effective than the process used in MPSO2?* This research question investigates whether other spawning and removal processes, namely the approaches used in MPSO2 [Blackwell, 2007], are a better choice for incorporation into CDE, and subsequently yield better results.

8. *Is DynPopDE more effective than CDE and DynDE on the set of environments used in Chapter 4?* DynPopDE is tested on the set of dynamic environments that were used in the previous chapter to compare DynDE to CDE. This analysis is included because several of the environments in this set contain large numbers of optima, and consist of functions which were not used for evaluation in research questions 2, 3, 4 and 5.

Questions 1 to 3 are investigated on benchmark problems where the number of optima is kept constant during the optimisation process. Questions 4 and 5 are investigated on problems where the number of optima fluctuates over time. Questions 6 and 7 are investigated on both sets of problems. The next section outlines the experimental procedure that was followed to answer the research questions listed above. Sections 5.3.2 to 5.3.9 respectively cover research questions 1 to 8.

5.3.1 Experimental Procedure

Chapter 2 concluded that the extended MPB (refer to Section 2.5.3) is ideal for studying the effect of number of optima in the environment on optimisation algorithms. This chapter investigates the performance of DynDE, CDE and DynPopDE on two types of dynamic environments. Firstly, environments with various numbers of optima are investigated. The experimental procedure for these investigations is discussed in Section 5.3.1.1. Secondly, environments where the numbers of optima fluctuate over time are investigated. The experimental procedure for fluctuating number of optima experiments is given in Section 5.3.1.2.

A stopping criterion of 60 changes in the environment was used for all experiments. The offline error was used as the performance measure for all environments. Experiments were repeated 30 times to facilitate drawing statistically valid conclusions from the results. Mann-Whitney U tests were used to test statistical significance when comparing algorithms.

5.3.1.1 Constant Numbers of Optima Experimental Procedure

Variations of the Scenario 2 settings of the MPB were used to simulate environments with various numbers of peaks. The set of variations is defined here as the n_p *standard set*. The n_p *standard set* consists of all combinations of settings given in Table 5.1. The number of dimensions, number of peaks, change period, and underlying function are varied. Five settings for number of dimensions, six settings for number of peaks, and eight settings for change period were investigated on both of the peak functions. The n_p *standard set* thus consists of a total of 480 environments.

5.3.1.2 Fluctuating Numbers of Optima Experimental Procedure

Various settings of the extended MPB were used to simulate environments where the number of peaks fluctuates over time. The set of variations is defined here as the $n_p(t)$ *standard set*. The $n_p(t)$ *standard set* consists of all combinations of settings given in Table 5.2. The number of dimensions, maximum number of peaks, percentage change in the number of peaks, change period, and underlying function are varied. Five settings

Table 5.1: The n_p standard set

Setting	**Value**
Number of dimensions (n_d)	5, 10, 25, 50, 100
Number of Peaks (n_p)	5, 10, 25, 50, 100, 200
Max and Min Peak height	[30,70]
Max and Min Peak width	[1.0,12.0]
Change period (Cp)	100, 500, 1000, 5000, 10000, 25000, 50000, 100000
Change severity (C_s)	1.0
Height severity	7.0
Width severity	1.0
Function (F)	Cone, Sphere
Correlation	0.0

for number of dimensions, five settings for maximum number of peaks, five settings for percentage change in the number of peaks, and eight settings for change period were investigated on both of the peak functions. A total of 2 000 environments are thus included in the $n_p(t)$ *standard set*.

Table 5.2: The $n_p(t)$ standard set

Setting	**Value**
Number of dimensions (n_d)	5, 10, 25, 50, 100
Maximum Number of Peaks (M_{n_p})	10, 25, 50, 100, 200
Percentage Change in n_p (pc)	5, 10, 20, 40, 80
Max and Min Peak height	[30,70]
Max and Min Peak width	[1.0,12.0]
Change period (Cp)	100, 500, 1000, 5000, 10000, 25000, 50000, 100000
Change severity (C_s)	1.0
Height severity	7.0
Width severity	1.0
Function (F)	Cone, Sphere
Correlation	0.0

5.3.2 Research Question 1

How many sub-populations should be used when the number of optima is known?

The motivation for employing multiple populations is that multiple (preferably all) optima should be tracked. This enables the algorithm to locate the position of a new global optimum quickly when a change in the environment results in a different optimum becoming the global optimum. However, algorithms generally do not have information regarding the number of optima that are present in a fitness landscape. This, potentially, makes the number of sub-populations a very important parameter of an optimisation algorithm aimed at DOPs.

The number of optima are assumed to be known in advance for the purpose of this research question. The number of sub-populations used by DynDE and CDE can thus be set equal to the number of optima. These two algorithms were compared to DynDE and CDE where each used 10 sub-populations (the setting used in Chapter 4). DynDE and CDE are referred to as DynDE10 and CDE10 when using 10 sub-populations. This baseline comparison can show how beneficial it is to know the number of optima in advance.

DynDE, CDE, DynDE10 and CDE10 were compared using the n_p standard set which was defined in Section 5.3.1.1. CDE performed statistically significantly better than DynDE in 231 of the 480 experiments and worse in only 47 cases (a full analysis is available in Appendix C). This is consistent with the conclusion of Chapter 4 that CDE is a superior algorithm for solving DOPs than DynDE.

A more interesting result was found when comparing DynDE and CDE to their respective counterparts that always used 10 sub-populations, i.e. DynDE10 and CDE10. DynDE10 performed better than DynDE in 253 of the 480 experiments and worse in only 131 cases. CDE10 performed better than CDE in 236 cases and performed worse in only 126 cases. The algorithms that used 10 sub-populations thus performed better more often than the algorithms which used a number of sub-populations equal to the number of peaks that were present in the environment. This result is counter-intuitive since, for most of the environments that were investigated, DynDE10 and CDE10 had fewer sub-populations than the number of peaks, and could consequently not track all optima.

The explanation for the fact that DynDE10 and CDE10 performed better, overall,

than DynDE and CDE can be found by noting that an increase in the number of sub-populations results in a decreased number of generations that can be performed between the changes that occur in the environments after a set number of function evaluations. For example, DynDE with 200 sub-populations would result in about 1 500 function evaluations per generation. A dynamic environment that has a change period of 5 000 function evaluations would thus allow the algorithm to perform less than four DynDE generations between changes in the environment. Such a small number of generations would naturally result in poor performance by DynDE. The problem is less applicable to CDE after the second generation, when only the best-performing sub-population is evolved at any given time. However, a significant number of function evaluations would still be wasted during the initial performance calculation phase.

A comparative performance analysis between CDE and CDE10 was conducted to support the above explanation. Table 5.3 gives a count of the number of times CDE10 performed significantly better (indicated by $\uparrow$) than CDE for each of the settings of number of peaks. The number of times that CDE performed better than CDE10 is indicated by the $\downarrow$ symbol. Data was stratified into categories for different values of number of dimensions and change period. Cases where CDE10 performed better more often than CDE are shaded and printed in boldface, while cases where CDE performed better more often than CDE10 are printed in italics. A similar analysis was performed for DynDE10 versus DynDE, and is included in Appendix C, since results similar to the CDE10 versus CDE comparison were found.

Table 5.3 shows that the cases where CDE performed better than CDE10 were isolated to high change period experiments. This is consistent with the explanation that large numbers of sub-populations result in too few generations that are performed between changes in the environment. More generations can be performed as the period between changes is increased, which resulted in CDE eventually outperforming CDE10 as CDE can track more optima. Figures 5.1 and 5.2 give the offline errors of DynDE, CDE, DynDE10 and CDE10 on the conical peak function in five dimensions for various settings of number of peaks, with a change period of 5 000 and 100 000, respectively. Figure 5.1 shows that, at a low change period, CDE10 performed better than CDE when a large number of peaks was present in the environment. At high change periods, of which Figure 5.2 is an example,

enough function evaluations were available for CDE to locate and track optima effectively.

Table 5.3 shows that, as the number of dimensions was increased, CDE10 started to outperform CDE at increasingly higher change periods. As the number of dimensions was increased, the number of generations, required by an algorithm to locate optima, also increased (refer to the discussion on the influence of number of dimensions on the scalability of DynDE and CDE in Section 4.6.4.3). The algorithms using large sub-populations did not perform an adequate number of generations to locate optima in high dimensional environments. Figures 5.3 and 5.4 give the offline errors of DynDE, CDE, DynDE10 and CDE10 in 25 dimensions, with a change period of 5 000 and 100 000, while Figures 5.5 and 5.6 give the same information in 100 dimensions. A comparison of Figures 5.3 and 5.4 with Figures 5.1 and 5.2 shows a considerable comparative improvement of CDE10 with respect to CDE, with CDE10 performing significantly better than CDE in all cases with a high number of peaks in the low change period experiments. CDE still performed better than CDE10 in the high change period experiments. This trend changed in the 100 dimensional experiments where CDE10 always yielded results similar to or better than those of CDE. These results confirm that using a large number of sub-populations is not appropriate in high dimensions, even when a large number of function evaluations are available. It is better to use a smaller number of sub-populations and consequently perform more generations between changes in the environment.

This research question investigated whether using a number of sub-populations equal to the number of optima in the environment is effective. The experimental results showed that the cases where the algorithms used the same number of sub-populations as number of optima only yielded superior results in experiments where the number of optima was small, or when the change period was large in low dimensional environments. Ten sub-populations proved to be a more beneficial setting in the majority of cases that were investigated. This section thus showed that, even when the number of optima in the environment is known, this information is generally not beneficial.

The following sections assume no knowledge regarding the number of optima. DynDE and CDE will use 10 sub-populations at all times, as this value has been shown to produce reasonable results over a wide range of environments. Note that, for the rest of this thesis, the “10” suffix will be omitted from DynDE10 and CDE10.

Table 5.3: CDE10 vs CDE performance analysis

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓15
10	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓1	↑1 ↓1
25	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑6 ↓10
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑5 ↓10
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑6 ↓9
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑8 ↓8
C	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑5 ↓1**	↑1 ↓3	↑0 ↓5	↑0 ↓5	↑0 ↓4	↑0 ↓5	↑14 ↓25
S	(6)	**↑3 ↓1**	**↑4 ↓1**	**↑4 ↓1**	↑1 ↓4	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑12 ↓28
All	(12)	**↑7 ↓2**	**↑8 ↓2**	**↑9 ↓2**	↑2 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓9	↑0 ↓11	↑26 ↓53
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓11
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	**↑7 ↓6**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	**↑8 ↓6**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	**↑11 ↓4**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑0 ↓2	**↑12 ↓4**
C	(6)	**↑2 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓0**	↑2 ↓2	↑0 ↓4	↑0 ↓4	↑0 ↓4	↑15 ↓17
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑2 ↓1**	↑1 ↓3	↑0 ↓5	**↑23 ↓14**
All	(12)	**↑6 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑7 ↓1**	**↑6 ↓3**	↑2 ↓5	↑1 ↓7	↑0 ↓9	**↑38 ↓31**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓11
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑12 ↓3**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑13 ↓2**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑14 ↓2**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑15 ↓1**
C	(6)	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓1**	**↑2 ↓1**	↑1 ↓3	↑0 ↓4	**↑22 ↓11**
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑32 ↓8**
All	(12)	**↑7 ↓1**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓1**	**↑8 ↓2**	**↑6 ↓2**	**↑5 ↓4**	↑4 ↓5	**↑54 ↓19**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓12
10	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑13 ↓2**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑14 ↓1**
100	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑3 ↓0**	**↑3 ↓1**	**↑5 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑2 ↓1**	↑1 ↓2	**↑26 ↓7**
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑32 ↓8**
All	(12)	**↑7 ↓1**	**↑7 ↓2**	**↑9 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓1**	**↑6 ↓2**	**↑5 ↓3**	**↑58 ↓15**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓0	↑0 ↓1	↑0 ↓7
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑13 ↓1**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑15 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓0**	**↑2 ↓1**	**↑28 ↓6**
S	(6)	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑32 ↓2**
All	(12)	**↑7 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑7 ↓0**	**↑6 ↓2**	**↑60 ↓8**
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	↑0 ↓7	↑0 ↓9	↑0 ↓9	↑0 ↓7	↑0 ↓8	↑0 ↓6	↑0 ↓4	↑0 ↓6	↑0 ↓56
10	(10)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓1	**↑2 ↓1**
25	(10)	**↑6 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑7 ↓2**	**↑7 ↓3**	**↑5 ↓4**	↑3 ↓6	↑3 ↓7	**↑51 ↓22**
50	(10)	**↑8 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓2**	**↑7 ↓3**	**↑5 ↓3**	↑4 ↓5	↑3 ↓6	**↑55 ↓19**
100	(10)	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑8 ↓1**	**↑8 ↓2**	**↑7 ↓3**	**↑5 ↓4**	↑4 ↓5	**↑61 ↓15**
200	(10)	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓2**	**↑7 ↓3**	**↑7 ↓3**	↑5 ↓5	**↑67 ↓13**
C	(30)	**↑15 ↓2**	**↑19 ↓5**	**↑22 ↓5**	**↑16 ↓5**	**↑14 ↓10**	↑10 ↓11	↑6 ↓12	↑3 ↓16	**↑105 ↓66**
S	(30)	**↑19 ↓5**	**↑20 ↓4**	**↑20 ↓4**	**↑17 ↓7**	**↑16 ↓8**	**↑14 ↓8**	**↑13 ↓10**	↑12 ↓14	**↑131 ↓60**
All	(60)	**↑34 ↓7**	**↑39 ↓9**	**↑42 ↓9**	**↑33 ↓12**	**↑30 ↓18**	**↑24 ↓19**	↑19 ↓22	↑15 ↓30	**↑236 ↓126**

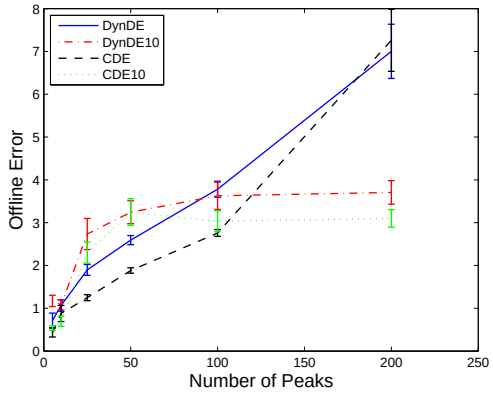

Figure 5.1: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 5 000 for various settings of number of peaks in 5 dimensions.

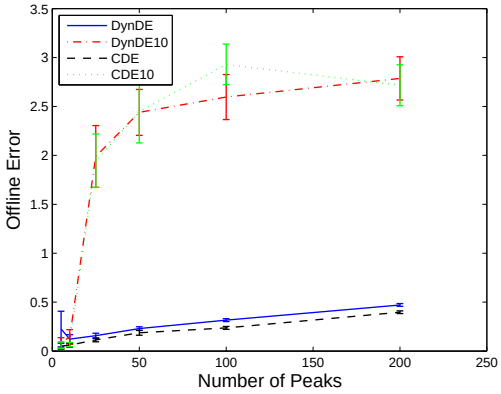

Figure 5.2: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 100 000 for various settings of number of peaks in 5 dimensions.

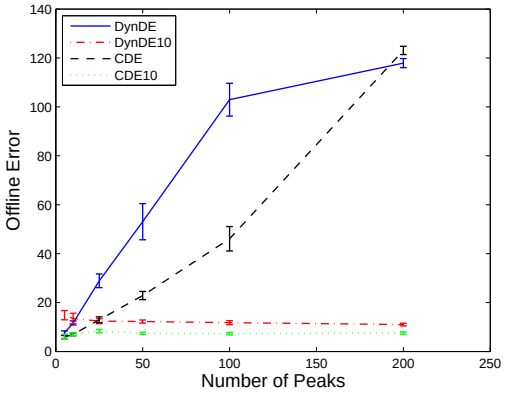

Figure 5.3: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 5 000 for various settings of number of peaks in 25 dimensions.

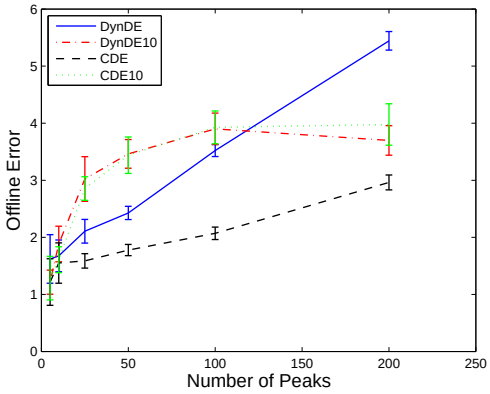

Figure 5.4: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 100 000 for various settings of number of peaks in 25 dimensions.

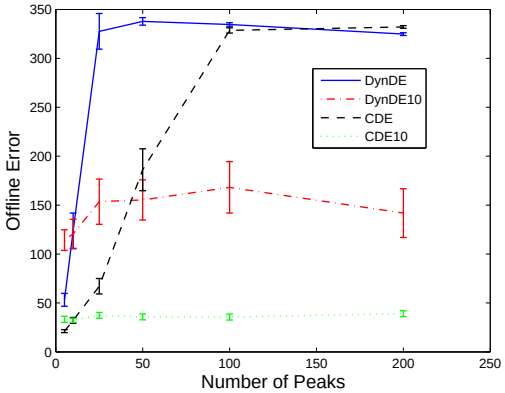

Figure 5.5: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 5 000 for various settings of number of peaks in 100 dimensions.

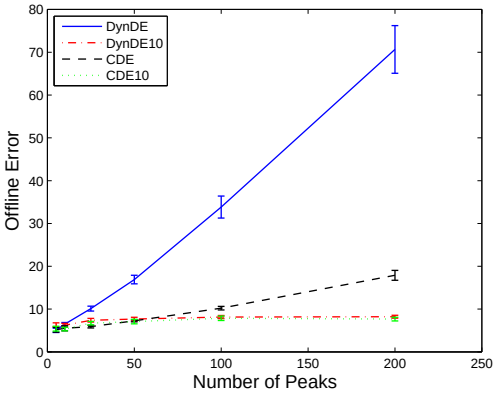

Figure 5.6: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 100 000 for various settings of number of peaks in 100 dimensions.

5.3.3 Research Question 2

How do DynDE, CDE and DynPopDE scale with respect to unknown numbers of optima, number of dimensions, and change period?

Chapter 4 investigated the scalability of DynDE and CDE with respect to change severity, underlying function, number of dimensions, change type, and change period. This research question investigates scalability with respect to the number of optima in the environment. DynPopDE, the extension to CDE proposed in this chapter, is included in the scalability study. DynPopDE was also executed on the n_p standard set of experiments, following the experimental procedure described in Section 5.3.1.1.

The first trend that can be observed is that increasing the number of optima tended to result in an increased offline error for the algorithms. Figure 5.7 shows that the offline error of DynDE, CDE and DynPopDE increased as the number of optima was increased, when using the conical peak function in five dimensions with a change period of 5 000. Several exceptions to this general trend were, however, observed. For example, consider Figure 5.12 (the 25 dimensional case of the experiment shown in Figure 5.7), where the offline error of DynDE decreased as the number of peaks was increased; the offline error of CDE and DynPopDE initially increased, but then decreased when the number of peaks exceeded 25.

An increased number of optima in an MPB environment can thus be either detrimental or beneficial to an optimisation algorithm. Large numbers of peaks make it impossible for an algorithm to track all peaks. This may have the result that a global optimum is not discovered, which explains the increasing offline errors visible in Figure 5.7. Conversely, an increased number of peaks raises the average function value of an MPB environment [Moser and Chiong, 2010], and makes it easier for an algorithm to locate local optima. A consequence of this effect is the decreasing offline errors visible in Figure 5.12.

CDE generally performed better than DynDE, while DynPopDE generally yielded lower offline errors than CDE. The magnitude of the difference in offline error was found to be related to the change period and the number of dimensions. The effect of the number of dimensions will be discussed first.

Figures 5.7 to 5.10 show the offline errors of DynDE, CDE and DynPopDE on the

conical peak function with a change period of 5 000 for different settings of the number of peaks in 5, 10, 50 and 100 dimensions, respectively. Similar trends were found when using the spherical peak function. CDE and DynPopDE generally performed considerably better than DynDE. A clear trend with regard to the comparative performance of CDE and DynPopDE over the increasing number of dimensions is visible. DynPopDE outperformed CDE by a wide margin in the five dimensional case (for all cases except when a small number of peaks was used). This margin is reduced in Figure 5.8 where the number of dimensions was increased to 10, but DynPopDE still performed better than CDE in the majority of cases. The difference in performance between CDE and DynPopDE narrowed and eventually disappeared as the number of dimensions was increased to 50 and 100 dimensions. Figures 5.9 and 5.10 show overlapping confidence intervals for the offline errors of CDE and DynPopDE.

DynPopDE thus scaled better than CDE on low dimensional problems, but not on high dimensional problems. Research question six will endeavour to shed some light on this phenomenon.

Recall from Chapter 4 that a very low change period was found to be detrimental to the performance of DynDE and CDE. This is due, in part, to the small number of generations that can be performed by DynDE and CDE at low change periods. DynPopDE has an advantage over DynDE and CDE in that it commences with a single sub-population, and can consequently perform more generations than DynDE and CDE.

Figure 5.11 shows the offline errors of DynDE, CDE and DynPopDE on the conical peak function in 25 dimensions with a change period of 500 function evaluations. The advantage of commencing with a small number of sub-populations is clearly illustrated by the magnitude by which the offline error of DynPopDE was lower than that of CDE, and especially than that of DynDE. As the change period was increased to 5 000 function evaluations, as shown in Figure 5.12, the difference in performance between CDE and DynPopDE narrowed to the point of overlapping confidence intervals. The difference in the number of generations that CDE and DynPopDE performed decreased as enough function evaluations were available for CDE to enter the phase where only one sub-population was evolved per generation.

DynPopDE scaled better than CDE when the change period was increased to values

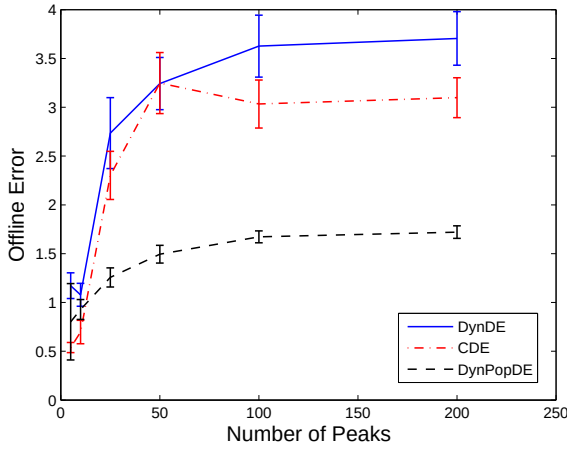

Figure 5.7: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 5 dimensions.

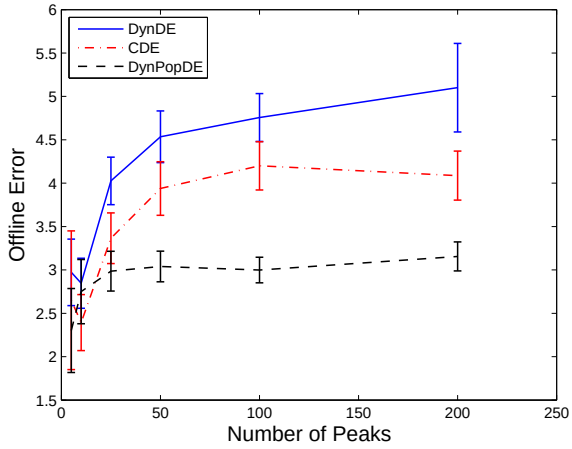

Figure 5.8: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 10 dimensions.

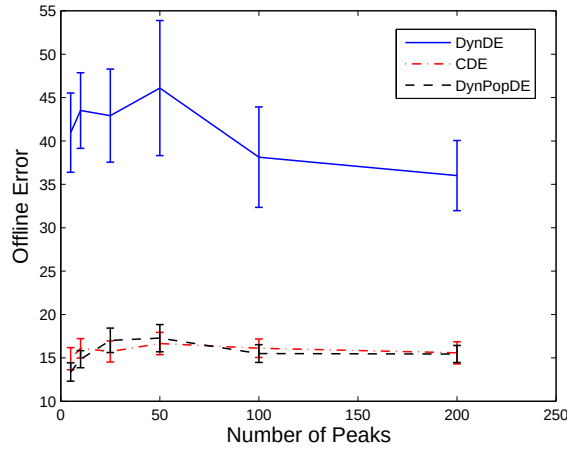

Figure 5.9: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 50 dimensions.

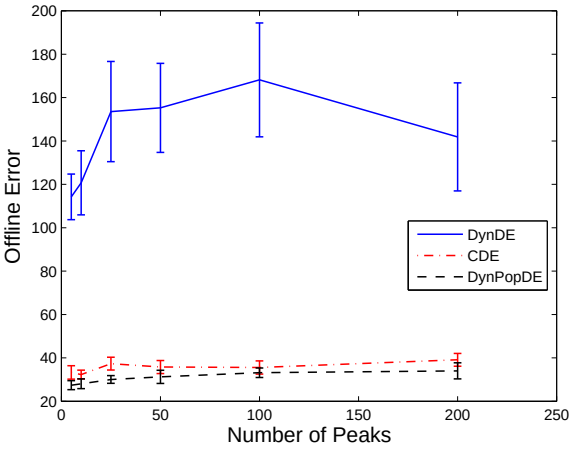

Figure 5.10: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 100 dimensions.

greater than 5 000. Figure 5.13 shows the performance of DynDE, CDE and DynPopDE when a change period of 25 000 was used. DynPopDE yielded considerably lower offline errors for larger numbers of peaks. The difference in performance increased as the change period was increased (refer to Figure 5.14). DynDE and CDE gave similar offline errors at a high change period. This is consistent with results found in Chapter 4 which showed that the difference in performance between DynDE and CDE diminishes as the change period is increased.

This research question compared the scalability of DynDE, CDE and DynPopDE relative to the number of optima in the environment. The experimental results indicate that CDE, generally, scaled better than DynDE, and that DynPopDE yielded further improved results. DynPopDE scaled better than CDE, especially in environments with a very low or high change period, and in low dimensions. The next research question investigates whether the differences between DynPopDE and the other two algorithms are statistically significant.

5.3.4 Research Question 3

Does DynPopDE perform better than CDE and DynDE on problems where the number of optima is constant, but unknown?

The previous section identified trends that suggest that DynPopDE is more scalable than DynDE and CDE in terms of number of optima in the environment. This research question aims to determine whether DynPopDE is indeed more effective than the other two algorithms.

A performance analysis was conducted on the results of DynDE, CDE and DynPopDE on the n_p standard set of experiments (defined in Section 5.3.1.1). The offline error of DynPopDE was compared to that of DynDE and CDE. The number of times that each algorithm performed statistically significantly better than the other (at a 95% confidence level according to a Mann-Whitney U test), was counted. Section 5.3.4.1 discusses the comparison of DynPopDE to DynDE, while Section 5.3.4.2 discusses the comparison of DynPopDE to CDE.

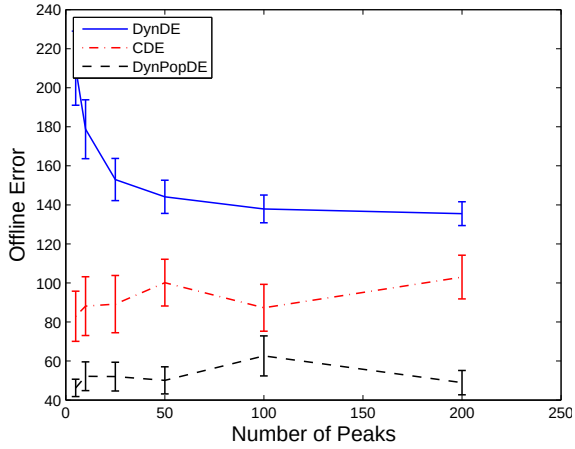

Figure 5.11: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of number of peaks in 25 dimensions.

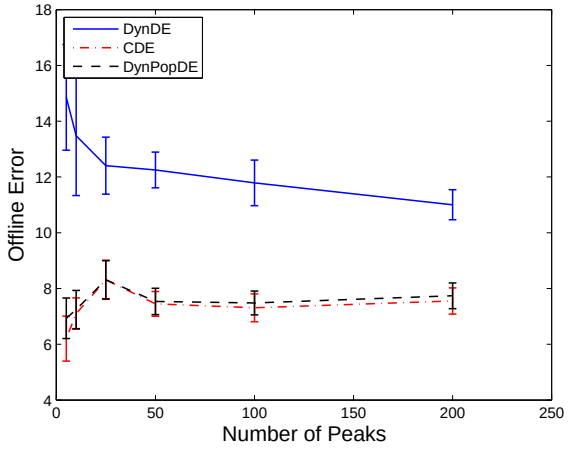

Figure 5.12: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 25 dimensions.

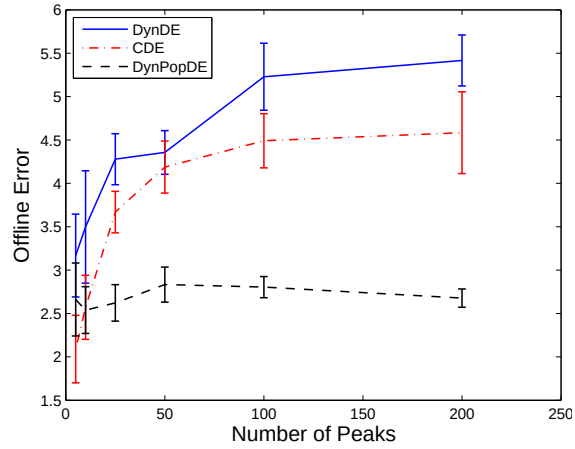

Figure 5.13: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 25 000 for various settings of number of peaks in 25 dimensions.

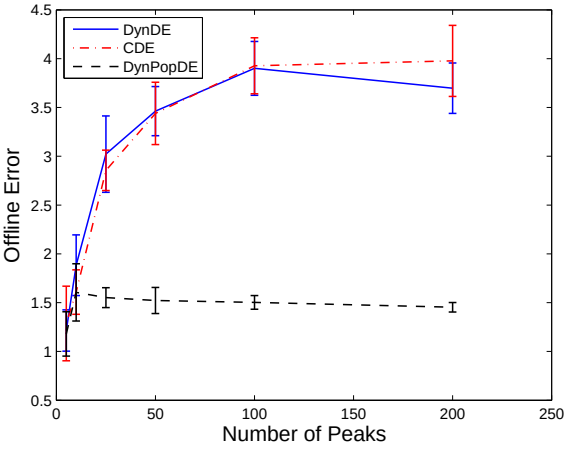

Figure 5.14: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 100 000 for various settings of number of peaks in 25 dimensions.

5.3.4.1 DynPopDE compared to DynDE on the n_p standard set

The result of the performance analysis of DynPopDE, compared to DynDE, is given in Table 5.4. The results are stratified into different settings for number of peaks, underlying function, change period and number of dimensions. Cells in which DynPopDE performed better than DynDE more often, are shaded and printed in boldface. Cells in which DynDE performed better than DynPopDE more often, are printed in italics. DynPopDE performed statistically significantly better than DynDE in 410 of the 480 experiments, and worse in only 21 experiments. The majority of the cases where DynDE performed better than DynPopDE were isolated in 100 dimensional experiments and this occurrence was more prevalent at lower change periods. Despite this, DynPopDE performed significantly better than DynDE in 60 of the 100 dimensional experiments, while DynDE outperformed DynPopDE in only 15 of the 100 dimensional experiments.

The average percentage improvement (API), calculated as in equation (4.6) on page 148, of DynPopDE over DynDE was calculated to determine how much better, on average, DynPopDE is than DynDE. The average percentage improvement of DynPopDE over DynDE over all experiments was found to be 42.04%. The APIs per dimension were found to be 52.46%, 45.07%, 49.12%, 49.66% and 13.90% for 5, 10, 25, 50 and 100 dimensions respectively. Greater improvements in offline error were thus found in lower dimensions. The APIs per change period were found to be 18.84%, 39.49%, 43.06%, 45.39%, 46.01%, 47.28%, 47.60% and 48.68% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations, respectively. The magnitude of the improvement over DynDE thus increased as the change period was increased. The APIs, in terms of the number of peaks, were found to be 34.36%, 26.09%, 43.97%, 47.54%, 49.96% and 50.35% for 5, 10, 25, 50, 100 and 200 peaks respectively. The percentage improvement increased as the number of peaks increased, with the exception of the 10 peak case. DynDE used 10 sub-populations and thus had an advantage when 10 peaks were present in the environment. This analysis indicates that DynPopDE is a more effective algorithm than DynDE over several settings of the number of peaks.

Table 5.4: DynPopDE vs DynDE performance analysis on the n_p standard set

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓1*	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑9 ↓2**
25	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
100	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓0**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑43 ↓2**
All	(12)	**↑10 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑10 ↓1**	**↑11 ↓1**	**↑10 ↓0**	**↑11 ↓0**	**↑12 ↓0**	**↑86 ↓2**
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑13 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
200	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
C	(6)	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑39 ↓0**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑10 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑87 ↓0**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑14 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
100	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
C	(6)	*↑1 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑40 ↓2**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑7 ↓2**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑11 ↓0**	**↑10 ↓0**	**↑88 ↓2**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
50	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
100	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
C	(6)	*↑0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑41 ↓2**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑6 ↓2**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑11 ↓0**	**↑89 ↓2**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑0 ↓0	**↑6 ↓5**
10	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑6 ↓5**
25	(2)	↑0 ↓0	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑7 ↓1**
50	(2)	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑11 ↓2**
100	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
C	(6)	*↑0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑40 ↓2**
S	(6)	**↑3 ↓0**	*↑2 ↓3*	*↑2 ↓3*	**↑3 ↓2**	↑2 ↓2	**↑3 ↓0**	↑2 ↓2	**↑3 ↓1**	**↑20 ↓13**
All	(12)	**↑3 ↓2**	**↑8 ↓3**	**↑8 ↓3**	**↑9 ↓2**	**↑8 ↓2**	**↑9 ↓0**	**↑8 ↓2**	**↑7 ↓1**	**↑60 ↓15**
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	**↑6 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑9 ↓0**	**↑8 ↓1**	**↑6 ↓0**	**↑64 ↓5**
10	(10)	**↑7 ↓0**	**↑9 ↓1**	**↑8 ↓1**	**↑6 ↓2**	**↑8 ↓2**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑55 ↓7**
25	(10)	**↑4 ↓0**	**↑8 ↓0**	**↑9 ↓1**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑66 ↓1**
50	(10)	**↑7 ↓1**	**↑9 ↓1**	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑10 ↓0**	**↑73 ↓3**
100	(10)	**↑6 ↓2**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓2**
200	(10)	**↑6 ↓3**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓3**
C	(30)	**↑9 ↓6**	**↑29 ↓0**	**↑30 ↓0**	**↑28 ↓0**	**↑29 ↓0**	**↑28 ↓0**	**↑27 ↓0**	**↑23 ↓0**	**↑203 ↓6**
S	(30)	**↑27 ↓0**	**↑26 ↓3**	**↑25 ↓3**	**↑26 ↓3**	**↑25 ↓3**	**↑26 ↓0**	**↑25 ↓2**	**↑27 ↓1**	**↑207 ↓15**
All	(60)	**↑36 ↓6**	**↑55 ↓3**	**↑55 ↓3**	**↑54 ↓3**	**↑54 ↓3**	**↑54 ↓0**	**↑52 ↓2**	**↑50 ↓1**	**↑410 ↓21**

5.3.4.2 DynPopDE compared to CDE on the n_p standard set

The result of the performance analysis of DynPopDE compared to CDE on the n_p standard set of experiments is given in Table 5.5. DynPopDE performed statistically significantly better than CDE in 323 of the 480 experiments and worse in only 49 cases. The scalability study which was conducted under the previous research question found that DynPopDE does not scale well to high dimensional problems. This observation is confirmed in the performance analysis in 100 dimensions, where CDE performed better, more often, than DynPopDE. CDE performed better more often than DynPopDE in 48 of the 96 experiment in 100 dimensions.

The average percentage improvement of DynPopDE over CDE over all experiments was found to be 28.72%. The APIs per dimension were found to be 43.79%, 43.71%, 36.95%, 31.40% and -12.25% for 5, 10, 25, 50 and 100 dimensions respectively. Considerable improvements of DynPopDE over CDE were thus found for most settings of dimension, but CDE outperformed DynPopDE in 100 dimensions. The APIs per change period were found to be 19.18%, 27.58%, 23.71%, 20.64%, 24.87%, 33.16%, 37.33% and 43.30% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations, respectively. The magnitude of the improvement over CDE thus increased as the change period was increased. The APIs, in terms of the number of peaks, were found to be 21.92%, 13.01%, 31.06%, 33.86%, 35.04% and 37.44% for 5, 10, 25, 50, 100 and 200 peaks respectively. The percentage improvement increased as the number of peaks increased, with the exception of the 10 peak case. CDE used 10 sub-populations and thus had an advantage when 10 peaks were present in the environment. DynPopDE was found to be superior to CDE on the n_p standard set of experiments, with the exception of very high dimensional environments, where CDE generally performed better.

5.3.4.3 Summary for Research Question 3

This research question investigated whether DynPopDE is a significantly better algorithm than DynDE and CDE on problems where the number of optima is unknown. DynPopDE was found, generally, to perform better than DynDE and CDE on a wide range of benchmark instances. The magnitudes of improvements were large, especially when a

Table 5.5: DynPopDE vs CDE performance analysis on the n_p standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
10	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	**↑5 ↓4**
25	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
50	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
100	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
200	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
C	(6)	**↑5 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑4 ↓1**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑29 ↓1**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑6 ↓0**	**↑5 ↓0**	**↑44 ↓3**
All	(12)	**↑11 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑9 ↓2**	**↑10 ↓1**	**↑9 ↓1**	**↑11 ↓0**	**↑9 ↓0**	**↑73 ↓4**
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓1**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
200	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
C	(6)	**↑4 ↓0**	**↑5 ↓0**	↑0 ↓0	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑28 ↓1**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑10 ↓0**	**↑11 ↓0**	**↑6 ↓0**	**↑9 ↓0**	**↑10 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓1**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓2**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
100	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
C	(6)	↑1 ↓1	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓1*	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑28 ↓3**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑7 ↓1**	**↑12 ↓0**	**↑12 ↓0**	**↑6 ↓1**	**↑9 ↓0**	**↑10 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓3**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑10 ↓1**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
100	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
C	(6)	*↑0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	↑0 ↓0	↑0 ↓0	**↑4 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑24 ↓3**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑6 ↓2**	**↑12 ↓0**	**↑12 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑10 ↓0**	**↑10 ↓1**	**↑10 ↓0**	**↑72 ↓3**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑3 ↓5*
10	(2)	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑2 ↓7*
25	(2)	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	*↑3 ↓7*
50	(2)	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑6 ↓7*
100	(2)	*↑0 ↓1*	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑5 ↓7*
200	(2)	↑1 ↓1	**↑2 ↓0**	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑7 ↓5**
C	(6)	*↑0 ↓2*	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	↑0 ↓0	**↑2 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑23 ↓2**
S	(6)	**↑2 ↓0**	*↑1 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑3 ↓36*
All	(12)	↑2 ↓2	**↑5 ↓4**	↑5 ↓5	*↑5 ↓6*	*↑0 ↓5*	*↑2 ↓5*	*↑3 ↓6*	*↑4 ↓5*	*↑26 ↓38*
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	**↑7 ↓0**	**↑9 ↓1**	**↑8 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑4 ↓1**	**↑5 ↓2**	**↑4 ↓0**	**↑47 ↓9**
10	(10)	**↑6 ↓0**	**↑7 ↓1**	**↑7 ↓1**	**↑4 ↓3**	**↑3 ↓2**	**↑3 ↓2**	**↑4 ↓1**	**↑3 ↓1**	**↑37 ↓11**
25	(10)	**↑5 ↓0**	**↑7 ↓1**	**↑7 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑9 ↓1**	**↑56 ↓7**
50	(10)	**↑7 ↓0**	**↑8 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑8 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑62 ↓7**
100	(10)	**↑5 ↓2**	**↑8 ↓0**	**↑7 ↓1**	**↑6 ↓1**	**↑7 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑60 ↓8**
200	(10)	**↑6 ↓3**	**↑8 ↓0**	**↑6 ↓0**	**↑7 ↓1**	**↑7 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑61 ↓7**
C	(30)	**↑10 ↓5**	**↑22 ↓0**	**↑18 ↓0**	**↑12 ↓2**	**↑12 ↓1**	**↑18 ↓1**	**↑20 ↓1**	**↑20 ↓0**	**↑132 ↓10**
S	(30)	**↑26 ↓0**	**↑25 ↓4**	**↑24 ↓5**	**↑23 ↓7**	**↑23 ↓6**	**↑23 ↓6**	**↑24 ↓6**	**↑23 ↓5**	**↑191 ↓39**
All	(60)	**↑36 ↓5**	**↑47 ↓4**	**↑42 ↓5**	**↑35 ↓9**	**↑35 ↓7**	**↑41 ↓7**	**↑44 ↓7**	**↑43 ↓5**	**↑323 ↓49**

large number of optima was present in the environment. DynPopDE has the advantage of commencing with a single sub-population, which makes it possible for the algorithm to perform more generations between changes than DynDE and CDE when a low change period is used. DynPopDE also yielded superior results for large change periods, but did not scale well to high dimensions.

5.3.5 Research Question 4

How do DynDE, CDE and DynPopDE scale in terms of the maximum number of optima and percentage change in the environment on dynamic optimisation problems where the number of optima is unknown and fluctuates over time?

The previous two research questions focused on the performance of DynDE, CDE and DynPopDE with respect to the number of optima in the environment. The focus of this research question is on dynamic environments in which the number of optima explicitly changes over time. The extended MPB benchmark was used to evaluate the performance of the three algorithms on the $n_p(t)$ standard set as described in Section 5.3.1.2. This section describes the trends observed in the comparative performance of DynDE, CDE and DynPopDE.

The effect of the maximum number of peaks on offline error was found to be dependent on the number of dimensions and the change period. Three distinct behaviour phases can be observed as the change period is increased. The first phase is shown in Figure 5.15, which depicts the offline errors of DynDE, CDE and DynPopDE for various settings of maximum number of peaks with a 5% change in the number of peaks on the conical peak function in 10 dimensions with a change period of 500 function evaluations. CDE generally performed better than DynDE during this phase, with DynPopDE generally performing better than CDE. The offline errors of all the algorithms decreased as the maximum number of peaks was increased. This is due to the fact that at these low change periods the average chances of locating optima are increased when more optima are present in the environment.

The second phase can be observed in Figure 5.16, which shows the same information as Figure 5.15, but with a change period of 5 000. This phase is characterised by high standard deviations in the experimental results (compare the confidence bars of Figure

5.16 to that of Figure 5.17), and, while DynDE generally yielded higher offline errors than CDE and DynPopDE during this phase, neither CDE nor DynPopDE was obviously better.

The third phase occurred at a higher change period than phase two, and is characterised by lower offline errors and a clearer differentiation between DynPopDE and the other algorithms. Phase three can be observed in Figures 5.17 and 5.18, which shows the offline errors when using change periods of 25 000 and 100 000, respectively. The offline error typically increased as the maximum number of peaks was increased during phase three. DynPopDE generally performed better than DynDE and CDE, especially for high values of maximum number of peaks. DynDE and CDE exhibited similar scaling behaviour during phase three, while CDE generally performed slightly better than DynDE.

The change period ranges in which each of the three phases occur, were found to be dependent on the number of dimensions. Phase one did not clearly exist in five dimensions, but appeared for change periods of less than 5 000, in dimensions higher than five. Phase two started directly after phase one and ended with increasingly higher change periods as the number of dimensions was increased. The transition from phase two to phase three occured at a change period of about 5 000 in 5 dimensions, 10 000 in 10 dimensions, 25 000 in 25 dimensions, and exhibited a broad transition period in 50 and 100 dimensions.

The percentage change in the number of peaks had a considerable impact on the offline error of all the algorithms, with the offline errors increasing as the percentage change was increased. Figure 5.19 shows the offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks. This figure illustrates the relative performance of the algorithms during phase one, and DynPopDE accordingly yielded lower offline errors. Figures 5.20 and 5.21 shows the offline errors at change periods of 5 000 and 25 000 respectively. This corresponds to phase two, and CDE performed the best for all of the larger percentage changes. DynPopDE performed the best for percentage changes lower than 40% in Figure 5.22, which shows the offline errors at a change period of 100 000, corresponding to phase three.

This research question investigated the scalability of DynDE, CDE and DynPopDE on DOPs with fluctuating numbers of peaks. The experimental results showed that Dyn-

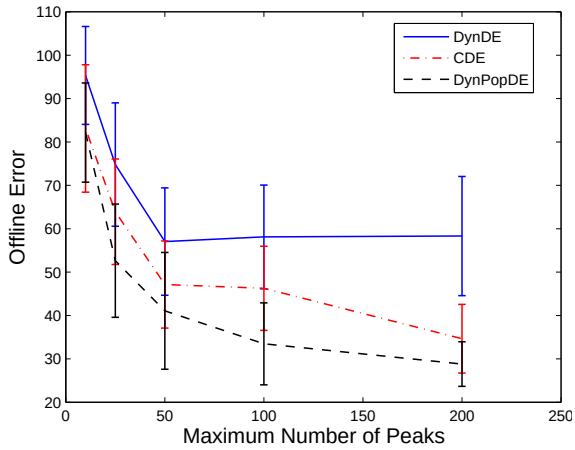

Figure 5.15: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

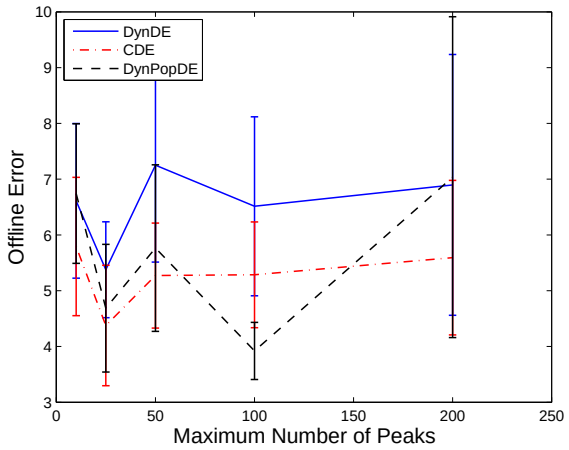

Figure 5.16: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

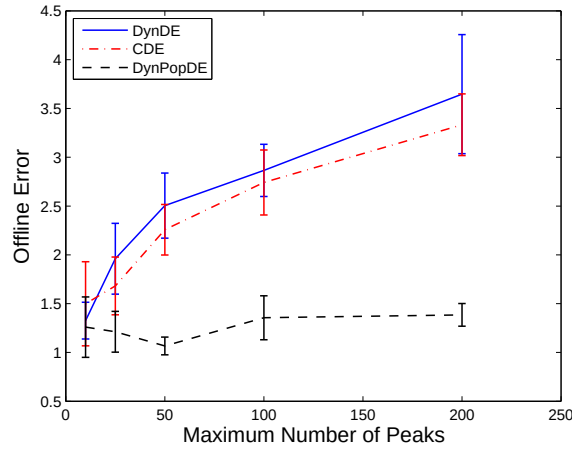

Figure 5.17: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 25 000 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

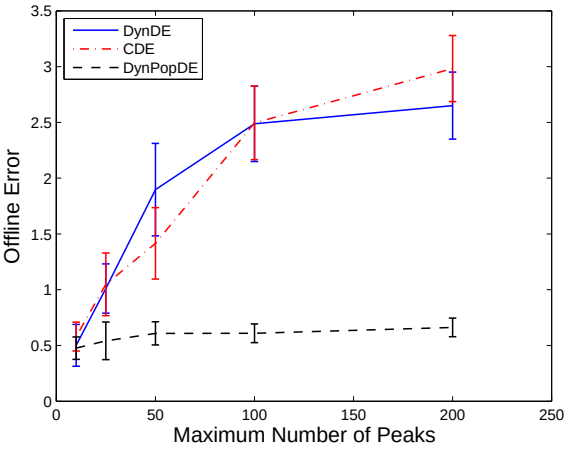

Figure 5.18: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 100 000 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

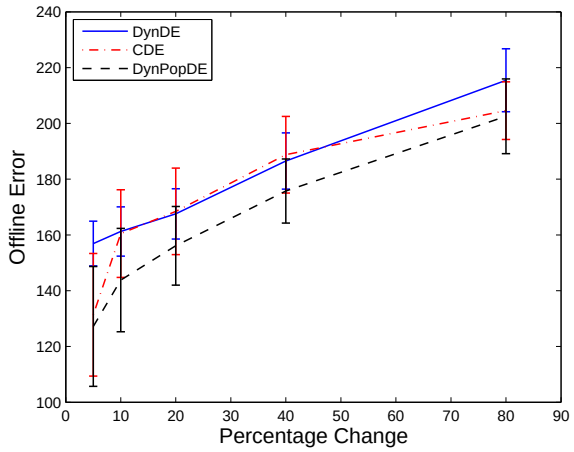

Figure 5.19: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

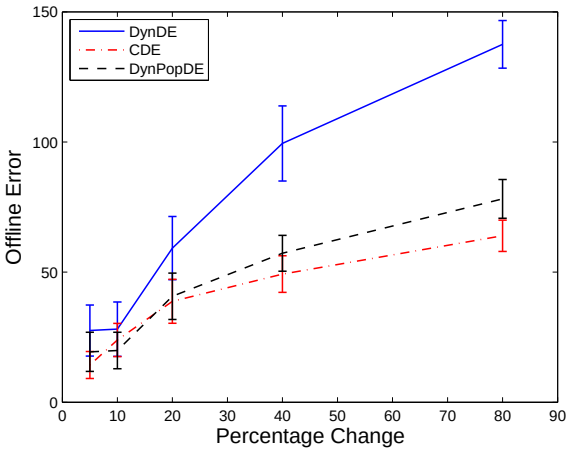

Figure 5.20: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

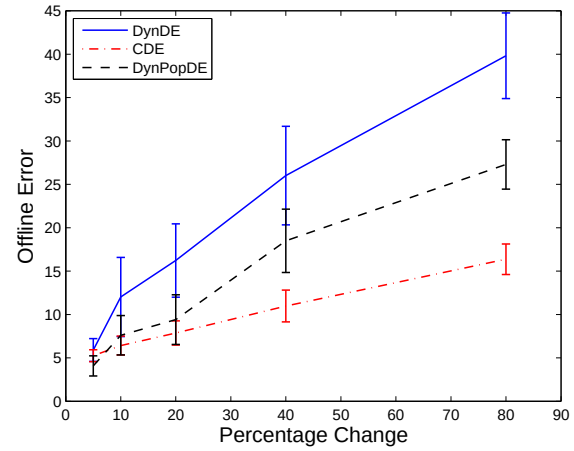

Figure 5.21: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 25 000 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

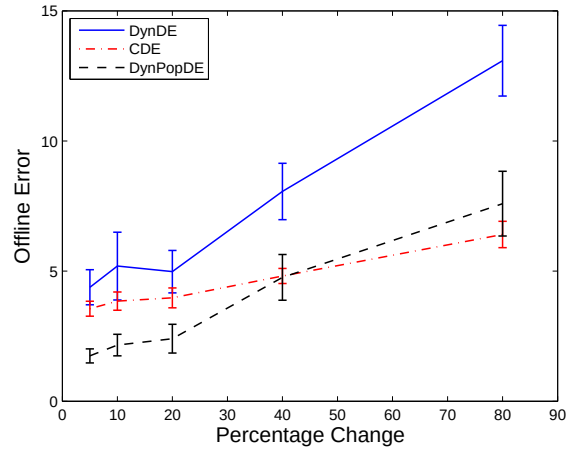

Figure 5.22: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 100 000 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

PopDE generally scaled better than DynDE and CPE at low change periods and at high change periods, but not at intermediate change periods. The focus of the next research question is to determine whether the differences between DynDE, CDE and DynPopDE are statistically significant.

5.3.6 Research Question 5

Is DynPopDE more effective than DynDE and CDE on dynamic problems where the number of optima is unknown and fluctuates over time?

The offline errors yielded by DynPopDE on the $n_p(t)$ standard set were compared to those yielded by DynDE and CDE. Table 5.6 gives the performance analysis of DynPopDE compared to DynDE, stratified with respect to change period, number of dimensions, peak function, percentage change in the number of peaks, and the maximum number of peaks. Each cell gives the number of times that DynPopDE performed better than DynDE (indicated by $\uparrow$), and the number of times that DynDE performed better than DynPopDE (indicated by $\downarrow$). Results that did not differ according to a Mann-Whitney U test at a 95% confidence level were not included in the respective totals.

DynPopDE performed significantly better than DynDE in 1 289 of the 2 000 experiments and worse in only 121 experiments. The cases where DynDE performed better than DynPopDE were concentrated in 100 dimensions and at lower change periods. DynDE and DynPopDE yielded similar results at a change period of 100.

The average percentage improvement of DynPopDE over DynDE over all experiments was found to be 29.01%. The APIs per dimension were found to be 45.01%, 33.41%, 30.63%, 26.98% and 9.04% for 5, 10, 25, 50 and 100 dimensions respectively. Larger improvements in offline error were thus found in lower dimensions. DynPopDE did result in a positive API in 100 dimensions, despite the fact that DynDE performed better more often than DynPopDE. The APIs per change period were found to be 1.28%, 15.18%, 19.38%, 30.75%, 32.84%, 35.19%, 44.17% and 53.33% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The improvement of DynPopDE thus increased as the change period was increased. The API values and the fact that DynPopDE performed significantly better than DynDE in the majority of experimental cases, leads to the conclusion that DynPopDE is superior to

Table 5.6: DynPopDE vs DynDE performance analysis on the $n_p(t)$ standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
M_{n_p} 10	(10)	↑*0* ↓*2*	↑**7** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**56** ↓**2**
25	(10)	↑0 ↓0	↑**6** ↓**0**	↑**5** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**60** ↓**0**
50	(10)	↑*0* ↓*1*	↑**7** ↓**1**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**64** ↓**2**
100	(10)	↑**1** ↓**0**	↑**4** ↓**0**	↑**4** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**59** ↓**0**
200	(10)	↑**2** ↓**0**	↑**6** ↓**1**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**65** ↓**1**
pc 5	(10)	↑**2** ↓**0**	↑**7** ↓**1**	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**67** ↓**1**
10	(10)	↑0 ↓0	↑**5** ↓**1**	↑**4** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**59** ↓**1**
20	(10)	↑1 ↓1	↑**3** ↓**0**	↑**2** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**56** ↓**1**
40	(10)	↑0 ↓0	↑**8** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**61** ↓**0**
80	(10)	↑*0* ↓*2*	↑**7** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**61** ↓**2**
C	(25)	↑*0* ↓*1*	↑**17** ↓**0**	↑**15** ↓**0**	↑**22** ↓**0**	↑**23** ↓**0**	↑**25** ↓**0**	↑**24** ↓**0**	↑**24** ↓**0**	↑**150** ↓**1**
S	(25)	↑**3** ↓**2**	↑**13** ↓**2**	↑**13** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**154** ↓**4**
All	(50)	↑3 ↓3	↑**30** ↓**2**	↑**28** ↓**0**	↑**47** ↓**0**	↑**48** ↓**0**	↑**50** ↓**0**	↑**49** ↓**0**	↑**49** ↓**0**	↑**304** ↓**5**
Set.	**Max**	**10 Dimensions**								
M_{n_p} 10	(10)	↑**2** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**8** ↓**0**	↑**5** ↓**1**	↑**6** ↓**0**	↑**5** ↓**2**	↑**44** ↓**3**
25	(10)	↑**2** ↓**1**	↑**8** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**57** ↓**3**
50	(10)	↑1 ↓1	↑**5** ↓**0**	↑**4** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**9** ↓**1**	↑**10** ↓**0**	↑**10** ↓**0**	↑**54** ↓**2**
100	(10)	↑**3** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**65** ↓**0**
200	(10)	↑**1** ↓**0**	↑**9** ↓**0**	↑**6** ↓**1**	↑**7** ↓**1**	↑**10** ↓**0**	↑**9** ↓**1**	↑**10** ↓**0**	↑**10** ↓**0**	↑**62** ↓**3**
pc 5	(10)	↑**5** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑**7** ↓**2**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**59** ↓**3**
10	(10)	↑0 ↓0	↑**7** ↓**0**	↑**4** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**7** ↓**1**	↑**9** ↓**0**	↑**9** ↓**0**	↑**51** ↓**1**
20	(10)	↑**1** ↓**0**	↑**8** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**8** ↓**1**	↑**8** ↓**1**	↑**9** ↓**0**	↑**9** ↓**1**	↑**52** ↓**3**
40	(10)	↑**2** ↓**1**	↑**7** ↓**0**	↑**8** ↓**0**	↑**5** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**9** ↓**1**	↑**56** ↓**2**
80	(10)	↑1 ↓1	↑**10** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**9** ↓**1**	↑**10** ↓**0**	↑**9** ↓**0**	↑**64** ↓**2**
C	(25)	↑**6** ↓**2**	↑**21** ↓**0**	↑**19** ↓**0**	↑**17** ↓**1**	↑**21** ↓**0**	↑**20** ↓**0**	↑**21** ↓**0**	↑**20** ↓**2**	↑**145** ↓**5**
S	(25)	↑**3** ↓**0**	↑**17** ↓**0**	↑**10** ↓**1**	↑**14** ↓**1**	↑**21** ↓**1**	↑**22** ↓**3**	↑**25** ↓**0**	↑**25** ↓**0**	↑**137** ↓**6**
All	(50)	↑**9** ↓**2**	↑**38** ↓**0**	↑**29** ↓**1**	↑**31** ↓**2**	↑**42** ↓**1**	↑**42** ↓**3**	↑**46** ↓**0**	↑**45** ↓**2**	↑**282** ↓**11**
Set.	**Max**	**25 Dimensions**								
M_{n_p} 10	(10)	↑0 ↓0	↑**9** ↓**0**	↑**10** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**59** ↓**0**
25	(10)	↑0 ↓0	↑**10** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**56** ↓**0**
50	(10)	↑1 ↓1	↑**8** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**59** ↓**1**
100	(10)	↑*0* ↓*1*	↑**7** ↓**0**	↑**9** ↓**1**	↑**8** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**57** ↓**2**
200	(10)	↑**2** ↓**0**	↑**4** ↓**0**	↑**7** ↓**1**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**8** ↓**1**	↑**9** ↓**0**	↑**51** ↓**3**
pc 5	(10)	↑**2** ↓**0**	↑**9** ↓**0**	↑**8** ↓**2**	↑**10** ↓**0**	↑**10** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**67** ↓**2**
10	(10)	↑**1** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**56** ↓**0**
20	(10)	↑0 ↓0	↑**6** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**49** ↓**0**
40	(10)	↑*0* ↓*1*	↑**9** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**9** ↓**0**	↑**9** ↓**0**	↑**54** ↓**2**
80	(10)	↑*0* ↓*1*	↑**7** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**8** ↓**1**	↑**9** ↓**0**	↑**56** ↓**2**
C	(25)	↑*0* ↓*2*	↑**18** ↓**0**	↑**25** ↓**0**	↑**22** ↓**0**	↑**21** ↓**0**	↑**18** ↓**0**	↑**25** ↓**0**	↑**24** ↓**0**	↑**153** ↓**2**
S	(25)	↑**3** ↓**0**	↑**20** ↓**0**	↑**17** ↓**2**	↑**17** ↓**0**	↑**11** ↓**0**	↑**15** ↓**1**	↑**22** ↓**1**	↑**24** ↓**0**	↑**129** ↓**4**
All	(50)	↑**3** ↓**2**	↑**38** ↓**0**	↑**42** ↓**2**	↑**39** ↓**0**	↑**32** ↓**0**	↑**33** ↓**1**	↑**47** ↓**1**	↑**48** ↓**0**	↑**282** ↓**6**
Set.	**Max**	**50 Dimensions**								
M_{n_p} 10	(10)	↑**1** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**62** ↓**0**
25	(10)	↑**1** ↓**0**	↑**4** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**55** ↓**0**
50	(10)	↑1 ↓1	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**7** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**10** ↓**0**	↑**56** ↓**2**
100	(10)	↑*1* ↓*3*	↑**7** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**6** ↓**1**	↑**8** ↓**0**	↑**50** ↓**4**
200	(10)	↑*0* ↓*2*	↑**6** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**44** ↓**2**
pc 5	(10)	↑**1** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**63** ↓**0**
10	(10)	↑**3** ↓**2**	↑**8** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**8** ↓**1**	↑**7** ↓**0**	↑**7** ↓**1**	↑**9** ↓**0**	↑**59** ↓**4**
20	(10)	↑*0* ↓*1*	↑**5** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**3** ↓**0**	↑**2** ↓**0**	↑**4** ↓**0**	↑**9** ↓**0**	↑**38** ↓**1**
40	(10)	↑0 ↓0	↑**6** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**52** ↓**0**
80	(10)	↑*0* ↓*3*	↑**6** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**55** ↓**3**
C	(25)	↑*0* ↓*4*	↑**12** ↓**0**	↑**21** ↓**0**	↑**25** ↓**0**	↑**21** ↓**0**	↑**16** ↓**0**	↑**17** ↓**1**	↑**23** ↓**0**	↑**135** ↓**5**
S	(25)	↑**4** ↓**2**	↑**20** ↓**0**	↑**21** ↓**0**	↑**22** ↓**0**	↑**16** ↓**1**	↑**12** ↓**0**	↑**17** ↓**0**	↑**20** ↓**0**	↑**132** ↓**3**
All	(50)	↑*4* ↓*6*	↑**32** ↓**0**	↑**42** ↓**0**	↑**47** ↓**0**	↑**37** ↓**1**	↑**28** ↓**0**	↑**34** ↓**1**	↑**43** ↓**0**	↑**267** ↓**8**
Set.	**Max**	**100 Dimensions**								
M_{n_p} 10	(10)	↑0 ↓0	↑*0* ↓*3*	↑*4* ↓*5*	↑5 ↓5	↑5 ↓5	↑4 ↓4	↑**3** ↓**0**	↑**8** ↓**0**	↑**29** ↓**22**
25	(10)	↑0 ↓0	↑*2* ↓*4*	↑4 ↓4	↑5 ↓5	↑**5** ↓**4**	↑**4** ↓**2**	↑**4** ↓**1**	↑**8** ↓**0**	↑**32** ↓**20**
50	(10)	↑*0* ↓*1*	↑*1* ↓*5*	↑*2* ↓*3*	↑**5** ↓**3**	↑**5** ↓**2**	↑**4** ↓**3**	↑**5** ↓**0**	↑**8** ↓**0**	↑**30** ↓**17**
100	(10)	↑*0* ↓*1*	↑*3* ↓*4*	↑**4** ↓**2**	↑**5** ↓**4**	↑**6** ↓**2**	↑**5** ↓**3**	↑**3** ↓**1**	↑**7** ↓**0**	↑**33** ↓**17**
200	(10)	↑*0* ↓*1*	↑**3** ↓**2**	↑3 ↓3	↑**5** ↓**2**	↑**5** ↓**3**	↑**4** ↓**2**	↑**5** ↓**1**	↑**5** ↓**1**	↑**30** ↓**15**
pc 5	(10)	↑0 ↓0	↑2 ↓2	↑**5** ↓**2**	↑**5** ↓**2**	↑**6** ↓**3**	↑**5** ↓**4**	↑**7** ↓**1**	↑**8** ↓**0**	↑**38** ↓**14**
10	(10)	↑0 ↓0	↑3 ↓3	↑**4** ↓**1**	↑**5** ↓**3**	↑**5** ↓**1**	↑**4** ↓**0**	↑**3** ↓**0**	↑**8** ↓**0**	↑**32** ↓**8**
20	(10)	↑0 ↓0	↑*0* ↓*4*	↑*2* ↓*4*	↑**5** ↓**4**	↑**5** ↓**2**	↑**2** ↓**1**	↑**1** ↓**0**	↑**8** ↓**0**	↑**23** ↓**15**
40	(10)	↑0 ↓0	↑*2* ↓*4*	↑*2* ↓*5*	↑5 ↓5	↑5 ↓5	↑**5** ↓**4**	↑**4** ↓**0**	↑**4** ↓**0**	↑**27** ↓**23**
80	(10)	↑*0* ↓*3*	↑*2* ↓*5*	↑*4* ↓*5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑**5** ↓**2**	↑**8** ↓**1**	↑**34** ↓**31**
C	(25)	↑0 ↓0	↑**9** ↓**0**	↑**17** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**21** ↓**0**	↑**18** ↓**0**	↑**18** ↓**0**	↑**133** ↓**0**
S	(25)	↑*0* ↓*3*	↑*0* ↓*18*	↑*0* ↓*17*	↑*0* ↓*19*	↑*1* ↓*16*	↑*0* ↓*14*	↑*2* ↓*3*	↑**18** ↓**1**	↑*21* ↓*91*
All	(50)	↑*0* ↓*3*	↑*9* ↓*18*	↑17 ↓17	↑**25** ↓**19**	↑**26** ↓**16**	↑**21** ↓**14**	↑**20** ↓**3**	↑**36** ↓**1**	↑**154** ↓**91**
Set.	**Max**	**All Dimensions**								
M_{n_p} 10	(50)	↑**3** ↓**2**	↑**32** ↓**3**	↑**35** ↓**5**	↑**34** ↓**5**	↑**36** ↓**5**	↑**32** ↓**5**	↑**37** ↓**0**	↑**41** ↓**2**	↑**250** ↓**27**
25	(50)	↑**3** ↓**1**	↑**30** ↓**4**	↑**33** ↓**4**	↑**36** ↓**6**	↑**35** ↓**5**	↑**34** ↓**2**	↑**42** ↓**1**	↑**47** ↓**0**	↑**260** ↓**23**
50	(50)	↑*3* ↓*5*	↑**28** ↓**6**	↑**30** ↓**3**	↑**39** ↓**3**	↑**37** ↓**3**	↑**37** ↓**4**	↑**41** ↓**0**	↑**48** ↓**0**	↑**263** ↓**24**
100	(50)	↑5 ↓5	↑**29** ↓**4**	↑**31** ↓**3**	↑**41** ↓**4**	↑**38** ↓**2**	↑**37** ↓**3**	↑**38** ↓**2**	↑**45** ↓**0**	↑**264** ↓**23**
200	(50)	↑**5** ↓**3**	↑**28** ↓**3**	↑**29** ↓**5**	↑**39** ↓**3**	↑**39** ↓**3**	↑**34** ↓**4**	↑**38** ↓**2**	↑**40** ↓**1**	↑**252** ↓**24**
pc 5	(50)	↑**10** ↓**0**	↑**31** ↓**3**	↑**35** ↓**5**	↑**41** ↓**4**	↑**45** ↓**3**	↑**39** ↓**4**	↑**46** ↓**1**	↑**47** ↓**0**	↑**294** ↓**20**
10	(50)	↑**4** ↓**2**	↑**30** ↓**4**	↑**28** ↓**1**	↑**38** ↓**3**	↑**37** ↓**2**	↑**35** ↓**1**	↑**39** ↓**1**	↑**46** ↓**0**	↑**257** ↓**14**
20	(50)	↑2 ↓2	↑**22** ↓**4**	↑**22** ↓**4**	↑**33** ↓**4**	↑**30** ↓**3**	↑**29** ↓**2**	↑**34** ↓**0**	↑**46** ↓**1**	↑**218** ↓**20**
40	(50)	↑2 ↓2	↑**32** ↓**4**	↑**34** ↓**5**	↑**36** ↓**5**	↑**34** ↓**5**	↑**35** ↓**5**	↑**38** ↓**0**	↑**39** ↓**1**	↑**250** ↓**27**
80	(50)	↑*1* ↓*10*	↑**32** ↓**5**	↑**39** ↓**5**	↑**41** ↓**5**	↑**39** ↓**5**	↑**36** ↓**6**	↑**39** ↓**3**	↑**43** ↓**1**	↑**270** ↓**40**
C	(125)	↑*6* ↓*9*	↑**77** ↓**0**	↑**97** ↓**0**	↑**111** ↓**1**	↑**111** ↓**0**	↑**100** ↓**0**	↑**105** ↓**1**	↑**109** ↓**2**	↑**716** ↓**13**
S	(125)	↑**13** ↓**7**	↑**70** ↓**20**	↑**61** ↓**20**	↑**78** ↓**20**	↑**74** ↓**18**	↑**74** ↓**18**	↑**91** ↓**4**	↑**112** ↓**1**	↑**573** ↓**108**
All	(250)	↑**19** ↓**16**	↑**147** ↓**20**	↑**158** ↓**20**	↑**189** ↓**21**	↑**185** ↓**18**	↑**174** ↓**18**	↑**196** ↓**5**	↑**221** ↓**3**	↑**1289** ↓**121**

DynDE on problems where the number of optima fluctuates over time.

The same performance analysis that was conducted for DynPopDE versus DynDE was repeated for DynPopDE versus CDE. The results are given in Table 5.7. The analysis shows that the average offline errors of DynPopDE and CDE did not differ statistically significantly in the majority of experimental environments. DynPopDE did, however, perform statistically significantly better than CDE in 719 of the 2 000 experiments, and performed worse in only 249 experiments.

The three phases that were identified in Section 5.3.5 can be observed in Table 5.7 by looking at the clusters of shaded cells (which indicate settings where DynPopDE performed better more often than CDE). Phase one, in which DynPopDE performed better than CDE at low change periods, is visible in the change period of 100 and 500 columns. Phase two, in which CDE typically performed better than DynPopDE, is visible in the intermediate range of change periods which broadens in higher dimensions. Phase three, in which DynPopDE typically performed better than CDE at high change periods, is most visible in low dimensions.

The average percentage improvement of DynPopDE over CDE over all experiments was found to be 12.45%. The APIs per dimension were found to be 37.69%, 23.18%, 6.11%, -3.42% and -1.29% for 5, 10, 25, 50 and 100 dimensions respectively. DynPopDE was thus inferior on high dimensions, but by a small margin. The APIs per change period, were found to be 1.82%, 9.01%, 5.79%, 1.85%, 5.84%, 11.22%, 26.04% and 38.06% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The improvement of DynPopDE thus increased as the function evaluations was increased. The APIs per setting of maximum number of peaks are 9.25%, 14.38%, 14.92%, 14.19% and 9.54% for values 5, 10, 25, 50, 100 and 200, respectively. The APIs, with regard to percentage change in the number of peaks, were found to be 22.78%, 17.25%, 11.63%, 6.84% and 3.78% for values 5%, 10%, 20%, 40% and 80%, respectively. Large improvements over CDE were thus made for the smaller percentage changes, with only minor improvements for large percentage changes.

The results presented in this section support the conclusion that, in general, DynPopDE is a more effective optimisation algorithm than DynDE and CDE on problems where the number of optima fluctuates over time. DynPopDE performed statistically

Table 5.7: DynPopDE vs CDE performance analysis on the $n_p(t)$ standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
M_{n_p} 10	(10)	↑0 ↓1	↑3 ↓0	↑3 ↓0	↑4 ↓0	↑6 ↓0	↑4 ↓0	↑6 ↓0	↑4 ↓0	↑30 ↓1
25	(10)	↑1 ↓0	↑1 ↓0	↑0 ↓1	↑7 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑49 ↓1
50	(10)	↑0 ↓1	↑0 ↓2	↑2 ↓0	↑7 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑49 ↓3
100	(10)	↑0 ↓1	↑2 ↓0	↑1 ↓0	↑9 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑52 ↓1
200	(10)	↑2 ↓1	↑2 ↓0	↑3 ↓0	↑8 ↓0	↑9 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑54 ↓1
pc 5	(10)	↑1 ↓1	↑4 ↓1	↑4 ↓1	↑8 ↓0	↑8 ↓0	↑8 ↓0	↑9 ↓0	↑9 ↓0	↑51 ↓3
10	(10)	↑1 ↓0	↑2 ↓1	↑2 ↓0	↑9 ↓0	↑10 ↓0	↑9 ↓0	↑9 ↓0	↑8 ↓0	↑50 ↓1
20	(10)	↑1 ↓0	↑0 ↓0	↑0 ↓0	↑9 ↓0	↑9 ↓0	↑8 ↓0	↑9 ↓0	↑9 ↓0	↑45 ↓0
40	(10)	↑0 ↓0	↑2 ↓0	↑1 ↓0	↑5 ↓0	↑9 ↓0	↑9 ↓0	↑9 ↓0	↑9 ↓0	↑44 ↓0
80	(10)	↑0 ↓3	↑0 ↓0	↑2 ↓0	↑4 ↓0	↑9 ↓0	↑10 ↓0	↑10 ↓0	↑9 ↓0	↑44 ↓3
C	(25)	↑1 ↓2	↑0 ↓0	↑2 ↓0	↑14 ↓0	↑21 ↓0	↑21 ↓0	↑21 ↓0	↑20 ↓0	↑100 ↓2
S	(25)	↑2 ↓2	↑8 ↓2	↑7 ↓1	↑21 ↓0	↑24 ↓0	↑23 ↓0	↑25 ↓0	↑24 ↓0	↑134 ↓5
All	(50)	↑3 ↓4	↑8 ↓2	↑9 ↓1	↑35 ↓0	↑45 ↓0	↑44 ↓0	↑46 ↓0	↑44 ↓0	↑234 ↓7
Set.	**Max**	**10 Dimensions**								
M_{n_p} 10	(10)	↑2 ↓0	↑6 ↓0	↑1 ↓0	↑3 ↓0	↑6 ↓0	↑4 ↓1	↑6 ↓0	↑5 ↓0	↑33 ↓1
25	(10)	↑1 ↓0	↑4 ↓0	↑0 ↓0	↑3 ↓0	↑4 ↓0	↑8 ↓0	↑10 ↓0	↑10 ↓0	↑40 ↓0
50	(10)	↑2 ↓0	↑3 ↓0	↑0 ↓1	↑1 ↓3	↑5 ↓0	↑9 ↓1	↑10 ↓0	↑10 ↓0	↑40 ↓5
100	(10)	↑1 ↓1	↑4 ↓1	↑3 ↓0	↑4 ↓1	↑5 ↓1	↑8 ↓0	↑10 ↓0	↑10 ↓0	↑45 ↓4
200	(10)	↑2 ↓0	↑3 ↓0	↑1 ↓0	↑1 ↓1	↑4 ↓2	↑5 ↓1	↑9 ↓0	↑9 ↓0	↑34 ↓4
pc 5	(10)	↑2 ↓0	↑4 ↓1	↑2 ↓1	↑6 ↓0	↑8 ↓0	↑9 ↓0	↑9 ↓0	↑9 ↓0	↑49 ↓2
10	(10)	↑2 ↓0	↑2 ↓0	↑0 ↓0	↑4 ↓3	↑8 ↓0	↑8 ↓1	↑9 ↓0	↑9 ↓0	↑42 ↓4
20	(10)	↑2 ↓0	↑3 ↓0	↑0 ↓0	↑1 ↓0	↑5 ↓0	↑8 ↓1	↑10 ↓0	↑9 ↓0	↑38 ↓1
40	(10)	↑1 ↓1	↑5 ↓0	↑2 ↓0	↑0 ↓1	↑2 ↓1	↑6 ↓0	↑9 ↓0	↑9 ↓0	↑34 ↓3
80	(10)	↑1 ↓0	↑6 ↓0	↑1 ↓0	↑1 ↓1	↑1 ↓2	↑3 ↓1	↑8 ↓0	↑8 ↓0	↑29 ↓4
C	(25)	↑6 ↓1	↑10 ↓0	↑1 ↓0	↑2 ↓2	↑7 ↓3	↑15 ↓0	↑21 ↓0	↑20 ↓0	↑82 ↓6
S	(25)	↑2 ↓0	↑10 ↓1	↑4 ↓1	↑10 ↓3	↑17 ↓0	↑19 ↓3	↑24 ↓0	↑24 ↓0	↑110 ↓8
All	(50)	↑8 ↓1	↑20 ↓1	↑5 ↓1	↑12 ↓5	↑24 ↓3	↑34 ↓3	↑45 ↓0	↑44 ↓0	↑192 ↓14
Set.	**Max**	**25 Dimensions**								
M_{n_p} 10	(10)	↑1 ↓0	↑3 ↓0	↑2 ↓0	↑1 ↓1	↑1 ↓1	↑3 ↓2	↑6 ↓0	↑5 ↓1	↑22 ↓5
25	(10)	↑0 ↓0	↑6 ↓0	↑2 ↓0	↑1 ↓2	↑1 ↓3	↑3 ↓1	↑5 ↓0	↑6 ↓0	↑24 ↓6
50	(10)	↑1 ↓0	↑5 ↓0	↑3 ↓0	↑2 ↓1	↑2 ↓1	↑3 ↓1	↑5 ↓0	↑9 ↓0	↑30 ↓3
100	(10)	↑0 ↓0	↑4 ↓0	↑4 ↓0	↑2 ↓0	↑1 ↓3	↑2 ↓3	↑5 ↓2	↑7 ↓0	↑25 ↓8
200	(10)	↑2 ↓2	↑1 ↓0	↑4 ↓0	↑1 ↓2	↑3 ↓4	↑2 ↓5	↑4 ↓4	↑6 ↓1	↑23 ↓18
pc 5	(10)	↑2 ↓1	↑4 ↓0	↑4 ↓0	↑4 ↓1	↑6 ↓0	↑7 ↓1	↑8 ↓0	↑8 ↓0	↑43 ↓3
10	(10)	↑1 ↓0	↑4 ↓0	↑2 ↓0	↑3 ↓0	↑2 ↓1	↑4 ↓2	↑7 ↓1	↑9 ↓0	↑32 ↓4
20	(10)	↑0 ↓1	↑3 ↓0	↑1 ↓0	↑0 ↓1	↑0 ↓0	↑2 ↓0	↑6 ↓0	↑8 ↓1	↑20 ↓3
40	(10)	↑0 ↓0	↑4 ↓0	↑3 ↓0	↑0 ↓1	↑0 ↓4	↑0 ↓2	↑2 ↓1	↑5 ↓0	↑14 ↓8
80	(10)	↑1 ↓0	↑4 ↓0	↑5 ↓0	↑0 ↓3	↑0 ↓7	↑0 ↓7	↑2 ↓4	↑3 ↓1	↑15 ↓22
C	(25)	↑0 ↓2	↑2 ↓0	↑4 ↓0	↑0 ↓5	↑1 ↓8	↑2 ↓8	↑8 ↓3	↑12 ↓1	↑29 ↓27
S	(25)	↑4 ↓0	↑17 ↓0	↑11 ↓0	↑7 ↓1	↑7 ↓4	↑11 ↓4	↑17 ↓3	↑21 ↓1	↑95 ↓13
All	(50)	↑4 ↓2	↑19 ↓0	↑15 ↓0	↑7 ↓6	↑8 ↓12	↑13 ↓12	↑25 ↓6	↑33 ↓2	↑124 ↓40
Set.	**Max**	**50 Dimensions**								
M_{n_p} 10	(10)	↑2 ↓0	↑6 ↓0	↑3 ↓0	↑0 ↓1	↑1 ↓2	↑2 ↓5	↑4 ↓2	↑5 ↓0	↑23 ↓10
25	(10)	↑0 ↓0	↑4 ↓0	↑2 ↓0	↑1 ↓2	↑2 ↓4	↑1 ↓5	↑3 ↓2	↑5 ↓0	↑18 ↓13
50	(10)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑2 ↓2	↑1 ↓4	↑1 ↓4	↑3 ↓3	↑7 ↓0	↑26 ↓13
100	(10)	↑0 ↓1	↑6 ↓0	↑2 ↓0	↑1 ↓1	↑1 ↓2	↑1 ↓5	↑0 ↓4	↑4 ↓3	↑15 ↓16
200	(10)	↑1 ↓1	↑6 ↓0	↑2 ↓0	↑2 ↓1	↑1 ↓5	↑1 ↓4	↑1 ↓7	↑3 ↓4	↑17 ↓22
pc 5	(10)	↑2 ↓1	↑7 ↓0	↑3 ↓0	↑5 ↓0	↑3 ↓3	↑5 ↓2	↑3 ↓3	↑9 ↓0	↑37 ↓9
10	(10)	↑1 ↓1	↑6 ↓0	↑4 ↓0	↑1 ↓2	↑3 ↓0	↑1 ↓3	↑4 ↓2	↑7 ↓1	↑27 ↓9
20	(10)	↑0 ↓0	↑5 ↓0	↑2 ↓0	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑3 ↓0	↑5 ↓1	↑15 ↓6
40	(10)	↑0 ↓0	↑4 ↓0	↑4 ↓0	↑0 ↓2	↑0 ↓5	↑0 ↓6	↑1 ↓5	↑2 ↓1	↑11 ↓19
80	(10)	↑0 ↓0	↑6 ↓0	↑2 ↓0	↑0 ↓2	↑0 ↓7	↑0 ↓10	↑0 ↓8	↑1 ↓4	↑9 ↓31
C	(25)	↑0 ↓2	↑9 ↓0	↑5 ↓0	↑1 ↓6	↑0 ↓11	↑0 ↓14	↑0 ↓12	↑9 ↓3	↑24 ↓48
S	(25)	↑3 ↓0	↑19 ↓0	↑10 ↓0	↑5 ↓1	↑6 ↓6	↑6 ↓9	↑11 ↓6	↑15 ↓4	↑75 ↓26
All	(50)	↑3 ↓2	↑28 ↓0	↑15 ↓0	↑6 ↓7	↑6 ↓17	↑6 ↓23	↑11 ↓18	↑24 ↓7	↑99 ↓74
Set.	**Max**	**100 Dimensions**								
M_{n_p} 10	(10)	↑0 ↓0	↑0 ↓3	↑0 ↓5	↑0 ↓2	↑2 ↓2	↑5 ↓4	↑5 ↓3	↑5 ↓2	↑17 ↓21
25	(10)	↑0 ↓0	↑0 ↓4	↑0 ↓4	↑0 ↓3	↑0 ↓2	↑5 ↓2	↑5 ↓2	↑5 ↓2	↑15 ↓19
50	(10)	↑0 ↓1	↑0 ↓5	↑1 ↓5	↑0 ↓3	↑0 ↓3	↑3 ↓2	↑4 ↓3	↑6 ↓2	↑14 ↓24
100	(10)	↑0 ↓0	↑2 ↓4	↑1 ↓5	↑0 ↓4	↑0 ↓4	↑2 ↓3	↑4 ↓2	↑5 ↓3	↑14 ↓25
200	(10)	↑1 ↓1	↑2 ↓1	↑0 ↓3	↑0 ↓6	↑1 ↓4	↑0 ↓5	↑2 ↓2	↑4 ↓3	↑10 ↓25
pc 5	(10)	↑1 ↓0	↑2 ↓2	↑1 ↓4	↑0 ↓2	↑2 ↓0	↑2 ↓1	↑3 ↓0	↑5 ↓2	↑16 ↓11
10	(10)	↑0 ↓0	↑1 ↓3	↑0 ↓3	↑0 ↓2	↑0 ↓1	↑3 ↓1	↑4 ↓0	↑5 ↓1	↑13 ↓11
20	(10)	↑0 ↓0	↑1 ↓3	↑0 ↓5	↑0 ↓2	↑1 ↓1	↑4 ↓3	↑4 ↓3	↑5 ↓1	↑15 ↓18
40	(10)	↑0 ↓0	↑0 ↓4	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑3 ↓5	↑5 ↓4	↑5 ↓4	↑13 ↓33
80	(10)	↑0 ↓2	↑0 ↓5	↑1 ↓5	↑0 ↓7	↑0 ↓7	↑3 ↓6	↑4 ↓5	↑5 ↓4	↑13 ↓41
C	(25)	↑0 ↓0	↑4 ↓0	↑2 ↓0	↑0 ↓5	↑0 ↓9	↑0 ↓14	↑1 ↓12	↑3 ↓12	↑10 ↓52
S	(25)	↑1 ↓2	↑0 ↓17	↑0 ↓22	↑0 ↓13	↑3 ↓6	↑15 ↓2	↑19 ↓0	↑22 ↓0	↑60 ↓62
All	(50)	↑1 ↓2	↑4 ↓17	↑2 ↓22	↑0 ↓18	↑3 ↓15	↑15 ↓16	↑20 ↓12	↑25 ↓12	↑70 ↓114
Set.	**Max**	**All Dimensions**								
M_{n_p} 10	(50)	↑5 ↓1	↑18 ↓3	↑9 ↓5	↑8 ↓4	↑16 ↓5	↑18 ↓12	↑27 ↓5	↑24 ↓3	↑125 ↓38
25	(50)	↑2 ↓0	↑15 ↓4	↑4 ↓5	↑12 ↓7	↑17 ↓9	↑27 ↓8	↑33 ↓4	↑36 ↓2	↑146 ↓39
50	(50)	↑3 ↓2	↑14 ↓7	↑12 ↓6	↑12 ↓9	↑18 ↓8	↑26 ↓8	↑32 ↓6	↑42 ↓2	↑159 ↓48
100	(50)	↑1 ↓3	↑18 ↓5	↑11 ↓5	↑16 ↓6	↑17 ↓10	↑23 ↓11	↑29 ↓8	↑36 ↓6	↑151 ↓54
200	(50)	↑8 ↓5	↑14 ↓1	↑10 ↓3	↑12 ↓10	↑18 ↓15	↑18 ↓15	↑26 ↓13	↑32 ↓8	↑138 ↓70
pc 5	(50)	↑8 ↓3	↑21 ↓4	↑14 ↓6	↑23 ↓3	↑27 ↓3	↑31 ↓4	↑32 ↓3	↑40 ↓2	↑196 ↓28
10	(50)	↑5 ↓1	↑15 ↓4	↑8 ↓3	↑17 ↓7	↑23 ↓2	↑25 ↓7	↑33 ↓3	↑38 ↓2	↑164 ↓29
20	(50)	↑3 ↓1	↑12 ↓3	↑3 ↓5	↑10 ↓4	↑15 ↓3	↑22 ↓6	↑32 ↓3	↑36 ↓3	↑133 ↓28
40	(50)	↑1 ↓1	↑15 ↓4	↑10 ↓5	↑5 ↓9	↑11 ↓16	↑18 ↓13	↑26 ↓10	↑30 ↓5	↑116 ↓63
80	(50)	↑2 ↓5	↑16 ↓5	↑11 ↓5	↑5 ↓13	↑10 ↓23	↑16 ↓24	↑24 ↓17	↑26 ↓9	↑110 ↓101
C	(125)	↑7 ↓7	↑25 ↓0	↑14 ↓0	↑17 ↓18	↑29 ↓31	↑38 ↓36	↑51 ↓27	↑64 ↓16	↑245 ↓135
S	(125)	↑12 ↓4	↑54 ↓20	↑32 ↓24	↑43 ↓18	↑57 ↓16	↑74 ↓18	↑96 ↓9	↑106 ↓5	↑474 ↓114
All	(250)	↑19 ↓11	↑79 ↓20	↑46 ↓24	↑60 ↓36	↑86 ↓47	↑112 ↓54	↑147 ↓36	↑170 ↓21	↑719 ↓249

significantly better more often than both the other algorithms. Furthermore, with the exception of high dimensional problems, DynPopDE yielded large percentage improvements over DynDE and CDE.

5.3.7 Research Question 6

What is the convergence profile of DynPopDE?

The previous sections found that DynPopDE is, generally, a better optimisation algorithm than DynDE and CDE on the environments which are the focus of this chapter. This research question compares the convergence behaviour, in terms of current error, offline error and diversity, of DynPopDE to that of DynDE and CDE. This research question also investigates the number of sub-populations that DynPopDE uses on various environments.

Figure 5.23 gives the offline and current errors of DynDE, CDE and DynPopDE on the MPB using the Scenario 2 settings (i.e. a change period of 5 000 in five dimensions). All values are averaged over 30 repeats and the first 50 000 function evaluations (10 changes in the environment) are depicted. The average number of sub-populations used by DynPopDE is included. DynPopDE commenced with a single sub-population, but the number of sub-populations quickly increased to about eight after 5 000 function evaluations, and stabilised at about 14 after 50 000 function evaluations. The number of sub-populations was higher than the number of peaks in this environment. However, this is not necessarily a negative occurrence, as only the best sub-population is evolved at any given time through the competitive population evaluation process.

The initial small number of sub-populations enabled the offline error of DynPopDE to decrease noticeably faster than that of DynDE and CDE, at the commencement of the optimisation process. The difference between the offline errors of the three algorithms decreased as time passed. CDE yielded a lower final offline error than DynPopDE on this environment, so CDE's offline error eventually dropped to a value lower than that of DynPopDE.

DynPopDE perpetually creates and removes sub-populations. DynPopDE should consequently have more sub-populations that have not converged to an optimum than DynDE and CDE at any given time. These sub-populations caused DynPopDE to have higher current errors than DynDE and CDE, immediately after changes in the environment (compare

Figure 5.23 to Figure 4.34). DynPopDE's current errors dropped very quickly after changes and subsequently followed similar trends to those of DynDE and CDE.

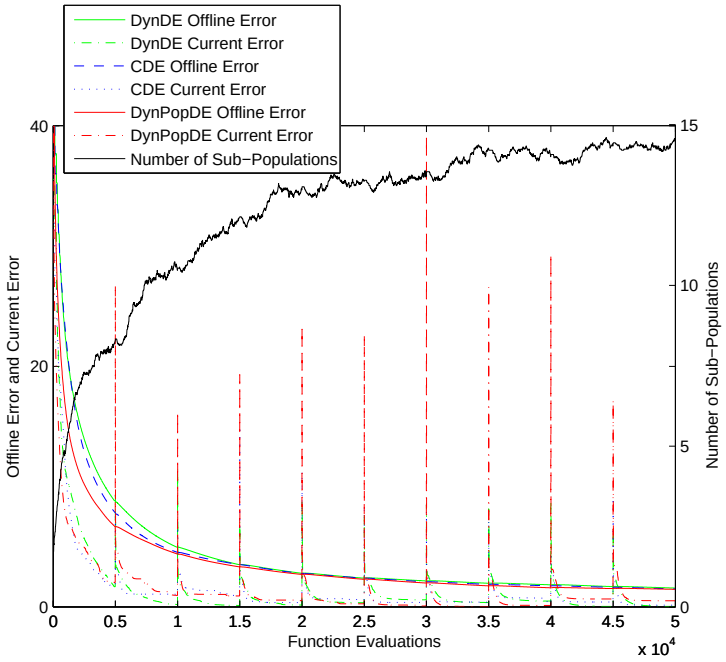

Figure 5.23: Offline errors and current errors of DynDE, CDE and DynPopDE on the MPB with the Scenario 2 settings.

Figure 5.24 gives the diversity and average sub-population diversity (calculated using equation (4.8) on page 156) of DynDE, CDE and DynPopDE for the same environment and period used in Figure 5.23. DynPopDE's diversity commences at a lower value than that of DynDE and CDE due the small number of initial sub-populations. However, the diversity value increases to a value very similar to that of DynDE and CDE after the first 1 000 function evaluations. The average sub-population diversity decreases faster than that of DynDE and CDE, as all initial function evaluations are allocated to a single sub-population. The fact that the average sub-population diversity of DynPopDE is very similar to that of CDE for most of the depicted period, shows the similarity in the behaviour of the two algorithms.

The previous sections found than DynPopDE is especially effective on problems with a large number of peaks when a high change period is used. Figure 5.25 gives the offline and current errors of DynDE, CDE and DynPopDE on the MPB, with the conical peak

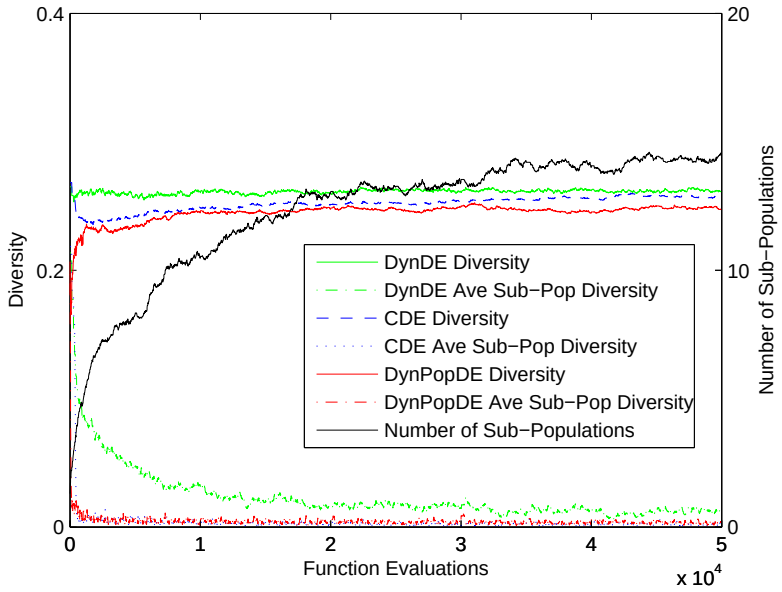

Figure 5.24: Diversity profiles of DynDE, CDE and DynPopDE on the MPB with the Scenario 2 settings.

function in five dimensions, with 100 peaks and a change period of 100 000. The average number of sub-populations used by DynPopDE is included. The first 300 000 function evaluations are shown.

The number of sub-populations used by DynPopDE was greater in Figure 5.25 than the number used in Figure 5.23. DynPopDE thus created more sub-populations in order to track more of the peaks. The benefit of creating more sub-populations is clear from the resulting offline error. The current errors of DynDE and CDE reached a plateau relatively quickly after a change in the environment, as the sub-populations converged to sub-optimal peaks. The current error of DynPopDE continued to decrease and tended towards zero as more sub-populations were created and new optima were consequently discovered. DynPopDE's offline error reached a value close to zero after 50 000 function evaluations. DynPopDE thus benefited greatly from having a large number of function evaluations available. This explains the trend, observed in previous sections, that DynPopDE was comparatively more effective than CDE and DynDE on problems with a large change period.

A high change period, however, does not guarantee a low offline error for DynPopDE.

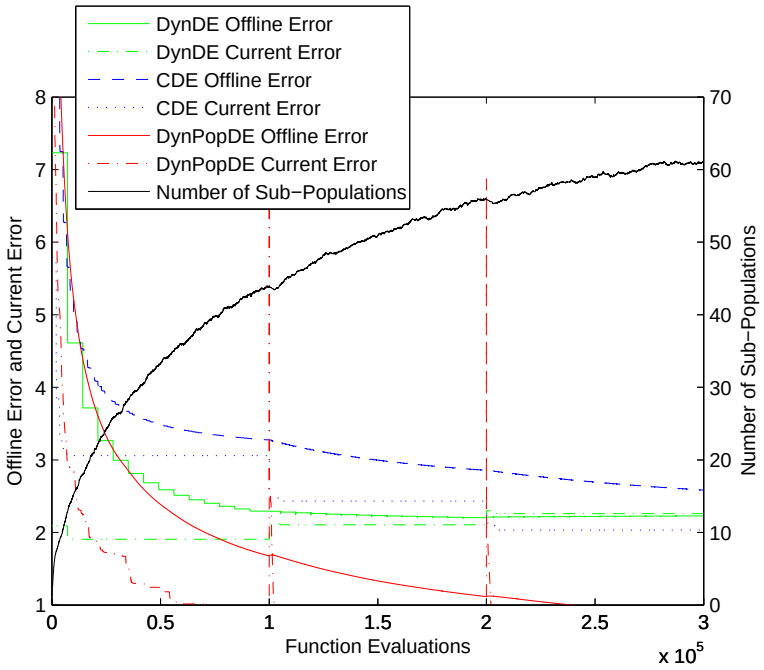

Figure 5.25: Offline errors and current errors of DynDE, CDE and DynPopDE on the MPB with the conical peak function in five dimensions with 100 peaks and a change period of 100 000

Figure 5.26 gives the offline and current errors of DynDE, CDE and DynPopDE over the first 300 000 function evaluations on the MPB with the conical peak function in 100 dimensions with 100 peaks and a change period of 100 000. The average number of sub-populations used by DynPopDE is included. The figure shows that the current errors of DynPopDE and CDE followed a very similar trend in high dimensions. DynPopDE's offline and current errors were initially slightly lower than those of CDE, but reached similar values after the second change in the environment. The number of sub-populations that was created by DynPopDE in 100 dimensions was considerably lower than that created in five dimensions (compare Figures 5.26 and 5.25). This suggests that the sub-population spawning process of DynPopDE is less effective in high dimensional environments.

The change period and number of dimensions were found to influence the number of sub-populations that DynPopDE creates. Figures 5.27 to 5.32 show how the number of sub-populations changes over time for various numbers of peaks being present in the environment. All figures give data collected over 30 repeats on the conical MPB function. Figures 5.27 and 5.28 give the number of sub-populations in five dimensional environments

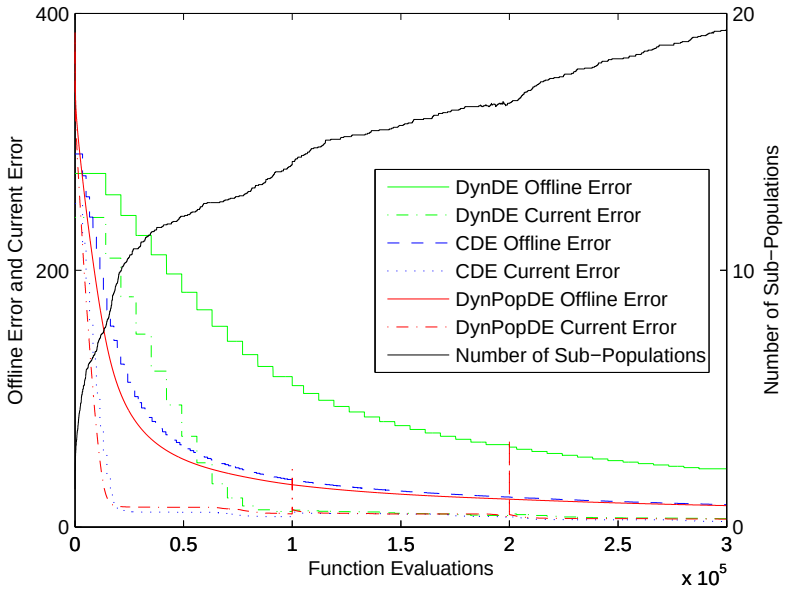

Figure 5.26: Offline errors and current errors of DynDE, CDE and DynPopDE on the MPB with the conical peak function in 100 dimensions with 100 peaks and a change period of 100 000

with a change period of 5 000 and 100 000, respectively. Figures 5.29 and 5.30 give the same information in 25 dimensions, while Figures 5.31 and 5.32 give the same information in 100 dimensions.

A considerable differentiation can be observed in the number of sub-populations created by DynPopDE for the various settings of number of peaks in five dimensions. A comparatively small number of sub-populations was created when a small number of peaks was present, but the number of sub-populations was typically greater than the number of peaks. A larger number of sub-populations was created when larger numbers of peaks were present in the environment, but the peaks typically outnumbered the sub-populations. This is consistent with the findings of research question 1, which showed that it is generally better to use a smaller number of sub-populations than the number of peaks.

DynPopDE created more sub-populations for five dimensional environments containing a large number of peaks when a change period of 100 000 was used than when a change period of 5 000 was used. A large change period enables the effective evolution of more sub-populations, making DynPopDE's choice of number of sub-populations sensible.

A distinctly different trend can be seen in Figure 5.29 as opposed to Figure 5.27. Dyn-

PopDE created a very similar number of sub-populations for the different settings of the number of peaks in the 25 dimensional case. This trend continues in Figure 5.31, which shows the 100 dimensional case. DynPopDE did not spawn sub-populations appropriately for the various environments, which results in DynPopDE's inferior performance in comparison to CDE in 100 dimensions.

The ineffective differentiation between the number of peak settings can be explained by considering the mechanism used by DynPopDE to detect stagnation. Equation (5.1) defines the function, $S(t)$, which must be true in order for DynPopDE to spawn a new sub-population. $S(t)$ is true when the change in fitness is equal to zero for all sub-populations. Ideally, this should occur when all sub-populations have converged to their respective optima. The high dimensional environments cause DynPopDE to converge slowly, which in turn causes $S(t)$ to be true infrequently. DynPopDE, consequently, creates too few sub-populations in high dimensional environments.

The problem is compounded by the sub-population removal mechanism, which is incorporated into the exclusion process. Exclusion occurs only when sub-populations converge to the same optima. However when the sub-populations converge slowly, the sub-populations infrequently converge to the same optima before changes occur in the environment. This means that unnecessary sub-populations are not removed, as can be seen in Figures 5.29 and 5.31, where more sub-populations were present than the number of peaks for the environments with low numbers of peaks.

DynPopDE's ineffective population spawning problem in high dimensions is alleviated by increasing the change period. Figures 5.30 and 5.32 give the number of sub-populations per number of peak setting in 25 and 100 dimensions, respectively, using a change period of 100 000. A comparison of these figures with Figures 5.29 and 5.31 shows that DynPopDE created a number of sub-populations proportional to the number of peaks that was present, when a high change period was used. The higher change period made it more likely that sub-populations would converge to optima, which assisted in detecting stagnation. The sub-population removal process was also improved by a high change period, as sub-populations converged to the same optima and were subsequently removed through the exclusion process. Note that DynPopDE was still less effective at creating different numbers of sub-populations for different numbers of peaks in 100 dimensions

than in 25. This explains why DynPopDE outperformed CDE only in environments with a high number of peaks and high change period in 100 dimensions.

DynPopDE can respond to changes in the environment when the number of optima fluctuates over time. The relationship between the number of peaks and number of sub-populations that was found in a typical run of the DynPopDE algorithm is illustrated in Figure 5.33. The environment used to produce this figure is the conical function in five dimensions. The maximum number of peaks was set to 50 and the percentage change in the number of peaks was set to 20%. Changes occurred once every 5 000 function evaluations. It is clear that the number of sub-populations increases when the number of peaks increases, but remains less than the number of peaks when the number of peaks is large. This is consistent with results found in the unknown number of peaks experiments which indicated that the number of sub-populations should be kept relatively small when many peaks are present.

The region between 100 000 and 130 000 function evaluations in Figure 5.33 represents a period during which the number of sub-populations outnumbered the number of peaks. In fact, for a brief period there were eight sub-populations present while there was only one peak in the environment. This is not necessarily a negative occurrence when the total number of sub-populations is small, since few sub-populations equate to more available function evaluations. Redundant sub-populations can assist in peak discovery and increase diversity, which is useful when new peaks are introduced. As long as the number of redundant sub-populations is small, the function evaluation overhead can be justified.

The existence of the three phases (refer to Sections 5.3.5 and 5.3.6) provides an insight into the comparative effectiveness of the DynPopDE algorithm. DynPopDE was effective at low change periods (phase one) because it commences with a single sub-population which resulted in more generations between changes in the environment.

DynPopDE was generally less effective than CDE during phase two. This phase was most evident in high dimensions on mid-range change periods. This corresponds to the environments in which, according to the discussion in the current section, it was found that DynPopDE's sub-population spawning and removal processes are not effective. DynPopDE's comparatively poor performance during phase two can thus be ascribed to the inability of the algorithm to respond effectively to the changes in numbers of peaks.

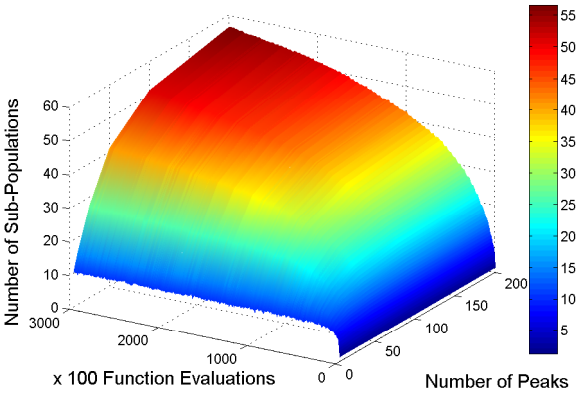

Figure 5.27: Number of sub-populations used by DynPopDE on the conical peak function in five dimensions with a change period of 5 000.

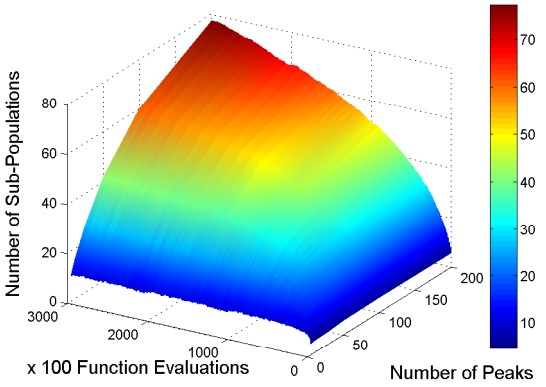

Figure 5.28: Number of sub-populations used by DynPopDE on the conical peak function in five dimensions with a change period of 100 000.

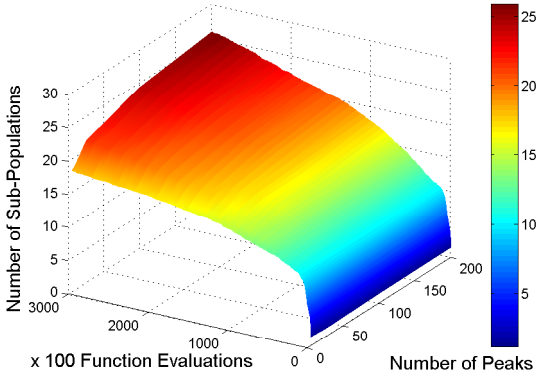

Figure 5.29: Number of sub-populations used by DynPopDE on the conical peak function in 25 dimensions with a change period of 5 000.

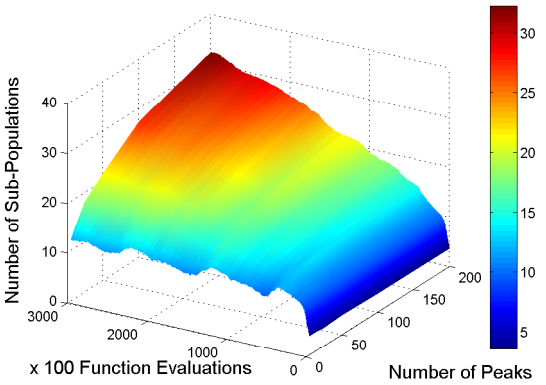

Figure 5.30: Number of sub-populations used by DynPopDE on the conical peak function in 25 dimensions with a change period of 100 000.

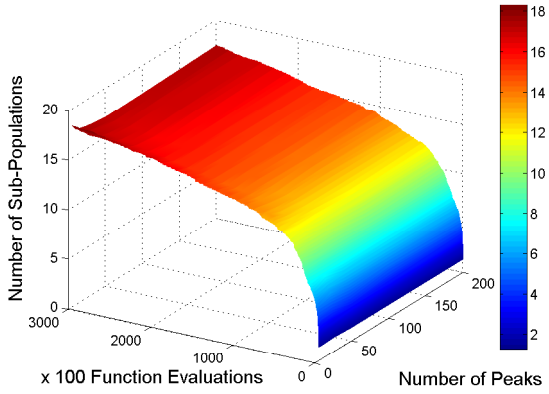

Figure 5.31: Number of sub-populations used by DynPopDE on the conical peak function in 100 dimensions with a change period of 5 000.

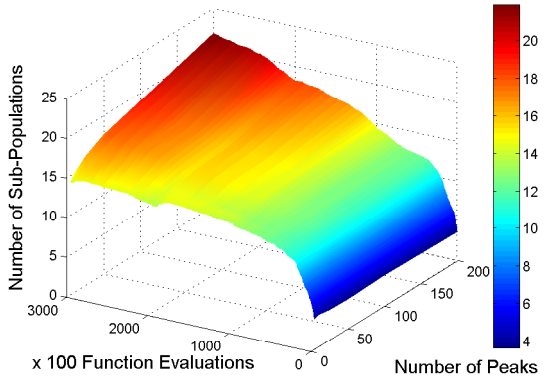

Figure 5.32: Number of sub-populations used by DynPopDE on the conical peak function in 100 dimensions with a change period of 100 000.

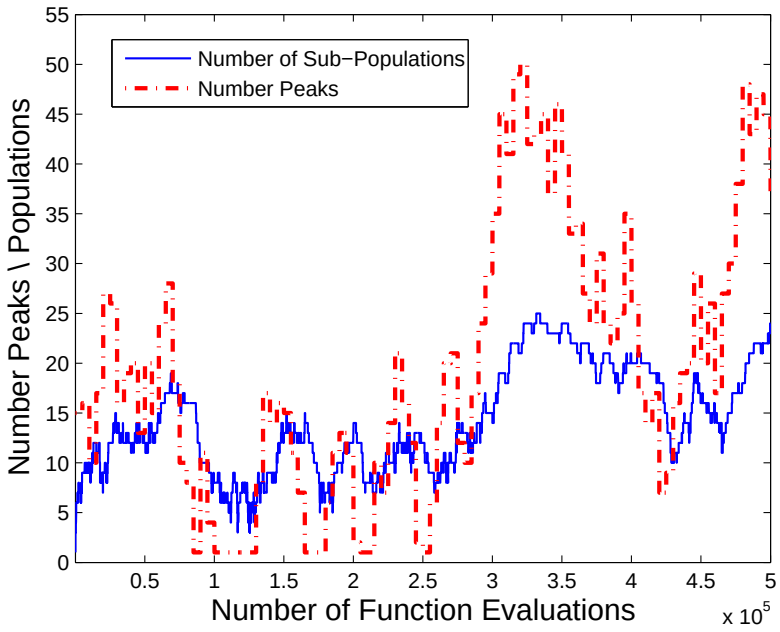

Figure 5.33: Comparison between the number of peaks and the number of populations in a typical execution of the DynPopDE algorithm on the conical peak function in five dimensions with a change period of 5 000, a maximum of 50 peaks and a 20% change in the number of peaks.

DynPopDE performed better than CDE during phase three, which occurred in low dimensions and at high change periods. This corresponds to environments in which DynPopDE effectively created sub-populations based on the number of peaks that were present. DynPopDE thus requires a sufficient number of function evaluations between changes in the environment to respond effectively to changes in the number of optima.

5.3.8 Research Question 7

Is the process of spawning and removing sub-populations as used in DynPopDE more effective than the process used in MPSO2?

This chapter introduced the sub-population spawning and removal processes that were incorporated into CDE to form DynPopDE. This research question investigates whether other spawning and removal processes, namely the approaches used in MPSO2, are a better choice for incorporation into CDE, and subsequently could yield better results.

The MPSO2 algorithm of Blackwell [2007] was discussed in Section 3.3.3.2. A dis-

advantage of MPSO2 is that it has a parameter, n_{excess}, that must be manually tuned. Blackwell [2007] did, however, find that for all values of $n_{excess} > 5$ the algorithm demonstrates identical behaviour to each other without significantly degrading performance. Despite the previously mentioned disadvantage associated with the MPSO2 algorithm, the reported results suggest that the swarm spawning and removing approach followed by Blackwell [2007] could potentially be more effective than the technique used in DynPopDE. This possibility was investigated by incorporating the approach to spawn and remove populations as used in MPSO2 into CDE. A sub-population is thus created when there are no available free sub-populations, i.e. all the current sub-populations had converged to within a diameter of $2r_{conv}$, where r_{conv} is calculated using the same equation as r_{excl} (equation (4.1)). The worst performing free sub-population is removed when the number of free sub-populations exceeds n_{excess}. The new algorithm is referred to as MPSO2CDE.

Experiments were conducted on the $n_p(t)$ standard set of experiments, using the same experimental procedure as for DynDE, CDE and DynPopDE. The parameter n_{excess} was set to $n_{excess} = 10$, as this value was found to be within the range where the algorithm exhibits identical behaviour [du Plessis and Engelbrecht, 2012a].

The performance analysis of MPSO2CDE compared to DynPopDE is given in Table 5.8. MPSO2CDE performed statistically significantly better than DynPopDE in only 64 of the 2 000 experiments, while it performed worse in 1 609 experiments. MPSO2CDE did, however, perform better more often than DynPopDE in experiments that used a change period of 100. The improvement over DynPopDE on these environments is minor as only 36 of the 250 experiments yielded statistically significantly different results.

The results of experiments that were conducted to answer this research question show that the population spawning and removal components of MPSO2, when incorporated into CDE, do not work as effectively as the spawning and removal components of DynPopDE.

5.3.9 Research Question 8

Is DynPopDE more effective than CDE and DynDE on the set of environments used in Chapter 4?

The experiments conducted in this chapter, up to this point, have all been on the MPB or the extended MPB, as these provide the functionality to vary or fluctuate the number of

Table 5.8: MPSO2CDE vs DynPopDE performance analysis on the $n_p(t)$ standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 10	(10)	↑**3** ↓**0**	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑3 ↓69
25	(10)	↑**2** ↓**0**	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑2 ↓66
50	(10)	↑0 ↓1	↑0 ↓6	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
100	(10)	↑**1** ↓**0**	↑2 ↓6	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑3 ↓64
200	(10)	↑1 ↓1	↑1 ↓5	↑0 ↓7	↑0 ↓9	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑2 ↓60
pc 5	(10)	↑1 ↓1	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓71
10	(10)	↑0 ↓1	↑1 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓68
20	(10)	↑**1** ↓**0**	↑0 ↓6	↑0 ↓8	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓63
40	(10)	↑**3** ↓**0**	↑1 ↓6	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑4 ↓63
80	(10)	↑**2** ↓**0**	↑1 ↓3	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑3 ↓59
C	(25)	↑**2** ↓**0**	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑2 ↓173
S	(25)	↑**5** ↓**2**	↑3 ↓10	↑0 ↓17	↑0 ↓24	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑8 ↓151
All	(50)	↑**7** ↓**2**	↑3 ↓33	↑0 ↓42	↑0 ↓49	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑10 ↓324
Set.	**Max**	**10 Dimensions**								
n_p 10	(10)	↑0 ↓1	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓69
25	(10)	↑**1** ↓**0**	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓68
50	(10)	↑0 ↓2	↑0 ↓6	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
100	(10)	↑0 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
200	(10)	↑0 ↓0	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓66
pc 5	(10)	↑0 ↓1	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓70
10	(10)	↑0 ↓2	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓72
20	(10)	↑0 ↓1	↑0 ↓9	↑0 ↓9	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
40	(10)	↑0 ↓0	↑0 ↓5	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
80	(10)	↑**1** ↓**0**	↑0 ↓3	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓62
C	(25)	↑1 ↓2	↑0 ↓21	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑1 ↓173
S	(25)	↑0 ↓2	↑0 ↓15	↑0 ↓23	↑0 ↓24	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓164
All	(50)	↑1 ↓4	↑0 ↓36	↑0 ↓48	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑1 ↓337
Set.	**Max**	**25 Dimensions**								
n_p 10	(10)	↑**1** ↓**0**	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓70
25	(10)	↑**1** ↓**0**	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓70
50	(10)	↑1 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓68
100	(10)	↑0 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
200	(10)	↑0 ↓1	↑0 ↓4	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
pc 5	(10)	↑0 ↓3	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓73
10	(10)	↑0 ↓0	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
20	(10)	↑0 ↓0	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
40	(10)	↑**2** ↓**0**	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑2 ↓67
80	(10)	↑**1** ↓**0**	↑0 ↓6	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓66
C	(25)	↑0 ↓0	↑0 ↓16	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓166
S	(25)	↑3 ↓3	↑0 ↓22	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑3 ↓175
All	(50)	↑3 ↓3	↑0 ↓38	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑3 ↓341
Set.	**Max**	**50 Dimensions**								
n_p 10	(10)	↑0 ↓0	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
25	(10)	↑0 ↓0	↑0 ↓6	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓66
50	(10)	↑0 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
100	(10)	↑0 ↓0	↑0 ↓9	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
200	(10)	↑0 ↓1	↑0 ↓8	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
pc 5	(10)	↑0 ↓2	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓71
10	(10)	↑0 ↓0	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓69
20	(10)	↑0 ↓0	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
40	(10)	↑0 ↓0	↑0 ↓6	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
80	(10)	↑0 ↓0	↑0 ↓5	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓64
C	(25)	↑0 ↓0	↑0 ↓12	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓160
S	(25)	↑0 ↓2	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓177
All	(50)	↑0 ↓2	↑0 ↓37	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓337
Set.	**Max**	**100 Dimensions**								
n_p 10	(10)	↑**2** ↓**0**	↑1 ↓2	↑1 ↓5	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑4 ↓57
25	(10)	↑**2** ↓**0**	↑**5** ↓**2**	↑2 ↓5	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑9 ↓56
50	(10)	↑**3** ↓**0**	↑**5** ↓**4**	↑2 ↓4	↑0 ↓9	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑10 ↓56
100	(10)	↑**5** ↓**0**	↑**5** ↓**3**	↑4 ↓5	↑1 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑15 ↓54
200	(10)	↑**2** ↓**0**	↑**5** ↓**3**	↑4 ↓4	↑1 ↓5	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓9	↑12 ↓47
pc 5	(10)	↑**1** ↓**0**	↑4 ↓4	↑2 ↓5	↑0 ↓8	↑0 ↓9	↑0 ↓8	↑0 ↓9	↑0 ↓9	↑7 ↓52
10	(10)	↑**5** ↓**0**	↑4 ↓4	↑0 ↓4	↑0 ↓8	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑9 ↓55
20	(10)	↑**2** ↓**0**	↑**4** ↓**3**	↑2 ↓5	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑8 ↓57
40	(10)	↑**1** ↓**0**	↑**4** ↓**2**	↑4 ↓5	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑9 ↓55
80	(10)	↑**5** ↓**0**	↑**5** ↓**1**	↑**5** ↓**4**	↑2 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑17 ↓51
C	(25)	↑0 ↓0	↑0 ↓14	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓162
S	(25)	↑**14** ↓**0**	↑**21** ↓**0**	↑**13** ↓**0**	↑2 ↓15	↑0 ↓22	↑0 ↓23	↑0 ↓24	↑0 ↓24	↑50 ↓108
All	(50)	↑**14** ↓**0**	↑**21** ↓**14**	↑13 ↓23	↑2 ↓40	↑0 ↓47	↑0 ↓48	↑0 ↓49	↑0 ↓49	↑50 ↓270
Set.	**Max**	**All Dimensions**								
n_p 10	(50)	↑**6** ↓**1**	↑1 ↓36	↑1 ↓45	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑8 ↓332
25	(50)	↑**6** ↓**0**	↑5 ↓33	↑2 ↓44	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑13 ↓326
50	(50)	↑4 ↓5	↑5 ↓30	↑2 ↓41	↑0 ↓49	↑0 ↓50	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑11 ↓324
100	(50)	↑**6** ↓**2**	↑7 ↓32	↑4 ↓42	↑1 ↓46	↑0 ↓50	↑0 ↓50	↑0 ↓49	↑0 ↓50	↑18 ↓321
200	(50)	↑3 ↓3	↑6 ↓27	↑4 ↓39	↑1 ↓44	↑0 ↓45	↑0 ↓49	↑0 ↓50	↑0 ↓49	↑14 ↓306
pc 5	(50)	↑2 ↓7	↑4 ↓42	↑2 ↓45	↑0 ↓48	↑0 ↓49	↑0 ↓48	↑0 ↓49	↑0 ↓49	↑8 ↓337
10	(50)	↑**5** ↓**3**	↑5 ↓39	↑0 ↓44	↑0 ↓48	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑10 ↓332
20	(50)	↑**3** ↓**1**	↑4 ↓33	↑2 ↓42	↑0 ↓48	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑9 ↓323
40	(50)	↑**6** ↓**0**	↑5 ↓26	↑4 ↓41	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑15 ↓315
80	(50)	↑**9** ↓**0**	↑6 ↓18	↑5 ↓39	↑2 ↓46	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑22 ↓302
C	(125)	↑**3** ↓**2**	↑0 ↓86	↑0 ↓121	↑0 ↓125	↑0 ↓125	↑0 ↓125	↑0 ↓125	↑0 ↓125	↑3 ↓834
S	(125)	↑**22** ↓**9**	↑24 ↓72	↑13 ↓90	↑2 ↓113	↑0 ↓120	↑0 ↓123	↑0 ↓124	↑0 ↓124	↑61 ↓775
All	(250)	↑**25** ↓**11**	↑24 ↓158	↑13 ↓211	↑2 ↓238	↑0 ↓245	↑0 ↓248	↑0 ↓249	↑0 ↓249	↑64 ↓1609

peaks. This research question investigates the performance of DynPopDE in comparision to DynDE and CDE on the standard set used in Chapter 4, since the number of optima in the functions of that set is also not known to the algorithm.

The standard set of experiments which was defined in Section 4.6.2 contains functions from both the MPB and the GDBG. The MPB environments contained 10 peaks in all the experiments conducted in Chapter 4, but the GDBG functions had various numbers of optima. For example, F_{1b} has 50 optima, and F_3 to F_6 have a large number of optima (refer to Section 2.5.4). DynPopDE was executed on the standard set for all variations of settings listed in Table 4.3. This is the same set of 2 160 experiments used to evaluate the algorithms in the previous chapter.

A performance analysis was conducted to determine the number of experimental cases where DynPopDE performed statistically significantly better than DynDE and CDE. The analysis showed that DynPopDE performed better than DynDE in 867 of the 2 160 experiments while performing worse in 886 (the full performance analysis is given in Appendix C). DynPopDE thus performed better than DynDE in fewer than half of the environments.

The performance analysis comparison between DynPopDE and CDE is given in Tables 5.9 and 5.10. DynPopDE performed significantly better than CDE in 367 of the 2 160 experiments, while CDE performed better than DynPopDE in 1 257 cases. The tables show that the cases where DynPopDE performed better more often than CDE were concentrated in the MPB environments, with the exception of environments that used a low change period. DynPopDE performed better than CDE on functions F_2 and F_3 for a high change period in five dimensions, but DynPopDE was inferior to CDE in the vast majority of GDBG environments in high dimensions for high change periods. The cases where DynPopDE performed better than CDE on the MPB functions were mainly isolated to the spherical peak function in high dimensions.

The average percentage improvement of DynPopDE over CDE, over all experiments, was found to be -11.50%. The APIs per dimension were found to be -10.15%, -12.01%, -11.04%, -8.09%, -16.19% and -10.07% for 5, 10, 25, 50 and 100 dimensions respectively. The APIs per change period, were found to be 2.37%, 2.89%, -0.82%, -14.98%, -18.87%, -21.64%, -21.38% and -19.56% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. DynPopDE was thus better than

Table 5.9: DynPopDE vs CDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	**↑4 ↓3**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓6*
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓7*
20	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓6*
40	(2)	*↑0 ↓2*	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑6 ↓2**
80	(2)	*↑0 ↓2*	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑7 ↓3**
C	(6)	*↑1 ↓2*	**↑3 ↓0**	**↑3 ↓0**	↑2 ↓2	*↑1 ↓2*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑11 ↓6**
S	(6)	*↑1 ↓3*	**↑3 ↓1**	**↑3 ↓0**	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓3*	*↑1 ↓4*	*↑1 ↓3*	*↑15 ↓21*
		GDBG								
F_{1a}	(6)	**↑4 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓2**	↑1 ↓1	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑18 ↓4**
F_{1b}	(6)	**↑3 ↓0**	**↑3 ↓2**	↑3 ↓3	*↑0 ↓5*	*↑0 ↓6*	*↑1 ↓4*	↑2 ↓2	**↑2 ↓1**	*↑14 ↓23*
F_2	(6)	**↑2 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓2**	**↑4 ↓2**	↑3 ↓3	↑1 ↓1	**↑1 ↓0**	**↑25 ↓9**
F_3	(6)	↑0 ↓0	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓1*	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑20 ↓11**
F_4	(6)	**↑1 ↓0**	**↑2 ↓1**	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑3 ↓33*
F_5	(6)	**↑2 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑2 ↓42*
F_6	(6)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓38*
T_1	(7)	**↑4 ↓0**	*↑1 ↓5*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓5*	*↑1 ↓4*	*↑1 ↓3*	*↑1 ↓3*	*↑8 ↓33*
T_2	(7)	**↑1 ↓0**	**↑4 ↓2**	↑3 ↓3	*↑2 ↓4*	*↑3 ↓4*	*↑2 ↓4*	*↑1 ↓3*	*↑1 ↓3*	*↑17 ↓23*
T_3	(7)	**↑1 ↓0**	**↑4 ↓2**	↑3 ↓3	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓3*	*↑1 ↓3*	*↑1 ↓3*	*↑16 ↓21*
T_4	(7)	**↑5 ↓0**	*↑2 ↓3*	*↑1 ↓5*	*↑0 ↓6*	*↑1 ↓6*	*↑1 ↓5*	*↑2 ↓5*	*↑2 ↓3*	*↑14 ↓33*
T_5	(7)	↑0 ↓0	**↑3 ↓1**	↑3 ↓3	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑16 ↓23*
T_6	(7)	**↑1 ↓0**	↑2 ↓2	*↑2 ↓5*	*↑1 ↓4*	*↑2 ↓4*	*↑1 ↓5*	*↑1 ↓4*	*↑1 ↓3*	*↑11 ↓27*
All	(54)	**↑14 ↓5**	**↑22 ↓16**	*↑18 ↓25*	*↑11 ↓32*	*↑13 ↓33*	*↑11 ↓28*	*↑9 ↓26*	*↑10 ↓22*	*↑108 ↓187*
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓3**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑5 ↓9*
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	*↑4 ↓8*
40	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓6*
80	(2)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑2 ↓4*
C	(6)	**↑2 ↓0**	**↑4 ↓0**	**↑3 ↓0**	*↑0 ↓3*	*↑1 ↓4*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑10 ↓12*
S	(6)	**↑3 ↓2**	**↑5 ↓0**	**↑5 ↓0**	**↑2 ↓1**	↑3 ↓3	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	**↑24 ↓18**
		GDBG								
F_{1a}	(6)	**↑4 ↓0**	**↑3 ↓2**	↑3 ↓3	*↑1 ↓3*	*↑0 ↓3*	*↑0 ↓3*	*↑0 ↓1*	↑0 ↓0	*↑11 ↓15*
F_{1b}	(6)	**↑2 ↓0**	*↑1 ↓2*	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓6*	*↑1 ↓5*	*↑0 ↓4*	↑2 ↓2	*↑7 ↓27*
F_2	(6)	**↑3 ↓0**	**↑2 ↓1**	**↑3 ↓2**	*↑2 ↓3*	*↑2 ↓3*	*↑2 ↓3*	*↑2 ↓4*	*↑1 ↓4*	*↑17 ↓20*
F_3	(6)	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓3*	**↑2 ↓1**	*↑2 ↓35*
F_4	(6)	**↑3 ↓0**	**↑2 ↓1**	**↑3 ↓2**	*↑2 ↓3*	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓3*	*↑2 ↓4*	*↑18 ↓20*
F_5	(6)	**↑1 ↓0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓41*
F_6	(6)	↑0 ↓0	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓40*
T_1	(7)	**↑5 ↓0**	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓5*	*↑1 ↓4*	*↑6 ↓44*
T_2	(7)	**↑3 ↓1**	↑3 ↓3	↑3 ↓3	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑9 ↓30*
T_3	(7)	↑1 ↓1	*↑2 ↓3*	**↑4 ↓3**	*↑2 ↓3*	*↑2 ↓4*	↑3 ↓3	*↑2 ↓3*	*↑1 ↓3*	*↑17 ↓23*
T_4	(7)	**↑3 ↓1**	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑2 ↓4*	*↑5 ↓44*
T_5	(7)	*↑0 ↓1*	**↑2 ↓1**	**↑3 ↓2**	*↑3 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓3*	↑2 ↓2	*↑16 ↓21*
T_6	(7)	**↑1 ↓0**	↑1 ↓1	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓5*	*↑1 ↓5*	*↑3 ↓36*
All	(54)	**↑18 ↓6**	*↑17 ↓20*	*↑18 ↓27*	*↑7 ↓36*	*↑8 ↓40*	*↑7 ↓38*	*↑6 ↓33*	*↑9 ↓28*	*↑90 ↓228*
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓2**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	**↑9 ↓4**
20	(2)	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	*↑4 ↓5*
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑1 ↓9*
80	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑2 ↓10*
C	(6)	↑0 ↓0	**↑3 ↓0**	**↑3 ↓0**	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓3*	*↑0 ↓2*	*↑6 ↓19*
S	(6)	**↑3 ↓1**	**↑4 ↓0**	**↑6 ↓0**	**↑3 ↓2**	**↑3 ↓2**	**↑3 ↓2**	**↑4 ↓2**	**↑3 ↓2**	**↑29 ↓11**
		GDBG								
F_{1a}	(6)	**↑3 ↓0**	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓3*	*↑0 ↓3*	*↑0 ↓3*	*↑3 ↓24*
F_{1b}	(6)	**↑3 ↓0**	*↑1 ↓2*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓3*	*↑4 ↓31*
F_2	(6)	**↑6 ↓0**	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	*↑8 ↓25*
F_3	(6)	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓45*
F_4	(6)	**↑5 ↓0**	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓5*	*↑1 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑1 ↓5*	*↑8 ↓27*
F_5	(6)	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓41*
F_6	(6)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓42*
T_1	(7)	**↑4 ↓0**	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑4 ↓47*
T_2	(7)	**↑2 ↓1**	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑3 ↓40*
T_3	(7)	**↑2 ↓1**	*↑0 ↓3*	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	*↑2 ↓31*
T_4	(7)	**↑4 ↓2**	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑4 ↓49*
T_5	(7)	**↑3 ↓2**	**↑3 ↓2**	*↑0 ↓3*	*↑0 ↓5*	*↑1 ↓4*	*↑0 ↓4*	*↑0 ↓4*	*↑1 ↓3*	*↑8 ↓27*
T_6	(7)	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑2 ↓41*
All	(54)	**↑20 ↓7**	*↑11 ↓19*	*↑9 ↓30*	*↑3 ↓44*	*↑4 ↓44*	*↑3 ↓43*	*↑4 ↓41*	*↑4 ↓37*	*↑58 ↓265*

Table 5.10: DynPopDE vs CDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓0**
5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑10 ↓4**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑9 ↓4**
20	(2)	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑7 ↓5**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓9*
80	(2)	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑1 ↓11*
C	(6)	↑0 ↓0	**↑3 ↓0**	**↑3 ↓0**	*↑1 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	*↑7 ↓23*
S	(6)	**↑3 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑4 ↓2**	**↑4 ↓2**	**↑4 ↓2**	**↑33 ↓10**
		GDBG								
F_{1a}	(6)	**↑3 ↓0**	↑1 ↓1	*↑0 ↓3*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓4*	*↑4 ↓29*
F_{1b}	(6)	**↑4 ↓0**	*↑1 ↓2*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓4*	*↑5 ↓34*
F_2	(6)	**↑6 ↓0**	**↑2 ↓1**	*↑0 ↓3*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑8 ↓34*
F_3	(6)	*↑0 ↓2*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓43*
F_4	(6)	**↑6 ↓0**	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑7 ↓32*
F_5	(6)	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑1 ↓30*
F_6	(6)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓42*
T_1	(7)	**↑4 ↓0**	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑4 ↓47*
T_2	(7)	**↑2 ↓1**	*↑1 ↓2*	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓6*	*↑3 ↓41*
T_3	(7)	**↑3 ↓0**	*↑0 ↓2*	*↑0 ↓3*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑3 ↓35*
T_4	(7)	**↑4 ↓0**	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑4 ↓45*
T_5	(7)	**↑4 ↓1**	↑2 ↓2	*↑0 ↓3*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓6*	*↑6 ↓38*
T_6	(7)	**↑3 ↓0**	**↑2 ↓1**	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑5 ↓38*
All	(54)	**↑23 ↓3**	*↑13 ↓16*	*↑8 ↓28*	*↑5 ↓45*	*↑4 ↓49*	*↑4 ↓47*	*↑4 ↓46*	*↑4 ↓43*	*↑65 ↓277*
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓2*	**↑1 ↓0**	*↑4 ↓6*
5	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	**↑1 ↓0**	*↑0 ↓1*	*↑4 ↓7*
10	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓11*
20	(2)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓12*
40	(2)	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑1 ↓13*
80	(2)	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓13*
C	(6)	↑0 ↓0	**↑5 ↓0**	**↑4 ↓0**	*↑1 ↓3*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	*↑10 ↓21*
S	(6)	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓5*	*↑0 ↓6*	*↑1 ↓5*	*↑1 ↓5*	*↑3 ↓41*
		GDBG								
F_{1a}	(6)	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓3*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑2 ↓33*
F_{1b}	(6)	**↑5 ↓0**	↑0 ↓0	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑5 ↓34*
F_2	(6)	**↑6 ↓0**	**↑2 ↓0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑8 ↓35*
F_3	(6)	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓38*
F_4	(6)	**↑6 ↓0**	**↑2 ↓1**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑8 ↓36*
F_5	(6)	**↑2 ↓1**	**↑3 ↓0**	**↑3 ↓0**	*↑1 ↓3*	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓4*	*↑9 ↓20*
F_6	(6)	**↑1 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓42*
T_1	(7)	**↑3 ↓0**	*↑0 ↓2*	*↑1 ↓6*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑4 ↓42*
T_2	(7)	**↑4 ↓1**	*↑1 ↓2*	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑5 ↓43*
T_3	(7)	**↑4 ↓0**	**↑2 ↓1**	*↑0 ↓4*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑6 ↓40*
T_4	(7)	**↑5 ↓0**	**↑3 ↓2**	*↑1 ↓4*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑9 ↓41*
T_5	(7)	**↑2 ↓1**	**↑2 ↓1**	*↑1 ↓4*	*↑1 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑6 ↓36*
T_6	(7)	**↑3 ↓0**	*↑0 ↓1*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑3 ↓36*
All	(54)	**↑21 ↓4**	*↑13 ↓15*	*↑7 ↓34*	*↑2 ↓48*	*↑1 ↓49*	*↑0 ↓51*	*↑1 ↓50*	*↑1 ↓49*	*↑46 ↓300*
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑6 ↓0**	**↑8 ↓1**	**↑6 ↓1**	**↑4 ↓2**	**↑4 ↓1**	**↑4 ↓1**	↑3 ↓3	**↑4 ↓0**	**↑39 ↓9**
5	(10)	**↑4 ↓0**	**↑9 ↓1**	**↑6 ↓1**	**↑4 ↓3**	*↑3 ↓6*	*↑3 ↓6*	**↑4 ↓2**	↑3 ↓3	**↑36 ↓22**
10	(10)	**↑1 ↓0**	**↑8 ↓1**	**↑9 ↓1**	*↑2 ↓6*	*↑2 ↓8*	*↑2 ↓7*	*↑2 ↓6*	*↑2 ↓6*	*↑28 ↓35*
20	(10)	**↑2 ↓1**	**↑3 ↓1**	**↑7 ↓1**	*↑1 ↓7*	*↑1 ↓7*	*↑1 ↓6*	*↑2 ↓7*	*↑1 ↓6*	*↑18 ↓36*
40	(10)	*↑0 ↓4*	**↑4 ↓1**	**↑5 ↓1**	*↑2 ↓5*	*↑2 ↓8*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑13 ↓39*
80	(10)	*↑0 ↓6*	**↑3 ↓2**	**↑2 ↓1**	*↑2 ↓6*	*↑3 ↓6*	*↑1 ↓7*	*↑1 ↓7*	*↑2 ↓6*	*↑14 ↓41*
C	(30)	**↑3 ↓2**	**↑18 ↓0**	**↑16 ↓0**	*↑4 ↓16*	*↑2 ↓20*	*↑0 ↓17*	*↑0 ↓15*	*↑1 ↓11*	*↑44 ↓81*
S	(30)	**↑10 ↓9**	**↑17 ↓7**	**↑19 ↓6**	*↑11 ↓13*	*↑13 ↓16*	*↑11 ↓17*	*↑12 ↓17*	*↑11 ↓16*	**↑104 ↓101**
		GDBG								
F_{1a}	(30)	**↑15 ↓0**	**↑11 ↓5**	*↑7 ↓13*	*↑4 ↓22*	*↑1 ↓20*	*↑0 ↓17*	*↑0 ↓15*	*↑0 ↓13*	*↑38 ↓105*
F_{1b}	(30)	**↑17 ↓0**	*↑6 ↓8*	*↑4 ↓20*	*↑0 ↓28*	*↑0 ↓29*	*↑2 ↓26*	*↑2 ↓22*	*↑4 ↓16*	*↑35 ↓149*
F_2	(30)	**↑23 ↓0**	**↑13 ↓2**	*↑8 ↓13*	*↑6 ↓21*	*↑6 ↓22*	*↑5 ↓23*	*↑3 ↓22*	*↑2 ↓20*	*↑66 ↓123*
F_3	(30)	*↑0 ↓11*	*↑0 ↓20*	*↑0 ↓29*	*↑0 ↓25*	*↑5 ↓24*	*↑5 ↓23*	*↑5 ↓21*	*↑7 ↓19*	*↑22 ↓172*
F_4	(30)	**↑21 ↓0**	**↑8 ↓3**	*↑3 ↓14*	*↑2 ↓25*	*↑3 ↓26*	*↑2 ↓27*	*↑2 ↓26*	*↑3 ↓27*	*↑44 ↓148*
F_5	(30)	**↑6 ↓3**	*↑3 ↓16*	*↑3 ↓20*	*↑1 ↓25*	*↑0 ↓28*	*↑0 ↓27*	*↑0 ↓28*	*↑0 ↓27*	*↑13 ↓174*
F_6	(30)	**↑1 ↓0**	*↑0 ↓25*	*↑0 ↓29*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑1 ↓204*
T_1	(35)	**↑20 ↓0**	*↑1 ↓24*	*↑1 ↓33*	*↑0 ↓34*	*↑0 ↓33*	*↑1 ↓32*	*↑1 ↓29*	*↑2 ↓28*	*↑26 ↓213*
T_2	(35)	**↑12 ↓4**	*↑10 ↓12*	*↑6 ↓21*	*↑2 ↓29*	*↑3 ↓29*	*↑2 ↓29*	*↑1 ↓27*	*↑1 ↓26*	*↑37 ↓177*
T_3	(35)	**↑11 ↓2**	*↑8 ↓11*	*↑7 ↓16*	*↑4 ↓25*	*↑4 ↓27*	*↑5 ↓24*	*↑3 ↓23*	*↑2 ↓22*	*↑44 ↓150*
T_4	(35)	**↑21 ↓3**	*↑5 ↓19*	*↑2 ↓29*	*↑0 ↓34*	*↑1 ↓34*	*↑1 ↓33*	*↑2 ↓32*	*↑4 ↓28*	*↑36 ↓212*
T_5	(35)	**↑9 ↓5**	**↑12 ↓7**	*↑7 ↓15*	*↑6 ↓24*	*↑5 ↓25*	*↑4 ↓24*	*↑4 ↓24*	*↑5 ↓21*	*↑52 ↓145*
T_6	(35)	**↑10 ↓0**	*↑5 ↓6*	*↑2 ↓24*	*↑1 ↓30*	*↑2 ↓31*	*↑1 ↓31*	*↑1 ↓29*	*↑2 ↓27*	*↑24 ↓178*
All	(270)	**↑96 ↓25**	*↑76 ↓86*	*↑60 ↓144*	*↑28 ↓205*	*↑30 ↓215*	*↑25 ↓207*	*↑24 ↓196*	*↑28 ↓179*	*↑367 ↓1257*

CDE only in very low change period environments, and the magnitudes of the improvements were small.

The results of this section indicate that the performance of DynPopDE is strongly dependent on the underlying function. DynPopDE performed better than CDE over a range of settings for the number of optima on the MPB (as proven in research questions 3 and 5). DynPopDE did, however, not scale well to other functions. This is a serious weakness of the algorithm, as the results in this section show that DynPopDE was generally outperformed by CDE on the majority of functions. CDE thus performed better than DynPopDE without fine-tuning CDE's parameter controlling the number of sub-populations for each function.

5.4 Comparison to Other Approaches

This section compares DynPopDE to the published results of other algorithms on variations of Scenario 2 of the MPB in terms of the number of peaks in the environments. Section 4.7.2 described four change detection strategies, that were incorporated into CDE, to ensure a fair comparison with results reported by other researchers. Two of these strategies, Det_{n_k-best} and $Det_{n_k-local}$, which only test for changes once every n_k generations, are not appropriate for incorporation into DynPopDE, since the number of sub-populations varies over time. A detection strategy that only tests for changes every n_k generations would thus be erratic, and would not ensure a uniform temporal sampling of the fitness landscape. DynPopDE was consequently tested using only the remaining two strategies, Det_{best} and Det_{local}.

Table 5.11 lists the offline errors and 95% confidence intervals of DynPopDE using the automatic change detection strategy, the Det_{best} detection strategy and the Det_{local} detection strategy. Several settings for the number of peaks were tested, but the other Scenario 2 settings (refer to Table 2.1) were left unchanged. A total of 500 000 function evaluations was performed for each of the 30 repeats on each of the environments.

The introduction of the detection strategies resulted in only minor changes in DynPopDE's average offline errors. The differences in offline error was statistically significant (according to a Mann-Whitney U test) in only one instance, which occurred when using

the Det_{local} detection strategy (printed in italics in Table 5.11).

Table 5.11: DynPopDE using various detection strategies

n_p	**Automatic**	Det_{best}	**p-val**	Det_{local}	**p-val**
1	0.53 ± 0.07	0.50 ± 0.06	0.562	0.52 ± 0.08	0.820
5	0.55 ± 0.04	0.65 ± 0.13	0.752	0.54 ± 0.04	0.719
10	0.82 ± 0.07	0.81 ± 0.07	0.775	0.76 ± 0.07	0.236
20	1.08 ± 0.06	1.13 ± 0.07	0.307	1.14 ± 0.07	0.286
30	1.31 ± 0.05	1.35 ± 0.07	0.676	1.38 ± 0.05	0.112
40	1.39 ± 0.05	1.44 ± 0.05	0.173	*1.48 ± 0.06*	0.031
50	1.58 ± 0.04	1.59 ± 0.08	0.654	1.56 ± 0.06	0.612
100	1.75 ± 0.06	1.82 ± 0.06	0.273	1.78 ± 0.05	0.719
200	1.83 ± 0.05	1.84 ± 0.05	0.959	1.85 ± 0.04	0.532

Table 5.12 lists the published results of 13 of the algorithms that were discussed in Section 3.3 on the variations of the MBP Scenario 2. The 95% confidence intervals were calculated from the reported standard errors or standard deviations in cases where the confidence interval was not reported. Each result was compared to the relevant DynPopDE result to determine which is better. Results were considered to be similar if the confidence intervals overlapped, i.e. neither algorithm was considered better than the other. Offline errors that were lower than the corresponding DynPopDE results are printed in italics. Offline errors that are higher than DynPopDE's (i.e. DynPopDE performed better) are printed in boldface in shaded cells.

Table 5.12: Reported offline errors on various numbers of peaks of various algorithms

n_p	**MMEO**	**HJEO**	**LSEO**	**CESO**	**ESCA**	**MPSO**	**CPSO**
1	**11.30 ± 6.98**	**7.08 ± 3.90**	**7.47 ± 3.88**	**1.04 ± 0.00**	**0.98 ± 0.00**	**5.07 ± 0.33**	*0.14 ± 0.03*
5	N/A	N/A	N/A	N/A	N/A	**1.81 ± 0.14**	0.72 ± 0.08
10	0.66 ± 0.39	*0.25 ± 0.20*	*0.25 ± 0.16*	**1.38 ± 0.04**	**1.54 ± 0.02**	**1.80 ± 0.12**	**1.06 ± 0.07**
20	0.90 ± 0.31	*0.39 ± 0.20*	*0.40 ± 0.22*	**1.72 ± 0.04**	**1.89 ± 0.08**	**2.42 ± 0.14**	**1.59 ± 0.06**
30	1.06 ± 0.27	*0.49 ± 0.18*	*0.49 ± 0.20*	*1.24 ± 0.02*	**1.52 ± 0.04**	**2.48 ± 0.14**	**1.58 ± 0.05**
40	1.18 ± 0.31	*0.56 ± 0.18*	*0.56 ± 0.18*	*1.30 ± 0.04*	**1.61 ± 0.04**	**2.55 ± 0.14**	1.51 ± 0.03
50	*1.23 ± 0.22*	*0.58 ± 0.18*	*0.59 ± 0.20*	*1.45 ± 0.02*	1.67 ± 0.04	**2.50 ± 0.12**	1.54 ± 0.03
100	*1.38 ± 0.18*	*0.66 ± 0.14*	*0.66 ± 0.14*	*1.28 ± 0.04*	*1.61 ± 0.06*	**2.36 ± 0.08**	*1.41 ± 0.02*
200	N/A	N/A	N/A	N/A	N/A	**2.26 ± 0.06**	*1.24 ± 0.02*
n_p	**HMSO**	**MSO**	**Cellular DE**	**Cellular PSO**	**SPSO**	**MPSO2**	**CDE**
1	**0.87 ± 0.05**	0.56 ± 0.04	**1.53 ± 0.07**	**5.23 ± 0.47**	N/A	N/A	**0.84 ± 0.11**
5	**1.18 ± 0.04**	**1.06 ± 0.06**	**1.50 ± 0.04**	**1.09 ± 0.22**	**1.98 ± 0.05**	N/A	0.55 ± 0.08
10	**1.42 ± 0.04**	**1.51 ± 0.04**	**1.64 ± 0.03**	**1.14 ± 0.13**	**1.98 ± 0.05**	**1.77 ± 0.05**	0.70 ± 0.11
20	**1.50 ± 0.06**	**1.89 ± 0.04**	**2.46 ± 0.05**	**2.20 ± 0.12**	N/A	N/A	**2.11 ± 0.19**
30	**1.65 ± 0.04**	**2.03 ± 0.06**	**2.62 ± 0.05**	**2.67 ± 0.13**	N/A	N/A	**2.74 ± 0.23**
40	**1.65 ± 0.05**	**2.04 ± 0.06**	**2.76 ± 0.05**	**2.70 ± 0.13**	N/A	N/A	**2.87 ± 0.29**
50	1.66 ± 0.02	**2.08 ± 0.02**	**2.75 ± 0.05**	**2.77 ± 0.13**	**3.47 ± 0.06**	N/A	**3.12 ± 0.20**
100	*1.68 ± 0.03*	**2.14 ± 0.02**	**2.73 ± 0.03**	**2.91 ± 0.14**	**3.60 ± 0.07**	N/A	**3.16 ± 0.24**
200	*1.71 ± 0.02*	**2.11 ± 0.03**	**2.61 ± 0.02**	**3.14 ± 0.12**	**3.47 ± 0.04**	**2.37 ± 0.03**	**3.42 ± 0.23**

The comparison of CDE to each of the tabulated algorithms is briefly discussed here.

MMEO [Moser and Hendtlass, 2007] MMEO is an algorithm based on extremal op-

timisation and searches for optima in parallel. The algorithm detects changes by re-evaluating all the best solutions in the search space and is thus comparable to the Det_{local} detection strategy. DynPopDE yielded a lower offline error than MMEO when a single peak was used. MMEO yielded a lower offline error than CDE on all other experiments. The confidence intervals of the two algorithms overlap for environments that used fewer than 50 peaks and MMEO, accordingly, cannot conclusively be considered superior to DynPopDE on these environments.

HJEO [Moser, 2007] HJEO is an extension of MMEO which incorporates a Hooke-Jeeves local search. HJEO performed better than DynPopDE on all reported cases, except when a single peak was present in the environment, in which case DynPopDE performed better than HJEO.

LSEO [Moser and Chiong, 2010] The local search component of MMEO was further improved in the LSEO algorithm. LSEO performed better than CDE on all reported cases, except when a single peak was present in the environment, in which case DynPopDE performed better than LSEO.

CESO [Lung and Dumitrescu, 2007] CESO uses a single DE population and a single PSO population to solve DOPs. Changes are detected by re-evaluating the best individual in the DE population and is thus comparable to the Det_{best} detection strategy. DynPopDE performed better than CESO in all environments that had fewer than 30 peaks. CESO performed better than DynPopDE when 30 or more peaks are present in the environment.

ESCA [Lung and Dumitrescu, 2010] ESCA is an adaptation of CESO which employs two DE populations in combination with the PSO population. DynPopDE performed better than ESCA in all environments that had less than 50 peaks. ESCA performed better than DynPopDE when 50 or more peaks are present in the environment.

MPSO [Blackwell and Branke, 2006] MPSO is a multi-swarm algorithm based on PSO. MPSO uses the Det_{local} detection strategy, and is consequently comparable to DynPopDE using the Det_{local} detection strategy. DynPopDE performed better than MPSO in all reported cases.

CPSO [Yang and Li, 2010] CPSO clusters particles of a PSO algorithm into sub-swarms to track optima in a dynamic environment. CPSO detects changes using the Det_{best} detection strategy and is thus roughly comparable to DynPopDE using the Det_{best} strategy. CPSO performed better than DynPopDE when a single peak was used, but overlapping confidence intervals were found when five peaks were present in the environment. DynPopDE performed better than CPSO when 10, 20 and 30 peaks were present. Overlapping confidence intervals were found when using 40 and 50 peaks. CPSO performed better than DynPopDE when 100 and 200 peaks were present in the environment.

HMSO [Kamosi *et al.*, 2010a] HMSO is an extension of MPSO which hibernates unproductive sub-swarms. HMSO uses the Det_{best} change detection strategy, and can be compared to DynPopDE using Det_{best}. DynPopDE performed better than HMSO in all experiments with fewer than 50 peaks. Overlapping confidence intervals were found when using 50 peaks, while HMSO performed better than DynPopDE in environments with 100 and 200 peaks.

MSO [Kamosi *et al.*, 2010b] MSO is a PSO-based algorithm that utilises a dynamic number of swarms to locate optima in a dynamic environment. Kamosi *et al.* [2010b] do not specify the change detection strategy that is used in MSO, but it is assumed that the Det_{best} strategy that was used in HMSO (which was developed by the same authors) is also used in MSO. DynPopDE performed better than MSO in all reported cases.

Cellular DE [Noroozi *et al.*, 2011] Cellular DE creates sub-populations by dividing the search space into equally sized cells. Cellular DE employs the Det_{local} change detection strategy and is thus compared to DynPopDE using the same strategy. DynPopDE performed better than Cellular DE in all reported cases.

Cellular PSO [Hashemi and Meybodi, 2009a] Cellular PSO creates sub-swarms by dividing the search space into equally sized cells. Cellular PSO uses the Det_{local} change detecting strategy and is thus roughly comparable to DynPopDE using the Det_{local} strategy. DynPopDE performed better than Cellular PSO in all cases.

SPSO [Li *et al.*, 2006] SPSO is a multi-swarm PSO based algorithm designed specifically for situations where the number of peaks is unknown. SPSO detects changes by re-evaluating the five best particles, and is thus comparable to DynPopDE using the Det_{local} strategy. DynPopDE performed better than SPSO in all reported cases.

MPSO2 [Blackwell, 2007] MPSO2 is a multi-swarm PSO based algorithm which self-adapts the number of swarms. MPSO2 detects changes using the Det_{local} strategy. DynPopDE performed better than MPSO2 in both the reported environments.

CDE The Det_{best} detection strategy was used in CDE to determine its performance on the environments used in this section. DynPopDE performed better than CDE when a single peak was used, and when more then 10 peaks were present in the environment. Overlapping confidence intervals were found when using 5 and 10 peaks.

This section compared DynPopDE results with the reported results of other algorithms. DynPopDE performed better than each of the discussed algorithms on at least one of the experimental environments. DynPopDE yielded better results on all environments than several of the algorithms (MPSO, Cellular DE, Cellular PSO, SPSO and MPSO2). The results presented in this section indicate that CDE compares favourably with the state-of-the-art algorithms in the literature.

5.5 Conclusions

This chapter introduced a new algorithm, named DynPopDE, which is aimed at problems in which the number of optima is unknown or fluctuating. DynPopDE spawns and removes sub-populations in order to find an effective number of sub-populations. A sub-population is created when the algorithm detects that all existing sub-populations have stagnated. The sub-population removal process is incorporated into the exclusion component, which removes a sub-population when it strays too close to another sub-population. DynPopDE has the advantage that it removes the need to tune the parameter controlling the number of sub-populations.

The performances of DynDE, CDE and DynPopDE were investigated on problems where the number of optima are unknown or unknown and fluctuating over time. The

extended MPB was used to simulate environments in which the number of optima fluctuate over time.

The first major result that was found in this chapter is that it is typically not advisable to use the same number of populations as the number of optima in DynDE and CDE when the number of populations is large. Better results were obtained when a relatively small number of populations was used. Knowing the number of optima in an environment in advance is thus generally not beneficial.

Research questions 2 and 3 investigated environments with various numbers of optima. CDE performed significantly better than DynDE over a wide range of experiments when the number of optima were unknown. DynPopDE generally performed better than DynDE and CDE on these environments, but was typically less effective than CDE in high dimensional environments. DynPopDE creates more sub-populations when a large number of optima is present than when a small number of optima is present. Experimental results showed that high dimensions and low change periods diminish DynPopDE's ability to distinguish effectively between environments with different numbers of optima.

Research questions 4 and 5 focused on environments in which the number of optima fluctuate over time. Fluctuating the number of optima during the course of the optimisation process had a negative effect on the results of DynDE and CDE. CDE did, however, still perform significantly better than DynDE. The experimental results showed that DynPopDE performed better than CDE when the number of optima fluctuates. Three distinct phases, with respect to change period, were identified in DynPopDE's comparative performance. DynPopDE performed better than CDE in phases one and three, which occur at very low and very high change periods. DynPopDE performed similarly or worse than CDE during phase two, which occurs at intermediate change periods. The range of change periods representing phase two was found to increase as the number of dimensions increases.

The spawning and removal processes used in DynPopDE were compared to the approaches used in MPSO2. The algorithm that used MPSO2's sub-population creation and removal components performed significantly worse than DynPopDE over a wide range of fluctuating number of peak experiments. The processes used in MPSO2 are thus not appropriate for use in conjunction with CDE.

DynPopDE was found to compare favourably to the reported results of several state-of-the-art algorithms on various settings of number of peaks on the MPB.

The comparison of DynPopDE and CDE on the standard set of experiments used in Chapter 4 showed that DynPopDE does not scale well to other functions. Despite the fact that DynPopDE performed well on a wide range of numbers of optima on the MPB functions, DynPopDE's poor performance on the GDBG functions leads to the conclusion that DynPopDE is not a generally applicable algorithm.

The following chapter investigates the incorporation of self-adaptive control parameters into CDE and DynPopDE.

Chapter 6

SELF-ADAPTIVE CONTROL PARAMETERS

The previous chapter dealt with adapting the number of sub-populations during the optimisation process. This chapter describes the incorporation of self-adaptive control parameters into CDE and DynPopDE. Variations of approaches from the literature are used to self-adapt the scale and crossover factors. Novel approaches are suggested to self-adapt the Brownian radius. The various self-adaptive approaches are empirically compared. Empirical evidence is presented to prove that the self-adaptive approaches do improve CDE and DynPopDE.

6.1 Introduction

The discussion in Section 3.4.1.5 concluded that a disadvantage of the DynDE algorithm is that it has several parameters that influence its performance, and may have to be manually tuned for best results. These parameters are: the size of sub-populations, the number of sub-populations, the number of Brownian individuals per sub-population, and the parameter r_{brown}, which is used to create Brownian individuals. DynDE also contains the scale and crossover factor parameters of DE. All these are also parameters of CDE.

The various parameters have, in part, been addressed in previous chapters. Section 4.6.3 empirically determined appropriate values for the sub-population size and the number

of Brownian individuals. Chapter 5 dealt extensively with DynPopDE, an algorithm that adapts the number of sub-populations, hence removing it as a parameter. The focus of this chapter is on self-adapting the scale and crossover factors, as well as the Brownian radius. The previous chapter found that DynPopDE does not scale well to different functions, and CDE is consequently used as the base algorithm for the investigations into the self-adaptive (SA) control parameters, in this chapter.

The algorithms that were selected from the literature for adapting the scale and crossover factors are discussed in Section 6.2. All the selected algorithms were originally discussed in Section 2.4.4. Section 6.2 also contains a description of alterations made to the selected algorithms, which are investigated as potentially more effective in dynamic environments. A total of 13 approaches to self-adapting the scale and crossover factors are identified. Section 6.3 presents a novel algorithm for adapting the Brownian radius used when creating Brownian individuals. A total of four alternative implementations of the new approach is presented.

The experimental results of this chapter are provided in Section 6.4. This section lists the specific research questions that were investigated and gives the experimental procedure followed to answer each research question. The research questions identify the most effective self-adaptive approach for the scale and crossover parameters, out of the 13 approaches that were investigated. Furthermore, the most effective self-adaptive approach for the Brownian radius is identified from the four approaches that were investigated. These two approaches are combined to form self-adaptive competing differential evolution (SACDE).

The research questions include a scalability study to observe the scaling behaviour of the self-adaptive algorithms with respect to factors such as change period, number of dimensions and underlying function. The research questions further investigate the incorporation of the self adaptive approaches into DynPopDE to form SADynPopDE.

The performance of SACDE is compared to the performance of other algorithms on the benchmark environments in Section 6.5. Conclusions from this chapter are presented in Section 6.6.

6.2 Adapting the Scale and Crossover Factors

Several approaches to self-adapting or eliminating DE control parameters can be found in the literature (refer to Section 2.4.4). The purpose of this section is to identify potential self-adaptive approaches that can be incorporated into CDE. Section 6.2.1 reviews the algorithms selected for incorporation into CDE. Alternative implementations of the selected algorithms are described in Section 6.2.2.

6.2.1 Selected Approaches

Several approaches to self-adapting or eliminating control parameters, specific to DE, were reviewed in Section 2.4.4. Three of these approaches were selected for incorporation into CDE and are discussed below. The reasons for selecting these algorithms are:

- they can all be incorporated into CDE with relatively few changes,
- these algorithms are some of the more recent SA approaches for DE,
- all performed well in static environments, and
- these algorithms do not introduce a large number of parameters that are obviously problem-dependent.

The first approach selected is that of Omran *et al.* [2005a], called SDE, discussed in Section 2.4.4. SDE associates a scale factor with each individual in the population. Before a new trial vector is created, the scale factor to be used is calculated from the scale factors of three randomly selected individuals, using equation (2.18). This new scale factor is then associated with the newly created trial individual. The initial scale factors that are used at the commencement of the algorithm are selected from a normal distribution, $N(0.5, 0.15)$. The crossover factor is selected from a normal distribution, $N(0.5, 0.15)$, for each crossover and thus is not self-adaptive.

The second approach to be incorporated into CDE is that of Brest *et al.* [2006], discussed in Section 2.4.4. This algorithm self-adapts both the crossover and the scale factors. The crossover and scale factors from the target individual are used in equations (2.19) and (2.20) to find crossover and scale factors for the trial individual. This approach is the

basis of the successful *jDE* algorithm [Brest *et al.*, 2009] which is aimed at dynamic environments. Brest *et al.* [2009] used $\mathcal{F}_l = 0.36$ rather than $\mathcal{F}_l = 0.1$ (refer to Section 3.4.2.1) for *jDE*. Brest *et al.* [2009] used initial values of 0.9 for the crossover factor and 0.5 for the scale factor.

The third approach selected for incorporation into CDE is the Barebones DE [Omran *et al.*, 2009], discussed in Section 2.4.4. This is a hybrid PSO and DE algorithm that eliminates the scale factor. Trial individuals are created from the weighted average of an individual's personal best position and the global best position using equation (2.26). A crossover is then performed between the trial individual and the personal best position of a randomly selected individual. The hybrid of the Barebones DE with CDE is referred to as BBSA. This algorithm does not make use of a self-adaptive component, but rather eliminates the scale factor. BBSA is, however, discussed within the category of self-adaptive algorithms, for the sake of brevity, in the rest of this chapter.

6.2.2 Alternative Techniques

This section describes the alterations made to the algorithms described in the previous section. The goal of using the alterations is to investigate alternative techniques of applying the SA algorithms to DOPs. The alternative techniques relate to the DE scheme, the response to changes in the environment, and the initial values that are used at the commencement of the algorithm.

SDE makes use of the DE/rand/1/bin scheme, while CDE uses DE/best/2/bin. Two versions of SDE are incorporated into CDE and compared. The first, SSA1, makes use of DE/rand/1/bin as in SDE. The second, SSA2, calculates scale and crossover factors as in SDE, but uses DE/best/2/bin to create trial individuals.

jDE makes use of DE/rand/1/bin. Consequently, two versions of this self-adaptive technique are considered. The first, jSA1, is the self-adaptive version of *jDE*. The second, jSA2, uses the *jDE* approach, but with DE/best/2/bin.

Brest *et al.* [2009] successfully used their self-adaptive approach in *jDE* to solve DOPs. This self-adaptive approach was implemented in the same way as for static environments. No special measures were taken to customise the approach for dynamic environments. This section considers alternative strategies to incorporate self-adaptation into CDE. Figures

which show the convergence profile of the scale and crossover factors are used to aid the discussion.

The average scale and crossover factor profiles of jSA2 on the Scenario 2 settings of the MPB (refer to Section 2.5.3) are given in Figure 6.1 for 10 changes in the environment and in Figure 6.2 for 100 changes in the environment. Averages over the scale and crossover factors of all individuals over 30 repeats are shown. It is evident from Figure 6.2 that, over a long period, the scale and the crossover factors stabilise to values around 0.63 and 0.52 respectively. However, these stabilised values are only reached after a period of about 100 000 iterations. During the initial period before stabilisation, the overall crossover factor slowly decreases and the overall scale factor slowly increases.

Figure 6.1 shows that, directly after a change in the environment, a sudden decrease in the value of the average crossover factor and a sudden increase in the value of the average scale factor occurs. For the rest of the period before the next change in the environment, the crossover factor slowly increases while the scale factor slowly decreases. This cycle continues until equilibrium is reached at the stabilisation point where increases are roughly matched by decreases in scale and crossover factors.

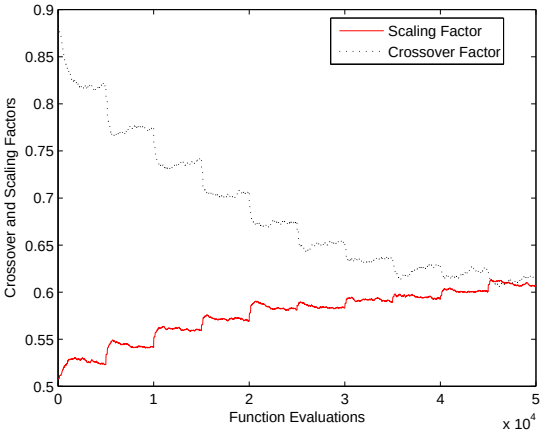

Figure 6.1: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2.

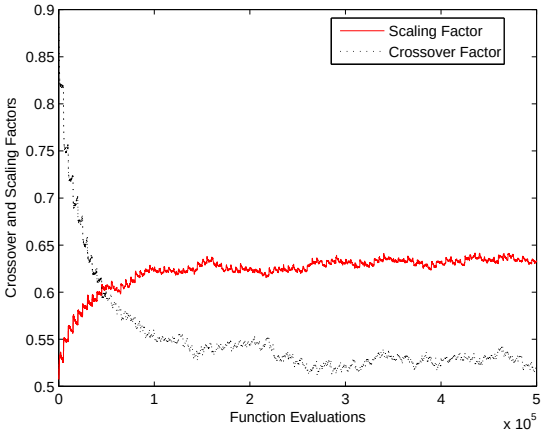

Figure 6.2: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2.

The scale and crossover profiles give evidence that jSA2 does respond to changes in the environment. The argument can be made that the initial high value of the crossover factor and initial low value of the scale factor corresponds to a period where jSA2 discovers

optima in the environment. The eventual lower value of the crossover factor and slightly higher value of the scale factor corresponds to a period when jSA2 no longer discovers new optima but merely tracks optima that have already been discovered. In other words, the adaptation trends exist because the algorithm is successfully adapting the scale and crossover factors to appropriate values at different stages of the evolution process.

Alternatively, the trends in scale and crossover factor profiles in Figures 6.1 and 6.2 could be the result of poorly selected initial values. The initial values of 0.9 for the crossover factor and 0.5 for the scale factor may be entirely inappropriate and may cause the prolonged initial period when the crossover factor decreases and the scale factor increases. If initial values are indeed ineffective, then the algorithm can potentially be improved by rather selecting initial values randomly as advocated by the creators of SDE who suggested initial values from a normal distribution, $N(0.5, 0.15)$ [Omran *et al.*, 2005a].

The two algorithms that use random initial values are referred to as jSA1Ran and jSA2Ran. The convergence behaviour that results from the random initial values of jSA2Ran is shown in Figures 6.3 and 6.4 for 10 and 100 changes, respectively, on the MPB Scenario 2. This algorithm exhibits much less variation in the value of the crossover factor than was observed in Figure 6.2. The scale factor still increases over the first 100 000 function evaluations.

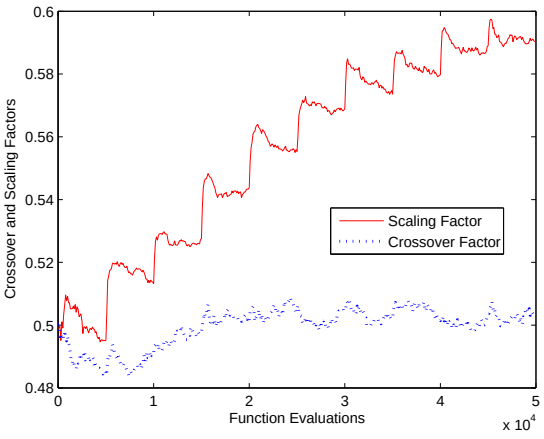

Figure 6.3: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2, when using random initial values.

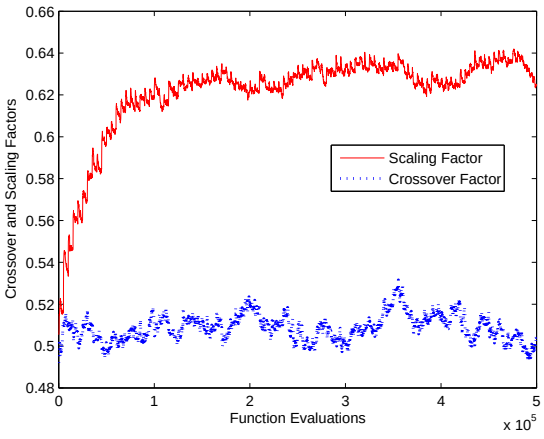

Figure 6.4: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2, when using random initial values.

Another alternative interpretation of the trends visible in Figures 6.1 and 6.2 is that the initial drop in crossover factor and increase in scale factor, after a change in the environment, negatively affects the values of the scale and crossover factors for the rest of the period before the next change in the environment. Ideal values for the scale and crossover factor may be respectively lower and higher, but the algorithm may adapt the factors by too much when a change in the environment occurs, and may be unable to undo the initial adaptation before the next change in the environment. Should this interpretation be correct, then the algorithm can be potentially improved by resetting the scale and crossover factors to initial values every time a change in the environment occurs.

Figures 6.5 and 6.6 show the scale and crossover value profiles of jSA2 when the factors are reset when a change in the environment occurs. The average value of the scale factor is kept considerably higher than was the case in Figures 6.1 and 6.2. The crossover factor repeatedly increases to a value similar to that in Figures 6.1 and 6.2 after each change in the environment.

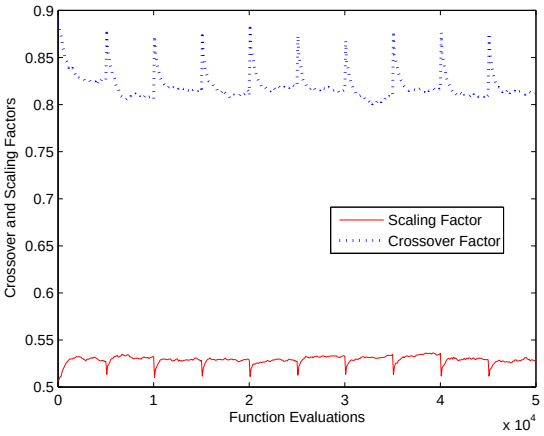

Figure 6.5: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2, when the factors are reset after changes in the environment.

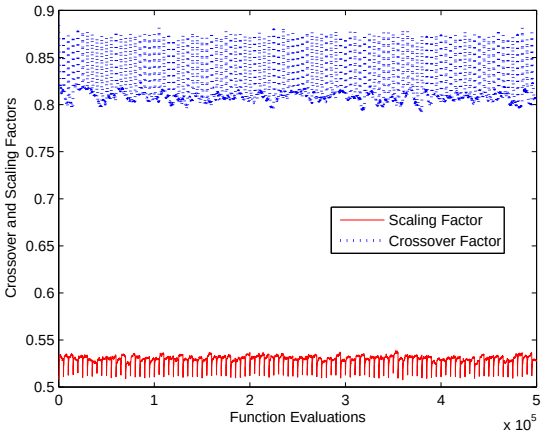

Figure 6.6: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2, when the factors are reset after changes in the environment.

Resetting the scale and crossover factors when a change in the environment occurs, is applicable to both *jDE* and SDE. The two SDE-based algorithms that use resetting are referred to as SSA1Res and SSA2Res. Two strategies are tested for the *jDE*-based algorithms, the first, resetting the factors to the values recommended by Brest *et al.* [2009]

(jSA1Res and jSA2Res), and the second, resetting the values from a normal distribution, $N(0.5, 0.15)$ as suggested by Omran *et al.* [2005a] (jSA1RanRes and jSA2RanRes).

Figures 6.7 and 6.8 show the scale and crossover value profiles of jSA2 when the factors are set to values from a normal distribution, $N(0.5, 0.15)$, when a change in the environment occurs. The scale factor increases sharply after being reset, but never reaches as high a value as it did in Figures 6.1 and 6.2. The value of the crossover factor is more chaotic than it was in previous strategies, and fluctuates around a lower value than previously.

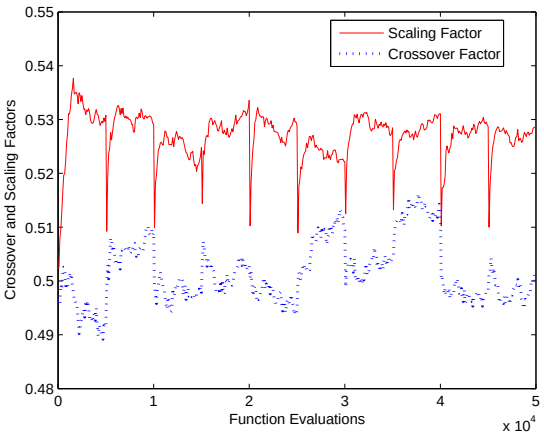

Figure 6.7: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2, when the factors are set to random values after changes in the environment.

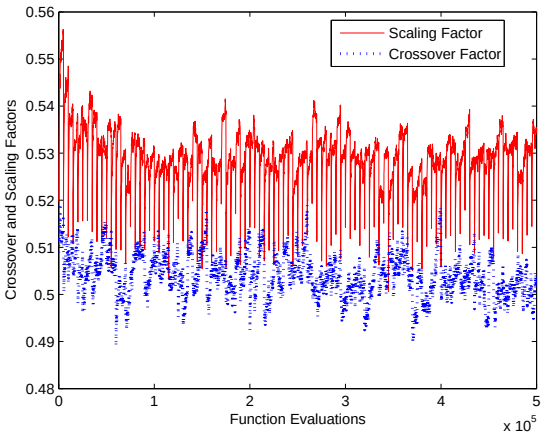

Figure 6.8: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2, when the factors are set to random values after changes in the environment.

6.2.3 Summary of Algorithms for Adapting the Scale and Crossover Factors

The previous sections described several algorithms for adapting the scale and crossover factors, which were incorporated into CDE. Each algorithm is thus a hybrid between CDE and a self-adaptive approach. These algorithms are summarised below.

SSA1: The algorithm of Omran *et al.* [2005a], which uses the DE/rand/1/bin scheme.

SSA2: The algorithm of Omran *et al.* [2005a], but using the DE/best/2/bin scheme.

SSA1Res: The algorithm of Omran *et al.* [2005a], which uses the DE/rand/1/bin scheme, in which the factors are reselected from a normal distribution after a change in the environment.

SSA2Res: The algorithm of Omran *et al.* [2005a], but using the DE/best/2/bin scheme, in which the factors are reselected from a normal distribution after a change in the environment.

jSA1: The algorithm of Brest *et al.* [2009], employed in *jDE*, which uses the DE/rand/1/bin scheme.

jSA2: The algorithm of Brest *et al.* [2009], but using the DE/best/2/bin scheme.

jSA1Ran: The algorithm of Brest *et al.* [2009], which uses the DE/rand/1/bin scheme, but with initial values selected from a normal distribution, as proposed by Omran *et al.* [2005a].

jSA2Ran: The algorithm of Brest *et al.* [2009], but using the DE/best/2/bin scheme, with initial values selected from a normal distribution, as proposed by Omran *et al.* [2005a].

jSA1Res: The algorithm of Brest *et al.* [2009], which uses the DE/rand/1/bin scheme, in which the factors are reset to their initial values after a change in the environment.

jSA2Res: The algorithm of Brest *et al.* [2009], but using the DE/best/2/bin scheme, in which the factors are reset to their initial values after a change in the environment.

jSA1RanRes: The algorithm of Brest *et al.* [2009], which uses the DE/rand/1/bin scheme, but with initial values selected from a normal distribution, in which the factors are reselected from a normal distribution after a change in the environment.

jSA2RanRes: The algorithm of Brest *et al.* [2009], but using the DE/best/2/bin scheme, with initial values selected from a normal distribution, in which the factors are reselected from a normal distribution after a change in the environment.

BBSA: The algorithm of Omran *et al.* [2009].

The 13 algorithms, summarised above, are experimentally compared in Section 6.4 to determine which is the most effective approach to use in conjunction with CDE in dynamic environments.

6.3 Adapting the Brownian Radius

The creation of Brownian individuals was discussed in Section 3.4.1.3. The purpose of the Brownian individuals is to increase the diversity within sub-populations, so that the DE process can respond effectively to changes in the environment. A secondary advantage of using Brownian individuals is that it acts as a random local search around the best individual in the sub-population, which helps to reduce the error. A self-adaptive approach is suggested in this section to eliminate the need for tuning the Brownian radius parameter. The new approach is presented in Section 6.3.1, while alternative implementations to the proposed approach are presented in Section 6.3.2.

6.3.1 Proposed Approach

Section 3.4.1.3 gave the equation used to create Brownian individuals and specified that a value for the Brownian radius of $r_{brown} = 0.2$ has been found to produce good results. Intuitively, it seems unlikely that a single value for r_{brown} would produce the best results at all stages of the optimisation process, for example, more diversity would likely be required after a change in the environment. Furthermore, the Brownian radius is likely to be function-dependent.

This section proposes a novel approach to self-adapting the Brownian radius to appropriate values for different functions and different stages of the optimisation process. The new approach relies on the assumption that the Brownian radius should initially be large, as new optima need to be discovered. Appropriate values for the Brownian radius, $r_{brown}(t)$, can later be deduced from radii used by successful Brownian individuals that resulted in lower error values. The Brownian radius, $r_{brown}(t)$, is set equal to the absolute value of a random number selected from a normal distribution:

$$r_{brown}(t) \quad \sim \quad | \, N(0, r_{dev}(t)) \, | \tag{6.1}$$

where $r_{dev}(t)$ is the standard deviation of the normal distribution, and the actual value that is self-adapted.

The proposed self-adaptive approach commences with a Brownian radius proportional to the sub-population radius of the first sub-population that is evaluated. The population radius, $r_{pop,k}$, is calculated for sub-population P_k using:

$$r_{pop,k} = \frac{\max_{i_1,i_2 \in \{1,\ldots,n_{I,k}\}} \left\{ \|\vec{x}_{i_1} - \vec{x}_{i_2}\|_2 \right\}}{2} \tag{6.2}$$

The initial value of $r_{dev}(t)$ is thus half of the maximum Euclidian distance between any two individuals in the sub-population, i.e. $r_{dev}(0) = r_{pop,k}$, where P_k is the first sub-population that is evaluated. The algorithm keeps track of the $r_{dev}(t)$ values of all Brownian individuals that have a higher fitness than the best individual within the sub-population, $\vec{x}_{best,k}(t)$, that was used in their creation. An average of all successful r_{dev} values are used to calculate successive values for r_{dev}. Algorithm 16 formally describes the new approach (assuming a function minimisation problem).

Algorithm 16: Self-adaptive Brownian radius algorithm

$t = 0$;
$DevSum = r_{pop,0}$ with P_0 being the first sub-population evaluated;
$DevCount = 1$;
foreach *Brownian individual that to be created in sub-population* P_k **do**
  $r_{dev}(t) = DevSum/DevCount$;
  $r_{brown}(t) \sim \mid N(0, r_{dev}(t)) \mid$;
  $\vec{x}_{brown} = \vec{x}_{best,k} + \vec{r}$ with $r_j \sim N(0, r_{brown}(t))$;
  if $F(\vec{x}_{brown}) < F(\vec{x}_{best,k})$ **then**
    $DevSum = DevSum + r_{dev}(t)$;
    $DevCount = DevCount + 1$;
  end
  $t = t + 1$;
end

The proposed algorithm uses the average of successful values of r_{dev} as the standard deviation of the normal distribution to select the next r_{dev} value. Smaller values than the average r_{dev} value are thus more likely to be produced, but larger values will also be produced by the normal distribution.

6.3.2 Alternative Techniques

Two alternative techniques of implementing the self-adaptive Brownian radius algorithm are presented here. The first technique pertains to the distribution used to create the value of $r_{dev}(t)$. The second technique involves a response to changes in the environment.

Equation (6.1) selects the new value for $r_{dev}(t)$ from a normal distribution. About 68.27% of values selected from a normal distribution fall within one standard deviation from the mean value, and 99.73% of values fall within three standard deviations of the mean. This means, in the context of equation (6.1), that the next selected value of $r_{brown}(t)$ would likely be smaller than the current value of $r_{dev}(t)$, and cases where $r_{brown}(t) > 3r_{dev}(t)$ are extremely unlikely. The propensity for selecting small values for $r_{brown}(t)$ could potentially negatively affect the performance of the algorithm, as too little diversity may be introduced into the sub-populations by the Brownian individuals.

The Cauchy distribution is a distribution function with two parameters, the location of the median and a scale value. A random number from a Cauchy distribution, in contrast to a random number from a normal distribution, is more likely to be greater than the scale value. For example, only 50% of values selected from a Cauchy distribution fall within one scale value from the median, while less than 80% of values fall within three scale values from the median.

The use of a Cauchy distribution, rather than a normal distribution, to select new values for $r_{brown}(t)$, would thus result in larger values being selected more frequently. This could potentially avoid the problem of low sub-population diversity which could result from using a normal distribution. This section thus proposes to use a Cauchy distribution to calculate the values for $r_{dev}(t)$, as follows:

$$r_{brown}(t) \sim \mid C(0, r_{dev}(t)) \mid \tag{6.3}$$

The algorithm that utilises a normal distribution is referred to as SABrNor and the algorithm that uses a Cauchy distribution is referred to as SABrCau.

Section 6.2.2 described an alternative implementation of the self-adaptive algorithms for adapting the scale and crossover factors, which resets the parameters to initial values when a change in the environment occurs. A similar resetting scheme is proposed here whereby the value of $r_{dev}(t)$ is reset to the original population radius, and the average over successful values is cleared. Algorithm 16 is consequently changed into Algorithm 17.

Algorithm 17: Self-adaptive Brownian radius algorithm with resetting

$t = 0$;
$DevSum = r_{pop,0}$ with P_0 being the first sub-population evaluated;
$DevCount = 1$;
foreach *Brownian individual that to be created in sub-population* P_k **do**
  if *A change in the environment occurred* **then**
    $DevSum = r_{pop,0}$;
    $DevCount = 1$;
  end
  $r_{dev}(t) = DevSum/DevCount$;
  $r_{brown}(t) \sim \mid N(0, r_{dev}(t)) \mid$;
  $\vec{x}_{brown} = \vec{x}_{best,k} + \vec{r}$ with $r_j \sim N(0, r_{brown}(t))$;
  if $F(\vec{x}_{brown}) < F(\vec{x}_{best,k})$ **then**
    $DevSum = DevSum + r_{dev}(t)$;
    $DevCount = DevCount + 1$;
  end
  $t = t + 1$;
end

Resetting of $r_{dev}(t)$ can also be used in conjunction with a Cauchy distribution. The algorithm that utilises a normal distribution with resetting is referred to as SABrNorRes and the algorithm that uses a Cauchy distribution with resetting is referred to as SABrCauRes.

6.3.3 Summary of Algorithms for Adapting the Brownian radius

The previous sections introduced four algorithms for adapting the Brownian radius. These algorithms are summarised below:

SABrNor: This algorithm uses a normal distribution to select values for $r_{dev}(t)$.

SABrCau: This algorithm uses a Cauchy distribution to select values for $r_{dev}(t)$.

SABrNorRes: This algorithm uses a normal distribution to select values for $r_{dev}(t)$, and resets $r_{dev}(t)$ to the original population radius when a change in the environment occurs.

SABrCauRes: This algorithm uses a Cauchy distribution to select values for $r_{dev}(t)$, and resets $r_{dev}(t)$ to the original population radius when a change in the environment occurs.

Experimental comparisons are performed in Section 6.4 on the four algorithms.

6.4 Experimental Results

The experimental investigations into the proposed self-adaptive control parameter approaches are described in this section. The investigations are guided by the research questions listed below:

1. *Which one of the 13 algorithms for self-adapting the scale and crossover factors is the most effective?* The 13 algorithms are compared to CDE to determine which provides the most improved results.

2. *How does the algorithm identified in the previous research question compare to DynDE and CDE on the set of environments used in Chapter 4?* The algorithm found to perform the best in the previous research question is compared to DynDE and CDE on a broad set of environments. Specific environments where use of the self-adaptive approach is effective, are identified.

3. *Which one of the four algorithms, for self-adapting the Brownian radius, is the most effective?* The four algorithms are compared to CDE to determine which provides the most improved results.

4. *How does the algorithm identified in the previous research question compare to DynDE and CDE on the set of environments used in Chapter 4?* The best algorithm for self-adapting the Brownian radius is compared to DynDE and CDE on a broad set of environments. Specific environments on which use of the self-adaptive approach is effective, are identified.

5. *How does the combination of the algorithms for adapting the scale and crossover factors and the Brownian radius, SACDE, compare to DynDE and CDE?* The two algorithms that were identified in research questions 1 and 3 are combined to form an algorithm that self-adapts the scale and crossover factors and the Brownian radius. The new algorithm is referred to as SACDE. SACDE is experimentally compared to DynDE and CDE.

6. *How do the self-adaptive algorithms scale under factors that influence the complexity of a dynamic optimisation problem?* This scalability study analyses the performance of the algorithms when varying combinations of the change period, number of dimensions, change severity, underlying function, and change type.

7. *What are the convergence profiles of the self-adaptive algorithms?* The convergence behaviours of SACDE and its predecessors are compared to DynDE and CDE in terms of diversity, current error, and the resulting offline error. The convergence profiles of the self-adapted values are also investigated.

8. *How do the self-adaptive components affect the performance of DynPopDE?* The self-adaptive components, which were used to create SACDE, are incorporated into DynPopDE to create SADynPopDE. This algorithm is evaluated on various experimental sets and compared to DynPopDE, CDE and SACDE.

The rest of this section is structured as follows: Section 6.4.1 gives the experimental procedure that was followed to answer each of the research questions. Sections 6.4.2 to 6.4.9 respectively cover research questions 1 to 8.

6.4.1 Experimental Procedure

This chapter uses the sets of environments defined in previous chapters to evaluate the self-adaptive approaches. Research question 1 is answered by evaluating the 13 algorithms on the set of environments used in Section 4.6.3 to determine the appropriate sub-population size and number of Brownian individuals. The standard set of experiments, defined in Section 4.6.2 on page 120 was varied for each of the change periods and numbers of dimensions listed in Table 4.3. Each of the 13 algorithms was consequently evaluated on a total of 648 environments.

Research question 2 uses the set of environments used in Chapter 4 to compare CDE to its subcomponents. This set is created by varying the standard set for all combinations of settings in Table 4.3. The algorithm evaluated in research question 2 is thus tested on 2 160 environments.

The same experimental environments that were used in research question 1 are also used to answer research question 3. Each of the four approaches to self-adapting the Brownian radius was tested on the 648 environments created by varying the standard set of experiments for each of the change periods and numbers of dimensions in Table 4.3.

Research questions 4, 5, 6, 7 and 8 employ the same environmental set as research question 2. The 2 160 environments described in Section 4.6.2 are used to evaluate the comparative performance of the algorithms.

Research question 8 also uses the n_p and the $n_p(t)$ standard sets defined in Sections 5.3.1.1 and 5.3.1.2, respectively, to evaluate the performance of the relevant algorithms on problems with various numbers of optima and fluctuating numbers of optima. The n_p standard set consists of 480 environments, while the $n_p(t)$ standard set consists of a total of 2 000 environments.

A stopping criterion of 60 changes in the environment was used for all experiments. The offline error was used as the performance measure for all environments. Experiments were repeated 30 times to facilitate drawing statistically valid conclusions from the results. Mann-Whitney U tests were used to test statistical significance when comparing algorithms.

6.4.2 Research Question 1

Which one of the 13 algorithms for self-adapting the scale and crossover factors is the most effective?

Section 6.2 outlined 13 algorithms for self-adapting the scale and crossover factors. The focus of this section is to determine which of the algorithms produces the best results. Each of the algorithms was executed on the 648 environments described in Section 6.4.1, and the results compared to those of CDE. Three metrics were used to gauge the comparative performance of the algorithms. The first is the number of experiments in which an algorithm performed statistically significantly better than CDE (**Nr Better**). The second is the number of times that CDE performed statistically significantly better than the algorithm (**Nr Worse**). The third is the average percentage improvement (**API**, calculated as in equation (4.6) on page 148) of the algorithm over CDE.

Table 6.1 lists the results of the 13 algorithms. The column labelled "**Difference**" gives the number of times that each algorithm performed better than CDE, minus the number of times that the algorithm performed worse than CDE. Only algorithms that used the DE/best/2/bin scheme performed better more often than CDE. The cases where the average percentage increase was positive are also isolated to algorithms that used the DE/best/2/bin scheme. The superiority of the DE/best/2/bin scheme is to be expected, as Mendes and Mohais [2005] showed that this scheme is the most appropriate to use in DynDE, the base algorithm of CDE.

The jSA2Ran performed better than CDE most often and yielded the highest API value, followed closely by jSA2. This leads to the conclusion that these algorithms are not very sensitive to the initial values that are used, as jSA2 commences with the values suggested by Brest *et al.* [2009], and jSA2Ran commences with values sampled from a normal distribution.

Algorithms in which the scale and crossover factors were reset did not perform better than CDE as often as jSA2 and jSA2Ran, but jSA2RanRes was outperformed by CDE the fewest times of all algorithms, and yielded an API only slightly lower than jSA2 and jSA2Ran. SSA2Res performed the best of all the approaches based on SDE.

The algorithm with the largest difference between the number of times it outperformed

Table 6.1: Performance of SA scale and crossover algorithms vs CDE

Algorithm	Nr Better	Nr Worse	Difference	API
SSA1	33	380	-347	-8.43 %
SSA2	70	88	-18	0.40 %
SSA1Res	42	378	-336	-7.71 %
SSA2Res	111	54	57	1.78 %
jSA1	83	366	-283	-6.45 %
jSA2	204	93	111	4.54 %
jSA1Ran	85	335	-250	-5.69 %
jSA2Ran	207	72	135	4.92 %
jSA1Res	68	390	-322	-8.04 %
jSA2Res	166	130	36	3.12 %
jSA1RanRes	81	339	-258	-6.16 %
jSA2RanRes	179	49	130	4.29 %
BBSA	11	521	-510	-19.73 %

CDE and the number of times it was outperformed, is jSA2Ran. This algorithm is thus selected as the best approach out of those investigated, as jSA2Ran also had the highest average percentage increase value. jSA2Ran has the advantage over jSA2 that it has fewer parameters to tune, since the initial values of the scale and crossover factors are selected randomly.

This research question concluded that jSA2Ran is the most effective of the 13 algorithms. The following section investigates jSA2Ran on a large set of environments in order to determine whether jSA2Ran is actually better than CDE.

6.4.3 Research Question 2

How does the algorithm identified in the previous research question compare to DynDE and CDE on the set of environments used in Chapter 4?

The previous section found that jSA2Ran is the most effective self-adaptive algorithm. The purpose of this section is to comparatively evaluate the performance of jSA2Ran on a broad range of dynamic environments. The set of environments used in Chapter 4 to evaluate algorithms was used for this purpose. The performance of jSA2Ran was compared to that of DynDE and CDE.

The performance analysis of jSA2Ran compared to DynDE found that jSA2Ran performed statistically significantly better than DynDE in 1 479 of the 2 160 environments,

and performed worse in only 198 environments. jSA2Ran thus performed better than DynDE in more cases than CDE, and worse in fewer cases than CDE (refer to Section 4.6.5.3). The average percentage improvement of jSA2Ran over DynDE was found to be 14.78%. The complete performance analysis of jSA2Ran versus DynDE is given in Appendix D.

The performance analysis of jSA2Ran versus CDE is given in Tables 6.2 and 6.3. The majority of experiments did not result in statistically significantly different results. jSA2Ran did, however, perform better than CDE in 724 of the 2 160 experiments, and worse in 339 experiments.

The environments in which jSA2Ran performed better than CDE in 5 and 10 dimensions are mainly concentrated on the MPB functions. Notable exceptions are functions F_3 and F_6 from the GDBG which showed improved results achieved by jSA2Ran. The trend in higher dimensions is that jSA2Ran is better than CDE on the GDBG functions at the higher change periods (greater than 10 000 function evaluations). The function F_5 is an exception in high dimensions as jSA2Ran performs better on this function, even at low change periods. The results thus indicate that the difference in performance between jSA2Ran depends mostly on the change period and dimension, but that there is also a dependence on the underlying function.

The average percentage improvement of jSA2Ran over CDE over all experiments was found to be 5.01%. The APIs per dimension were found to be 1.97%, 3.70%, 7.48%, 8.48% and 3.43% for 5, 10, 25, 50 and 100 dimensions respectively. Larger improvements in offline error were thus found in higher dimensions, with the exception of 100 dimensions. The APIs per change period were found to be 0.36%, 3.75%, 3.99%, 2.54%, 3.28%, 5.31%, 9.86% and 11.01% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The percentage improvement thus increases as the change period increases.

This research question used a large experimental set to determine whether jSA2Ran is a better algorithm than DynDE and CDE. The performance analysis found that jSA2Ran performed better more often than DynDE and CDE, and that jSA2Ran results in a considerable percentage improvement over DynDE and CDE, on average. The conclusion that is drawn from this section is that jSA2Ran is a superior algorithm to DynDE and CDE.

Table 6.2: jSA2Ran vs CDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑7 ↓0**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
20	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑13 ↓0**
40	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑15 ↓0**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑12 ↓0**
C	(6)	**↑2 ↓0**	**↑3 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑25 ↓0**
S	(6)	**↑3 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑36 ↓0**
		GDBG								
F_{1a}	(6)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓5*	*↑0 ↓3*	*↑0 ↓2*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓12*
F_{1b}	(6)	↑1 ↓1	↑0 ↓0	*↑0 ↓4*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑1 ↓12*
F_2	(6)	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓3*	↑0 ↓0	↑0 ↓0	*↑0 ↓14*
F_3	(6)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑7 ↓0**
F_4	(6)	↑0 ↓0	↑1 ↓1	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑1 ↓14*
F_5	(6)	↑0 ↓0	**↑3 ↓1**	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓3*	*↑0 ↓1*	↑0 ↓0	*↑3 ↓20*
F_6	(6)	↑0 ↓0	**↑1 ↓0**	↑1 ↓1	**↑3 ↓1**	**↑4 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑13 ↓2**
T_1	(7)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓3*	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓16*
T_2	(7)	↑0 ↓0	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓3*	*↑1 ↓4*	*↑0 ↓3*	↑0 ↓0	*↑0 ↓1*	*↑3 ↓12*
T_3	(7)	↑0 ↓0	**↑2 ↓0**	*↑0 ↓2*	*↑2 ↓3*	*↑1 ↓3*	*↑1 ↓2*	**↑2 ↓0**	↑0 ↓0	*↑8 ↓10*
T_4	(7)	*↑0 ↓1*	*↑0 ↓1*	*↑1 ↓5*	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑1 ↓16*
T_5	(7)	↑0 ↓0	**↑1 ↓0**	*↑0 ↓4*	*↑1 ↓3*	*↑1 ↓2*	*↑1 ↓2*	*↑0 ↓2*	↑0 ↓0	*↑4 ↓13*
T_6	(7)	**↑1 ↓0**	**↑1 ↓0**	*↑1 ↓2*	↑1 ↓1	*↑2 ↓3*	**↑2 ↓1**	↑0 ↓0	**↑1 ↓0**	**↑9 ↓7**
All	(54)	**↑6 ↓1**	**↑13 ↓3**	*↑13 ↓20*	*↑14 ↓19*	*↑14 ↓19*	**↑11 ↓9**	**↑7 ↓2**	**↑8 ↓1**	**↑86 ↓74**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑5 ↓0**
5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑5 ↓0**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓0**
20	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑13 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
80	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
C	(6)	↑0 ↓0	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑21 ↓0**
S	(6)	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑38 ↓0**
		GDBG								
F_{1a}	(6)	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓1*	*↑0 ↓2*	↑0 ↓0	↑0 ↓0	*↑0 ↓13*
F_{1b}	(6)	↑0 ↓0	*↑1 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑1 ↓13*
F_2	(6)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓3*	*↑0 ↓1*	*↑0 ↓2*	↑0 ↓0	*↑0 ↓13*
F_3	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑3 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓0**
F_4	(6)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓3*	*↑0 ↓3*	↑0 ↓0	*↑0 ↓18*
F_5	(6)	**↑2 ↓0**	**↑3 ↓2**	↑3 ↓3	*↑1 ↓3*	*↑2 ↓3*	↑2 ↓2	*↑2 ↓3*	**↑2 ↓0**	**↑17 ↓16**
F_6	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑3 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑18 ↓2**
T_1	(7)	**↑1 ↓0**	*↑0 ↓5*	*↑0 ↓5*	*↑1 ↓6*	*↑1 ↓4*	*↑1 ↓2*	↑1 ↓1	**↑1 ↓0**	*↑6 ↓23*
T_2	(7)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑1 ↓3*	*↑1 ↓2*	*↑1 ↓2*	*↑0 ↓2*	↑0 ↓0	*↑3 ↓10*
T_3	(7)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	*↑1 ↓2*	*↑1 ↓2*	↑2 ↓2	↑1 ↓1	**↑2 ↓0**	**↑10 ↓7**
T_4	(7)	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓3*	↑0 ↓0	↑2 ↓2	↑0 ↓0	*↑2 ↓19*
T_5	(7)	↑0 ↓0	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑2 ↓1**	**↑2 ↓1**	↑1 ↓1	**↑2 ↓0**	**↑11 ↓5**
T_6	(7)	↑0 ↓0	**↑2 ↓0**	*↑1 ↓2*	*↑3 ↓4*	↑2 ↓2	↑2 ↓2	**↑2 ↓1**	**↑2 ↓0**	**↑14 ↓11**
All	(54)	**↑6 ↓1**	**↑15 ↓8**	*↑12 ↓14*	*↑16 ↓21*	↑14 ↓14	**↑16 ↓9**	**↑13 ↓8**	**↑13 ↓0**	**↑105 ↓75**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓0**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑10 ↓1**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
80	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
C	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	**↑21 ↓1**
S	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓0**
		GDBG								
F_{1a}	(6)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓5*	↑0 ↓0	**↑3 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑11 ↓9**
F_{1b}	(6)	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓1*	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	↑12 ↓12
F_2	(6)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓3*	*↑0 ↓1*	**↑2 ↓1**	**↑3 ↓0**	**↑3 ↓0**	*↑8 ↓9*
F_3	(6)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑4 ↓0**	**↑2 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑14 ↓1**
F_4	(6)	*↑0 ↓1*	**↑1 ↓0**	*↑1 ↓2*	*↑0 ↓3*	*↑0 ↓4*	**↑2 ↓1**	**↑3 ↓0**	**↑4 ↓0**	↑11 ↓11
F_5	(6)	*↑0 ↓1*	**↑3 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑33 ↓1**
F_6	(6)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑2 ↓2	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑21 ↓3**
T_1	(7)	↑0 ↓0	*↑1 ↓2*	*↑0 ↓5*	*↑1 ↓5*	**↑2 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑25 ↓12**
T_2	(7)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓3*	↑1 ↓1	**↑2 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑11 ↓4**
T_3	(7)	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**	**↑2 ↓1**	**↑2 ↓1**	**↑3 ↓0**	**↑4 ↓0**	**↑14 ↓4**
T_4	(7)	*↑0 ↓2*	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓5*	↑2 ↓2	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑20 ↓16**
T_5	(7)	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	↑2 ↓2	↑1 ↓1	**↑3 ↓1**	**↑5 ↓0**	**↑4 ↓0**	**↑17 ↓5**
T_6	(7)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**	*↑1 ↓3*	**↑3 ↓1**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑23 ↓5**
All	(54)	*↑0 ↓4*	**↑16 ↓5**	**↑17 ↓10**	*↑15 ↓19*	**↑19 ↓6**	**↑32 ↓2**	**↑38 ↓0**	**↑36 ↓1**	**↑173 ↓47**

Table 6.3: jSA2Ran vs CDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑11 ↓0**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑10 ↓0**
20	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓2**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓0**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑12 ↓0**
C	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑3 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑20 ↓2**
S	(6)	**↑1 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓0**
		GDBG								
F_{1a}	(6)	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓2*	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑17 ↓13**
F_{1b}	(6)	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓3*	*↑0 ↓6*	*↑0 ↓2*	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	*↑15 ↓18*
F_2	(6)	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓4*	**↑2 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑12 ↓10**
F_3	(6)	↑1 ↓1	*↑0 ↓2*	*↑0 ↓3*	↑0 ↓0	**↑1 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑17 ↓6**
F_4	(6)	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓3*	*↑0 ↓4*	**↑3 ↓0**	**↑5 ↓0**	**↑5 ↓0**	↑13 ↓13
F_5	(6)	**↑1 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
F_6	(6)	*↑0 ↓2*	↑0 ↓0	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑19 ↓5**
T_1	(7)	*↑0 ↓3*	*↑0 ↓3*	*↑1 ↓3*	*↑1 ↓5*	↑1 ↓1	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑24 ↓15**
T_2	(7)	*↑0 ↓1*	**↑1 ↓0**	*↑1 ↓2*	*↑1 ↓2*	*↑1 ↓4*	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑23 ↓9**
T_3	(7)	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑20 ↓4**
T_4	(7)	*↑1 ↓3*	*↑1 ↓4*	*↑1 ↓3*	*↑1 ↓4*	*↑1 ↓2*	**↑7 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑25 ↓16**
T_5	(7)	*↑0 ↓2*	↑1 ↓1	*↑1 ↓3*	↑2 ↓2	**↑3 ↓2**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑22 ↓10**
T_6	(7)	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓2*	*↑1 ↓3*	*↑1 ↓2*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑19 ↓11**
All	(54)	*↑3 ↓12*	**↑15 ↓11**	**↑17 ↓14**	*↑16 ↓17*	**↑15 ↓13**	**↑41 ↓0**	**↑43 ↓0**	**↑46 ↓0**	**↑196 ↓67**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑6 ↓1**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑1 ↓1	**↑7 ↓1**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑4 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑5 ↓0**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑4 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑6 ↓0**
C	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑17 ↓0**
S	(6)	*↑0 ↓1*	**↑3 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑15 ↓2**
		GDBG								
F_{1a}	(6)	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓4*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	*↑14 ↓16*
F_{1b}	(6)	**↑1 ↓0**	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	*↑15 ↓21*
F_2	(6)	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓4*	**↑2 ↓1**	**↑4 ↓0**	**↑5 ↓0**	*↑11 ↓18*
F_3	(6)	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑23 ↓1**
F_4	(6)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓4*	↑2 ↓2	**↑4 ↓0**	**↑5 ↓0**	*↑11 ↓14*
F_5	(6)	**↑1 ↓0**	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓0**
F_6	(6)	**↑2 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	**↑2 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑4 ↓0**	**↑19 ↓4**
T_1	(7)	**↑1 ↓0**	*↑1 ↓2*	*↑1 ↓2*	*↑1 ↓4*	↑2 ↓2	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑27 ↓10**
T_2	(7)	**↑1 ↓0**	*↑0 ↓2*	*↑1 ↓4*	*↑1 ↓2*	*↑2 ↓4*	**↑3 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑21 ↓12**
T_3	(7)	**↑2 ↓1**	↑1 ↓1	↑1 ↓1	*↑1 ↓4*	**↑3 ↓1**	**↑3 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑21 ↓8**
T_4	(7)	**↑1 ↓0**	*↑1 ↓3*	*↑1 ↓4*	*↑1 ↓3*	*↑2 ↓4*	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑27 ↓14**
T_5	(7)	↑0 ↓0	*↑0 ↓2*	*↑1 ↓3*	*↑1 ↓4*	*↑3 ↓4*	**↑3 ↓2**	**↑4 ↓1**	**↑5 ↓0**	**↑17 ↓16**
T_6	(7)	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓3*	*↑1 ↓4*	*↑1 ↓2*	**↑4 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑19 ↓14**
All	(54)	**↑5 ↓3**	*↑10 ↓13*	*↑15 ↓17*	*↑14 ↓21*	*↑16 ↓17*	**↑28 ↓3**	**↑37 ↓1**	**↑39 ↓1**	**↑164 ↓76**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	*↑0 ↓1*	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑33 ↓1**
5	(10)	**↑1 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑3 ↓0**	**↑2 ↓0**	**↑3 ↓0**	**↑4 ↓2**	**↑38 ↓2**
10	(10)	**↑2 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑44 ↓0**
20	(10)	**↑4 ↓1**	**↑9 ↓0**	**↑9 ↓0**	**↑7 ↓0**	**↑7 ↓1**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑50 ↓2**
40	(10)	**↑2 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑7 ↓0**	**↑4 ↓0**	**↑56 ↓0**
80	(10)	**↑1 ↓0**	**↑5 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑57 ↓0**
C	(30)	**↑2 ↓1**	**↑22 ↓0**	**↑28 ↓0**	**↑19 ↓0**	**↑11 ↓1**	**↑10 ↓0**	**↑7 ↓0**	**↑5 ↓1**	**↑104 ↓3**
S	(30)	**↑8 ↓1**	**↑25 ↓0**	**↑26 ↓0**	**↑26 ↓0**	**↑23 ↓0**	**↑22 ↓0**	**↑21 ↓0**	**↑23 ↓1**	**↑174 ↓2**
		GDBG								
F_{1a}	(30)	*↑0 ↓4*	*↑0 ↓8*	*↑0 ↓16*	*↑0 ↓23*	*↑0 ↓9*	**↑12 ↓3**	**↑14 ↓0**	**↑16 ↓0**	*↑42 ↓63*
F_{1b}	(30)	*↑2 ↓5*	*↑1 ↓13*	*↑0 ↓18*	*↑0 ↓25*	*↑0 ↓11*	**↑12 ↓2**	**↑14 ↓1**	**↑15 ↓1**	*↑44 ↓76*
F_2	(30)	*↑0 ↓2*	*↑0 ↓8*	*↑0 ↓12*	*↑0 ↓17*	*↑0 ↓17*	↑6 ↓6	**↑12 ↓2**	**↑13 ↓0**	*↑31 ↓64*
F_3	(30)	↑2 ↓2	*↑1 ↓2*	*↑1 ↓4*	**↑5 ↓0**	**↑12 ↓0**	**↑17 ↓0**	**↑17 ↓0**	**↑16 ↓0**	**↑71 ↓8**
F_4	(30)	*↑0 ↓3*	*↑2 ↓6*	*↑1 ↓11*	*↑0 ↓18*	*↑0 ↓22*	↑7 ↓7	**↑12 ↓3**	**↑14 ↓0**	*↑36 ↓70*
F_5	(30)	**↑4 ↓1**	**↑16 ↓3**	**↑17 ↓8**	**↑16 ↓8**	**↑19 ↓8**	**↑20 ↓5**	**↑20 ↓4**	**↑20 ↓0**	**↑132 ↓37**
F_6	(30)	↑2 ↓2	**↑2 ↓0**	*↑1 ↓6*	**↑9 ↓6**	**↑13 ↓1**	**↑22 ↓0**	**↑21 ↓1**	**↑20 ↓0**	**↑90 ↓16**
T_1	(35)	*↑2 ↓3*	*↑2 ↓14*	*↑2 ↓21*	*↑4 ↓25*	*↑6 ↓10*	**↑22 ↓2**	**↑22 ↓1**	**↑22 ↓0**	**↑82 ↓76**
T_2	(35)	↑1 ↓1	**↑4 ↓2**	*↑3 ↓8*	*↑3 ↓13*	*↑6 ↓15*	**↑11 ↓5**	**↑17 ↓2**	**↑16 ↓1**	**↑61 ↓47**
T_3	(35)	**↑4 ↓3**	**↑5 ↓1**	↑4 ↓4	*↑7 ↓11*	↑8 ↓8	**↑12 ↓5**	**↑15 ↓1**	**↑18 ↓0**	**↑73 ↓33**
T_4	(35)	*↑2 ↓7*	*↑2 ↓14*	*↑3 ↓21*	*↑2 ↓21*	*↑5 ↓15*	**↑20 ↓1**	**↑21 ↓2**	**↑20 ↓0**	*↑75 ↓81*
T_5	(35)	*↑0 ↓3*	**↑5 ↓3**	*↑4 ↓11*	*↑7 ↓12*	↑10 ↓10	**↑14 ↓6**	**↑15 ↓4**	**↑16 ↓0**	**↑71 ↓49**
T_6	(35)	*↑1 ↓2*	*↑4 ↓6*	*↑4 ↓10*	*↑7 ↓15*	*↑9 ↓10*	**↑17 ↓4**	**↑20 ↓1**	**↑22 ↓0**	**↑84 ↓48**
All	(270)	*↑20 ↓21*	**↑69 ↓40**	*↑74 ↓75*	*↑75 ↓97*	**↑78 ↓69**	**↑128 ↓23**	**↑138 ↓11**	**↑142 ↓3**	**↑724 ↓339**

6.4.4 Research Question 3

Which one of the four algorithms for self-adapting the Brownian radius is the most effective?

This section investigates the performance of each of the four self-adaptive Brownian radius algorithms in comparison with CDE. The purpose of this comparison is to determine which of the four algorithms yields the best results.

Table 6.4 lists the results of the performance analysis comparing each of the four algorithms to CDE. The table gives the number of times that each of the algorithms performed statistically significantly better than CDE (**Nr Better**), the number of times that each algorithm performed significantly worse than CDE (**Nr Worse**), the difference between the number of times that each algorithm performed better and worse (**Difference**), and the average percentage improvement (**API**) over CDE.

The analysis of the experimental results shows that SABrNor and SABrCau, the two algorithms that did not reset the $r_{dev}(t)$ value after changes in the environment, performed worse than CDE more often than they performed better. Furthermore, the average percentage improvement values for these algorithms were found to be negative. These two approaches thus degrade the performance of CDE.

Algorithms SABrNorRes and SABrCauRes both yielded positive API values, indicating that, on average, their performances were superior to those of CDE. The number of times that these algorithms performed better than CDE was also more than the number of times that they performed worse. SABrNorRes, which used a normal distribution to calculate $r_{dev}(t)$, performed better than CDE slightly more often than SABrCauRes, which used a Cauchy distribution. The average percentage improvement of SABrNorRes over CDE was also slightly higher than the average percentage improvement of SABrCauRes over CDE.

Table 6.4: Performance of SA Brownian radius algorithms vs CDE

Algorithm	**Nr Better**	**Nr Worse**	**Difference**	**API**
SABrNor	241	283	-42	-4.41 %
SABrCau	241	269	-28	-4.20 %
SABrNorRes	419	125	294	18.44 %
SABrCauRes	413	126	287	18.23 %

This section described a comparative evaluation of the four approaches to self-adapting

the Brownian radius. The experimental results indicate that SABrNorRes, the algorithm that utilises a normal distribution and which resets $r_{dev}(t)$ values after changes in the environment, is the most effective of the four algorithms.

6.4.5 Research Question 4

How does the algorithm identified in the previous research question compare to DynDE and CDE on the set of environments used in Chapter 4?

SABrNorRes, the self-adaptive Brownian radius algorithm which employs a normal distribution and resets the $r_{dev}(t)$ value after a change in the environment, was identified under the previous research question as the most effective of the algorithms that were investigated. The purpose of this research question is to perform a comparative performance analysis of SABrNorRes, with respect to DynDE and CDE, on a wide range of experimental environments.

The analysis of the experimental results found that SABrNorRes performed statistically significantly better than DynDE in 1 532 of the 2 160 environments, and worse in 325. SABrNorRes thus performed better than DynDE more often than CDE performed better than DynDE. The number of times that SABrNorRes performed worse than DynDE is, however, higher than the number of times that CDE performed worse than DynDE. The average percentage improvement of SABrNorRes over DynDE was found to be 18.19%. The complete comparative performance analysis of SABrNorRes compared to DynDE is given in Appendix D.

The performance analysis of SABrNorRes compared to CDE is given in Tables 6.5 and 6.6. The shaded, italicised and boldfaced text have their usual meaning. SABrNorRes performed statistically significantly better than CDE on 1 076 environments and worse in 686 environments. SABrNorRes performed better more often than CDE on change periods of more than 5 000 function evaluations. CDE performed better more often than SABrNorRes in 100 dimensions.

The analysis indicates that the comparative performances of CDE and SABrNorRes was dependent on the underlying function. SABrNorRes was especially effective on the MPB functions, with the exception of experiments in 100 dimensions. The GDBG function F_5 proved particularly challenging to SABrNorRes in dimensions below 50, where CDE

performed better more often. Conversely, F_5 was one of only two GDBG functions in which SABrNorRes performed better than CDE more often in 50 and 100 dimensions.

SABrNorRes performed better than CDE more often than CDE performed better than SABrNorRes. However, the CDE did perform better than SABrNorRes in almost a third of all experimental environments. SABrNorRes can therefor not definitively be described as better than CDE without considering the magnitude of the differences between the two algorithms.

The average percentage improvement (API), calculated as in equation (4.6) on page 148, of SABrNorRes over CDE was calculated to determine how much better, on average, SABrNorRes is than CDE. The average percentage improvement of SABrNorRes over CDE over all experiments was found to be 9.5%. The APIs per dimension were found to be 19.96%, 13.57%, 9.17%, 10.03% and -5.23% for 5, 10, 25, 50 and 100 dimensions respectively. SABrNorRes thus consistently yielded substantial improvements with the exception of 100 dimensional environments. The APIs per change period were found to be 0.17%, 8.43%, 7.05%, 2.89%, 4.42%, 10.79%, 17.64% and 24.60% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations, respectively. SABrNorRes, on average, thus performed very similar to CDE at a change period of 100 function evaluations. The magnitude of the improvement over CDE increased to a considerable percentage as the change period was increased.

This research question investigated the comparative performance of SABRNorRes and its predecessor algorithms. SABrNorRes performed better more often than both DynDE and CDE on a large set of dynamic environments. This, and the fact that sizable percentage improvements of SABrNorRes over CDE were found (and consequently DynDE), leads to the conclusion that SABrNorRes is a more effective algorithm for solving DOPs than DynDE and CDE.

6.4.6 Research Question 5

How does the combination of the algorithms for adapting the scale and crossover factors and the Brownian radius, SACDE, compare to DynDE and CDE?

SABRNorRes and jSA2Ran, the two self-adaptive approaches that were investigated in research questions 2 and 4 can be combined into a single CDE-based algorithm. This

Table 6.5: SABrNorRes vs CDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑**1** ↓**0**	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**8** ↓**2**
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**11** ↓**0**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**14** ↓**0**
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
C	(6)	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**4** ↓**0**	↑**4** ↓**1**	↑**3** ↓**0**	↑**3** ↓**0**	↑**34** ↓**2**
S	(6)	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**47** ↓**0**
		GDBG								
F_{1a}	(6)	↑1 ↓1	↑*0* ↓*3*	↑*0* ↓*5*	↑2 ↓2	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**26** ↓**11**
F_{1b}	(6)	↑*2* ↓*3*	↑*0* ↓*4*	↑*0* ↓*6*	↑**3** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**28** ↓**13**
F_2	(6)	↑0 ↓0	↑*0* ↓*3*	↑*0* ↓*6*	↑*0* ↓*5*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**22** ↓**14**
F_3	(6)	↑1 ↓1	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**1**
F_4	(6)	↑*0* ↓*2*	↑1 ↓1	↑*0* ↓*4*	↑*0* ↓*4*	↑*1* ↓*3*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**18** ↓**14**
F_5	(6)	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑3 ↓3	↑**3** ↓**2**	↑**3** ↓**2**	↑*9* ↓*35*
F_6	(6)	↑**3** ↓**1**	↑**3** ↓**1**	↑2 ↓2	↑**4** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**35** ↓**5**
T_1	(7)	↑*0* ↓*3*	↑*1* ↓*6*	↑*1* ↓*6*	↑*2* ↓*5*	↑**4** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**29** ↓**21**
T_2	(7)	↑**3** ↓**1**	↑2 ↓2	↑*1* ↓*6*	↑*2* ↓*4*	↑**4** ↓**2**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**33** ↓**15**
T_3	(7)	↑1 ↓1	↑**3** ↓**1**	↑*2* ↓*3*	↑*2* ↓*3*	↑2 ↓2	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**27** ↓**13**
T_4	(7)	↑*0* ↓*6*	↑*1* ↓*4*	↑*1* ↓*5*	↑**3** ↓**2**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**32** ↓**17**
T_5	(7)	↑*0* ↓*2*	↑*0* ↓*2*	↑*1* ↓*5*	↑*2* ↓*3*	↑**5** ↓**2**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**25** ↓**17**
T_6	(7)	↑**3** ↓**0**	↑*2* ↓*3*	↑*2* ↓*4*	↑**4** ↓**1**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**34** ↓**10**
All	(54)	↑**18** ↓**13**	↑**20** ↓**18**	↑*19* ↓*29*	↑**25** ↓**19**	↑**36** ↓**8**	↑**47** ↓**4**	↑**48** ↓**2**	↑**48** ↓**2**	↑**261** ↓**95**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**12** ↓**0**
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**11** ↓**0**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**12** ↓**0**
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**14** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
C	(6)	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**3** ↓**0**	↑**3** ↓**0**	↑**2** ↓**0**	↑**3** ↓**0**	↑**32** ↓**0**
S	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**48** ↓**0**
		GDBG								
F_{1a}	(6)	↑1 ↓1	↑2 ↓2	↑*1* ↓*2*	↑**3** ↓**2**	↑**6** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**30** ↓**7**
F_{1b}	(6)	↑1 ↓1	↑*1* ↓*2*	↑*1* ↓*3*	↑**2** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**28** ↓**6**
F_2	(6)	↑*0* ↓*3*	↑2 ↓2	↑*0* ↓*3*	↑*2* ↓*3*	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**26** ↓**12**
F_3	(6)	↑*0* ↓*2*	↑**6** ↓**0**	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**39** ↓**2**
F_4	(6)	↑*0* ↓*2*	↑*1* ↓*2*	↑*0* ↓*3*	↑*1* ↓*3*	↑**3** ↓**2**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**23** ↓**12**
F_5	(6)	↑*2* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*4*	↑*2* ↓*43*
F_6	(6)	↑0 ↓0	↑**3** ↓**1**	↑**3** ↓**2**	↑**4** ↓**1**	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**32** ↓**5**
T_1	(7)	↑*0* ↓*2*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*5*	↑*3* ↓*4*	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**0**	↑*24* ↓*25*
T_2	(7)	↑0 ↓0	↑**2** ↓**1**	↑*1* ↓*3*	↑*2* ↓*4*	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**0**	↑**28** ↓**11**
T_3	(7)	↑1 ↓1	↑**3** ↓**1**	↑1 ↓1	↑**4** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**31** ↓**8**
T_4	(7)	↑*0* ↓*6*	↑*1* ↓*5*	↑*1* ↓*6*	↑*2* ↓*3*	↑**3** ↓**2**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑25 ↓25
T_5	(7)	↑*0* ↓*3*	↑**3** ↓**1**	↑**3** ↓**1**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**35** ↓**10**
T_6	(7)	↑**3** ↓**0**	↑**5** ↓**1**	↑*1* ↓*2*	↑**4** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**37** ↓**8**
All	(54)	↑**15** ↓**12**	↑**27** ↓**15**	↑**20** ↓**19**	↑**28** ↓**15**	↑**37** ↓**10**	↑**44** ↓**6**	↑**44** ↓**6**	↑**45** ↓**4**	↑**260** ↓**87**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**12** ↓**0**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**10** ↓**2**
10	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**10** ↓**3**
20	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**10** ↓**2**
40	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**12** ↓**0**
80	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
C	(6)	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**3** ↓**1**	↑*2* ↓*3*	↑*1* ↓*3*	↑**2** ↓**0**	↑**1** ↓**0**	↑**21** ↓**8**
S	(6)	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**47** ↓**1**
		GDBG								
F_{1a}	(6)	↑*0* ↓*4*	↑*0* ↓*3*	↑*0* ↓*3*	↑*1* ↓*2*	↑**3** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**20** ↓**12**
F_{1b}	(6)	↑*0* ↓*5*	↑*0* ↓*3*	↑*1* ↓*3*	↑*1* ↓*2*	↑**2** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**21** ↓**13**
F_2	(6)	↑*0* ↓*6*	↑*0* ↓*2*	↑*0* ↓*3*	↑*0* ↓*6*	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑18 ↓18
F_3	(6)	↑*0* ↓*6*	↑**5** ↓**0**	↑*1* ↓*3*	↑**3** ↓**1**	↑**4** ↓**0**	↑**4** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**26** ↓**10**
F_4	(6)	↑*0* ↓*6*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*4*	↑*0* ↓*2*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**18** ↓**16**
F_5	(6)	↑**6** ↓**0**	↑**4** ↓**2**	↑3 ↓3	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*13* ↓*35*
F_6	(6)	↑*0* ↓*4*	↑**2** ↓**0**	↑**3** ↓**2**	↑**3** ↓**1**	↑**3** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**26** ↓**10**
T_1	(7)	↑*1* ↓*3*	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑1 ↓1	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑*23* ↓*24*
T_2	(7)	↑*1* ↓*6*	↑**3** ↓**0**	↑*1* ↓*5*	↑*0* ↓*5*	↑*0* ↓*2*	↑**5** ↓**2**	↑**5** ↓**2**	↑**5** ↓**2**	↑*20* ↓*24*
T_3	(7)	↑*1* ↓*5*	↑**3** ↓**1**	↑**2** ↓**1**	↑*1* ↓*4*	↑1 ↓1	↑**3** ↓**1**	↑**4** ↓**1**	↑**6** ↓**1**	↑**21** ↓**15**
T_4	(7)	↑*1* ↓*5*	↑*1* ↓*4*	↑*0* ↓*5*	↑*1* ↓*3*	↑3 ↓3	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**24** ↓**23**
T_5	(7)	↑*1* ↓*6*	↑2 ↓2	↑**2** ↓**1**	↑**4** ↓**2**	↑**4** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**30** ↓**15**
T_6	(7)	↑*1* ↓*6*	↑**1** ↓**0**	↑**2** ↓**1**	↑*1* ↓*2*	↑**3** ↓**1**	↑**5** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**24** ↓**13**
All	(54)	↑*11* ↓*33*	↑**23** ↓**12**	↑**20** ↓**19**	↑*17* ↓*23*	↑**20** ↓**12**	↑**38** ↓**10**	↑**41** ↓**7**	↑**40** ↓**7**	↑**210** ↓**123**

Table 6.6: SABrNorRes vs CDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
5	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓3**
10	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑9 ↓5**
20	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑10 ↓4**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑10 ↓2**
80	(2)	*↑0 ↓2*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑12 ↓3**
C	(6)	*↑0 ↓5*	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓1**	*↑1 ↓3*	*↑0 ↓4*	*↑1 ↓4*	**↑2 ↓0**	**↑19 ↓17**
S	(6)	**↑2 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑44 ↓1**
		GDBG								
F_{1a}	(6)	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	**↑2 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑19 ↓18**
F_{1b}	(6)	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	↑2 ↓2	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑20 ↓24*
F_2	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓6*	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	*↑15 ↓26*
F_3	(6)	*↑0 ↓6*	**↑3 ↓0**	*↑0 ↓3*	*↑1 ↓4*	*↑1 ↓3*	**↑1 ↓0**	↑2 ↓2	*↑1 ↓2*	*↑9 ↓20*
F_4	(6)	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑15 ↓25*
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑3 ↓3	*↑1 ↓3*	*↑1 ↓4*	*↑1 ↓5*	*↑1 ↓5*	**↑25 ↓20**
F_6	(6)	*↑0 ↓6*	**↑2 ↓0**	↑1 ↓1	**↑4 ↓1**	**↑4 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑4 ↓1**	**↑25 ↓11**
T_1	(7)	*↑1 ↓5*	*↑3 ↓4*	*↑1 ↓4*	*↑1 ↓6*	*↑1 ↓4*	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	*↑25 ↓26*
T_2	(7)	*↑1 ↓5*	*↑2 ↓4*	*↑2 ↓3*	*↑1 ↓6*	*↑0 ↓6*	**↑4 ↓2**	**↑4 ↓2**	**↑4 ↓2**	*↑18 ↓30*
T_3	(7)	*↑1 ↓5*	*↑1 ↓2*	*↑1 ↓2*	*↑2 ↓3*	*↑1 ↓3*	**↑2 ↓1**	**↑5 ↓2**	**↑4 ↓2**	*↑17 ↓20*
T_4	(7)	*↑1 ↓6*	*↑2 ↓4*	*↑1 ↓5*	*↑1 ↓6*	↑3 ↓3	**↑5 ↓1**	**↑5 ↓2**	**↑5 ↓1**	*↑23 ↓28*
T_5	(7)	*↑1 ↓6*	*↑2 ↓3*	*↑2 ↓3*	*↑1 ↓2*	**↑3 ↓2**	**↑3 ↓0**	**↑5 ↓1**	**↑3 ↓1**	**↑20 ↓18**
T_6	(7)	*↑1 ↓6*	*↑1 ↓4*	*↑1 ↓4*	*↑2 ↓5*	↑2 ↓2	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑25 ↓22**
All	(54)	*↑8 ↓39*	**↑23 ↓21**	*↑20 ↓21*	*↑17 ↓29*	*↑17 ↓23*	**↑32 ↓9**	**↑38 ↓12**	**↑36 ↓8**	**↑191 ↓162**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	*↑4 ↓9*
5	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑2 ↓12*
10	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓14*
20	(2)	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓13*
40	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓13*
80	(2)	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑4 ↓10*
C	(6)	*↑0 ↓5*	**↑6 ↓0**	**↑6 ↓0**	*↑1 ↓2*	*↑2 ↓4*	*↑0 ↓4*	*↑0 ↓5*	*↑1 ↓4*	*↑16 ↓24*
S	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓47*
		GDBG								
F_{1a}	(6)	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	*↑16 ↓27*
F_{1b}	(6)	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑17 ↓28*
F_2	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓4*	*↑0 ↓4*	*↑2 ↓3*	**↑5 ↓0**	**↑5 ↓0**	*↑13 ↓29*
F_3	(6)	*↑0 ↓6*	**↑3 ↓0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓3*	*↑1 ↓3*	**↑3 ↓0**	**↑4 ↓0**	*↑11 ↓23*
F_4	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑1 ↓4*	*↑0 ↓4*	*↑2 ↓3*	**↑4 ↓0**	**↑5 ↓0**	*↑12 ↓28*
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓2**	**↑43 ↓3**
F_6	(6)	*↑0 ↓6*	**↑3 ↓2**	**↑3 ↓2**	**↑2 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑26 ↓10**
T_1	(7)	*↑1 ↓5*	*↑3 ↓4*	*↑2 ↓5*	*↑1 ↓5*	*↑1 ↓4*	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑28 ↓23**
T_2	(7)	*↑1 ↓6*	*↑2 ↓4*	*↑1 ↓5*	*↑2 ↓4*	*↑1 ↓5*	**↑3 ↓1**	**↑5 ↓1**	**↑5 ↓1**	*↑20 ↓27*
T_3	(7)	*↑1 ↓5*	*↑1 ↓4*	*↑1 ↓6*	*↑1 ↓3*	*↑1 ↓2*	*↑2 ↓3*	**↑3 ↓0**	**↑6 ↓0**	*↑16 ↓23*
T_4	(7)	*↑1 ↓6*	*↑2 ↓4*	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	**↑6 ↓1**	**↑6 ↓0**	**↑5 ↓1**	*↑26 ↓27*
T_5	(7)	*↑1 ↓6*	*↑3 ↓4*	*↑2 ↓3*	↑3 ↓3	↑2 ↓2	**↑5 ↓2**	**↑5 ↓0**	**↑5 ↓0**	**↑26 ↓20**
T_6	(7)	*↑1 ↓6*	*↑1 ↓5*	*↑1 ↓6*	*↑1 ↓5*	*↑2 ↓4*	**↑4 ↓2**	**↑6 ↓0**	**↑6 ↓0**	*↑22 ↓28*
All	(54)	*↑6 ↓45*	*↑18 ↓31*	*↑15 ↓35*	*↑11 ↓33*	*↑11 ↓32*	**↑26 ↓19**	**↑32 ↓12**	**↑35 ↓12**	*↑154 ↓219*
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑5 ↓3**	**↑8 ↓1**	**↑8 ↓1**	**↑6 ↓2**	**↑5 ↓1**	**↑4 ↓2**	**↑6 ↓1**	**↑7 ↓1**	**↑49 ↓12**
5	(10)	**↑5 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑4 ↓1**	↑4 ↓4	↑4 ↓4	**↑4 ↓2**	**↑4 ↓1**	**↑43 ↓17**
10	(10)	**↑5 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑6 ↓4**	**↑5 ↓4**	**↑5 ↓4**	**↑4 ↓3**	**↑4 ↓2**	**↑47 ↓22**
20	(10)	**↑6 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑6 ↓2**	**↑6 ↓4**	**↑6 ↓4**	**↑5 ↓3**	**↑5 ↓2**	**↑52 ↓19**
40	(10)	**↑5 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑7 ↓2**	**↑6 ↓3**	**↑6 ↓3**	**↑6 ↓2**	**↑56 ↓15**
80	(10)	*↑3 ↓6*	**↑9 ↓1**	**↑9 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑7 ↓3**	**↑8 ↓2**	**↑61 ↓15**
C	(30)	↑11 ↓11	**↑29 ↓0**	**↑29 ↓0**	**↑15 ↓5**	**↑12 ↓10**	*↑8 ↓12*	*↑8 ↓9*	**↑10 ↓4**	**↑122 ↓51**
S	(30)	**↑18 ↓8**	**↑24 ↓6**	**↑24 ↓5**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑186 ↓49**
		GDBG								
F_{1a}	(30)	*↑2 ↓14*	*↑2 ↓18*	*↑1 ↓21*	*↑6 ↓15*	**↑16 ↓7**	**↑26 ↓0**	**↑28 ↓0**	**↑30 ↓0**	**↑111 ↓75**
F_{1b}	(30)	*↑3 ↓20*	*↑1 ↓20*	*↑2 ↓23*	*↑6 ↓14*	**↑14 ↓7**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑114 ↓84**
F_2	(30)	*↑0 ↓21*	*↑2 ↓19*	*↑1 ↓21*	*↑3 ↓23*	*↑8 ↓12*	**↑24 ↓3**	**↑28 ↓0**	**↑28 ↓0**	*↑94 ↓99*
F_3	(30)	*↑1 ↓21*	**↑22 ↓0**	*↑10 ↓11*	**↑16 ↓11**	**↑17 ↓6**	**↑18 ↓3**	**↑22 ↓2**	**↑21 ↓2**	**↑127 ↓56**
F_4	(30)	*↑0 ↓22*	*↑2 ↓16*	*↑0 ↓18*	*↑2 ↓20*	*↑4 ↓16*	**↑22 ↓3**	**↑28 ↓0**	**↑28 ↓0**	*↑86 ↓95*
F_5	(30)	**↑20 ↓8**	**↑16 ↓14**	↑15 ↓15	*↑9 ↓21*	*↑6 ↓20*	*↑9 ↓19*	*↑9 ↓20*	*↑8 ↓19*	*↑92 ↓136*
F_6	(30)	*↑3 ↓17*	**↑13 ↓4**	**↑12 ↓9**	**↑17 ↓4**	**↑20 ↓1**	**↑28 ↓2**	**↑26 ↓2**	**↑25 ↓2**	**↑144 ↓41**
T_1	(35)	*↑3 ↓18*	*↑9 ↓25*	*↑6 ↓27*	*↑6 ↓27*	*↑10 ↓14*	**↑31 ↓3**	**↑32 ↓3**	**↑32 ↓2**	**↑129 ↓119**
T_2	(35)	*↑6 ↓18*	↑11 ↓11	*↑6 ↓22*	*↑7 ↓23*	*↑10 ↓16*	**↑25 ↓6**	**↑27 ↓6**	**↑27 ↓5**	**↑119 ↓107**
T_3	(35)	*↑5 ↓17*	**↑11 ↓9**	*↑7 ↓13*	*↑10 ↓14*	**↑10 ↓9**	**↑17 ↓7**	**↑24 ↓5**	**↑28 ↓5**	**↑112 ↓79**
T_4	(35)	*↑3 ↓29*	*↑7 ↓21*	*↑5 ↓26*	*↑9 ↓19*	**↑17 ↓13**	**↑30 ↓4**	**↑30 ↓4**	**↑29 ↓4**	**↑130 ↓120**
T_5	(35)	*↑3 ↓23*	*↑10 ↓12*	*↑10 ↓13*	**↑15 ↓11**	**↑20 ↓8**	**↑25 ↓5**	**↑28 ↓4**	**↑25 ↓4**	**↑136 ↓80**
T_6	(35)	*↑9 ↓18*	*↑10 ↓13*	*↑7 ↓17*	*↑12 ↓14*	**↑18 ↓9**	**↑27 ↓5**	**↑30 ↓2**	**↑29 ↓3**	**↑142 ↓81**
All	(270)	*↑58 ↓142*	**↑111 ↓97**	*↑94 ↓123*	*↑98 ↓119*	**↑121 ↓85**	**↑187 ↓48**	**↑203 ↓39**	**↑204 ↓33**	**↑1076 ↓686**

algorithm is referred to as self-adaptive competing differential evolution, SACDE. SACDE thus self-adapts both the scale and crossover factors and the Brownian radius. The focus of this research question is on the performance of SACDE compared to that of DynDE and CDE.

The performance analysis that compared the results of SACDE to that of DynDE found that SACDE performed statistically significantly better than DynDE in 1 485 environments (the analysis is given in Appendix D). This number is slightly larger than what was found when comparing jSA2Ran to DynDE, and slightly smaller than that found when comparing SABRNorRes to DynDE. SACDE performed statistically significantly worse than DynDE in 360 environments. This value is greater than that found when comparing DynDE to jSA2Ran and SABRNorRes. An average percentage improvement of SACDE over DynDE, of 16.76% was found.

Tables 6.7 and 6.8 contain the performance analysis of SACDE compared to CDE. SACDE performed statistically significantly better than CDE in 1 013 of the 2 160 environments, but performed worse in 814 of the environments. Similar trends to what was evident when comparing SABRNorRes to CDE can be observed in Tables 6.7 and 6.8. SACDE outperformed CDE more often in low dimensional and high change period environments. The comparative performance also depends on the underlying function, for example, consider the performance of SACDE on function F_5 over the various dimensions.

The average percentage improvement of SACDE over CDE over all experiments was found to be 7.8%. The APIs per dimension were found to be 17.94%, 11.84%, 7.26%, 8.24% and -6.29% for 5, 10, 25, 50 and 100 dimensions respectively. SACDE thus consistently yielded substantial improvements with the exception of 100 dimensional environments. The APIs per change period were found to be -0.13%, 7.47%, 5.37%, -0.13%, 1.19%, 8.66%, 15.98% and 23.98% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations, respectively. SACDE performed very similar to CDE at a change periods of 100 and 5 000 function evaluations. The magnitude of the improvement over CDE increased as the change period was increased to larger values.

A comparison of the APIs of SACDE to those found for jSA2Ran in Section 6.4.3 shows that the magnitude of improvements by SACDE was greater than the improvements achieved by jSA2Ran in the majority of cases. The APIs of SACDE were slightly lower

Table 6.7: SACDE vs CDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**8** ↓**4**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑**7** ↓**3**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**13** ↓**0**
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
C	(6)	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**4** ↓**2**	↑**3** ↓**2**	↑**3** ↓**2**	↑**3** ↓**0**	↑**32** ↓**7**
S	(6)	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**43** ↓**0**
		GDBG								
F_{1a}	(6)	*↑1 ↓2*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓3*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**23** ↓**15**
F_{1b}	(6)	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓6*	↑2 ↓2	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**26** ↓**16**
F_2	(6)	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓6*	↑**3** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**21** ↓**16**
F_3	(6)	*↑0 ↓1*	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**40** ↓**1**
F_4	(6)	*↑0 ↓3*	↑**3** ↓**2**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓3*	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	*↑19 ↓20*
F_5	(6)	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑2 ↓3*	↑3 ↓3	↑3 ↓3	*↑8 ↓37*
F_6	(6)	↑**3** ↓**1**	↑**3** ↓**1**	*↑2 ↓3*	↑**3** ↓**1**	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**34** ↓**7**
T_1	(7)	*↑0 ↓4*	*↑1 ↓6*	*↑1 ↓6*	*↑2 ↓5*	↑**4** ↓**2**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**29** ↓**23**
T_2	(7)	↑1 ↓1	↑3 ↓3	*↑1 ↓6*	*↑1 ↓5*	↑3 ↓3	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**29** ↓**18**
T_3	(7)	↑1 ↓1	↑**3** ↓**1**	*↑2 ↓5*	*↑2 ↓3*	↑2 ↓2	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**27** ↓**15**
T_4	(7)	*↑0 ↓7*	*↑1 ↓5*	*↑1 ↓6*	*↑2 ↓3*	↑**5** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**30** ↓**22**
T_5	(7)	*↑0 ↓2*	*↑0 ↓3*	*↑0 ↓4*	*↑2 ↓5*	↑**5** ↓**2**	↑**5** ↓**2**	↑**6** ↓**1**	↑**6** ↓**1**	↑**24** ↓**20**
T_6	(7)	↑**3** ↓**0**	↑3 ↓3	*↑2 ↓4*	*↑2 ↓3*	↑**4** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**32** ↓**14**
All	(54)	*↑14 ↓15*	↑**22** ↓**21**	*↑18 ↓31*	*↑21 ↓25*	↑**33** ↓**13**	↑**44** ↓**6**	↑**47** ↓**5**	↑**47** ↓**3**	↑**246** ↓**119**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**11** ↓**0**
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**11** ↓**2**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**11** ↓**2**
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**13** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
C	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**3** ↓**0**	↑2 ↓2	↑**3** ↓**2**	↑**2** ↓**0**	↑**3** ↓**0**	↑**30** ↓**4**
S	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**48** ↓**0**
		GDBG								
F_{1a}	(6)	*↑0 ↓1*	*↑1 ↓2*	*↑0 ↓5*	*↑0 ↓3*	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**22** ↓**11**
F_{1b}	(6)	*↑0 ↓1*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓5*	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**21** ↓**16**
F_2	(6)	*↑0 ↓5*	*↑0 ↓2*	*↑0 ↓3*	*↑1 ↓5*	↑2 ↓2	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**21** ↓**17**
F_3	(6)	*↑0 ↓4*	↑**5** ↓**0**	↑**2** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**37** ↓**4**
F_4	(6)	*↑0 ↓5*	*↑1 ↓2*	*↑0 ↓4*	*↑0 ↓4*	↑2 ↓2	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**19** ↓**17**
F_5	(6)	*↑2 ↓3*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓4*	*↑3 ↓43*
F_6	(6)	↑1 ↓1	↑**3** ↓**2**	↑**3** ↓**2**	↑**3** ↓**2**	↑**4** ↓**2**	↑**5** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**30** ↓**9**
T_1	(7)	*↑0 ↓2*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓4*	↑**5** ↓**1**	↑**6** ↓**1**	↑**7** ↓**0**	*↑22 ↓26*
T_2	(7)	*↑1 ↓3*	↑2 ↓2	*↑1 ↓5*	*↑1 ↓5*	↑**2** ↓**1**	↑**5** ↓**1**	↑**6** ↓**1**	↑**5** ↓**0**	↑**23** ↓**18**
T_3	(7)	*↑1 ↓3*	↑1 ↓1	*↑1 ↓2*	*↑2 ↓3*	↑**4** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**27** ↓**13**
T_4	(7)	*↑0 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓5*	*↑3 ↓4*	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	*↑23 ↓30*
T_5	(7)	*↑0 ↓4*	↑**2** ↓**1**	*↑0 ↓3*	↑**3** ↓**2**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**29** ↓**14**
T_6	(7)	*↑1 ↓2*	↑**3** ↓**2**	*↑1 ↓4*	*↑2 ↓4*	↑**4** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**29** ↓**16**
All	(54)	*↑15 ↓20*	↑**22** ↓**18**	*↑16 ↓26*	*↑19 ↓25*	↑**28** ↓**14**	↑**42** ↓**8**	↑**44** ↓**6**	↑**45** ↓**4**	↑**231** ↓**121**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**10** ↓**0**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**10** ↓**3**
10	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**10** ↓**4**
20	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**10** ↓**3**
40	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**11** ↓**1**
80	(2)	*↑0 ↓1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**13** ↓**1**
C	(6)	↑0 ↓0	↑**6** ↓**0**	↑**6** ↓**0**	↑**2** ↓**1**	*↑1 ↓3*	*↑1 ↓4*	*↑1 ↓3*	↑0 ↓0	↑**17** ↓**11**
S	(6)	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**47** ↓**1**
		GDBG								
F_{1a}	(6)	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	↑**1** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	*↑18 ↓19*
F_{1b}	(6)	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	↑1 ↓1	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	*↑18 ↓23*
F_2	(6)	*↑0 ↓6*	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓3*	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	*↑15 ↓23*
F_3	(6)	*↑0 ↓6*	↑**4** ↓**0**	*↑1 ↓2*	↑2 ↓2	↑**3** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**27** ↓**10**
F_4	(6)	*↑0 ↓6*	*↑0 ↓4*	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓5*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	*↑16 ↓23*
F_5	(6)	↑**6** ↓**0**	↑**4** ↓**2**	↑3 ↓3	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑13 ↓35*
F_6	(6)	*↑0 ↓6*	↑**1** ↓**0**	*↑2 ↓3*	↑**3** ↓**2**	↑**3** ↓**2**	↑**5** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**24** ↓**15**
T_1	(7)	*↑1 ↓4*	*↑1 ↓5*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓4*	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	*↑23 ↓28*
T_2	(7)	*↑1 ↓5*	↑3 ↓3	*↑1 ↓4*	*↑0 ↓6*	*↑0 ↓4*	↑**3** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	*↑18 ↓27*
T_3	(7)	*↑1 ↓5*	*↑1 ↓2*	*↑1 ↓3*	*↑1 ↓6*	*↑1 ↓2*	↑**4** ↓**1**	↑**5** ↓**1**	↑**6** ↓**1**	*↑20 ↓21*
T_4	(7)	*↑1 ↓6*	*↑1 ↓5*	*↑0 ↓6*	*↑1 ↓6*	*↑3 ↓4*	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	*↑24 ↓30*
T_5	(7)	*↑1 ↓6*	↑2 ↓2	*↑1 ↓4*	*↑1 ↓5*	↑**2** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**25** ↓**21**
T_6	(7)	*↑1 ↓6*	*↑1 ↓3*	↑2 ↓2	*↑1 ↓5*	*↑1 ↓2*	↑**3** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑21 ↓21
All	(54)	*↑11 ↓33*	↑**21** ↓**20**	*↑18 ↓25*	*↑13 ↓35*	*↑15 ↓20*	↑**35** ↓**10**	↑**41** ↓**10**	↑**41** ↓**7**	↑**195** ↓**160**

Table 6.8: SACDE vs CDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
5	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑9 ↓5**
10	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑9 ↓5**
20	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑10 ↓5**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑10 ↓3**
80	(2)	*↑0 ↓2*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑11 ↓3**
C	(6)	*↑0 ↓4*	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	*↑1 ↓3*	*↑0 ↓4*	*↑1 ↓5*	*↑1 ↓2*	*↑18 ↓20*
S	(6)	↑1 ↓1	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓1**
		GDBG								
F_{1a}	(6)	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓4*	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	*↑17 ↓24*
F_{1b}	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑17 ↓29*
F_2	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	*↑14 ↓30*
F_3	(6)	*↑0 ↓6*	**↑2 ↓1**	*↑0 ↓4*	*↑1 ↓5*	↑2 ↓2	**↑3 ↓0**	**↑4 ↓0**	**↑4 ↓0**	*↑16 ↓18*
F_4	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	*↑14 ↓30*
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑3 ↓3	*↑1 ↓3*	*↑1 ↓4*	*↑1 ↓5*	*↑1 ↓5*	**↑25 ↓20**
F_6	(6)	*↑0 ↓6*	*↑0 ↓2*	*↑0 ↓2*	**↑3 ↓1**	**↑3 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑21 ↓12**
T_1	(7)	*↑1 ↓5*	*↑1 ↓4*	*↑1 ↓4*	*↑1 ↓6*	*↑1 ↓6*	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	*↑23 ↓28*
T_2	(7)	*↑1 ↓6*	*↑1 ↓4*	*↑1 ↓5*	*↑1 ↓6*	*↑0 ↓5*	**↑4 ↓1**	**↑5 ↓1**	**↑4 ↓1**	*↑17 ↓29*
T_3	(7)	*↑1 ↓5*	*↑2 ↓4*	*↑1 ↓4*	*↑2 ↓5*	*↑1 ↓4*	**↑2 ↓1**	**↑4 ↓1**	**↑6 ↓1**	*↑19 ↓25*
T_4	(7)	*↑1 ↓6*	*↑1 ↓5*	*↑1 ↓5*	*↑0 ↓6*	*↑0 ↓4*	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	*↑21 ↓29*
T_5	(7)	*↑1 ↓6*	*↑2 ↓4*	*↑1 ↓5*	*↑1 ↓5*	*↑2 ↓4*	**↑4 ↓0**	**↑4 ↓1**	**↑5 ↓1**	*↑20 ↓26*
T_6	(7)	*↑1 ↓6*	*↑1 ↓5*	*↑1 ↓5*	*↑2 ↓5*	*↑2 ↓5*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑24 ↓26*
All	(54)	*↑7 ↓39*	*↑20 ↓26*	*↑18 ↓28*	*↑16 ↓35*	*↑13 ↓31*	**↑33 ↓8**	**↑38 ↓10**	**↑40 ↓7**	**↑185 ↓184**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑6 ↓9*
5	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑2 ↓12*
10	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓14*
20	(2)	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓13*
40	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓13*
80	(2)	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑3 ↓11*
C	(6)	*↑0 ↓5*	**↑6 ↓0**	**↑6 ↓0**	↑2 ↓2	*↑1 ↓4*	*↑0 ↓5*	*↑1 ↓5*	*↑1 ↓4*	*↑17 ↓25*
S	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓47*
		GDBG								
F_{1a}	(6)	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	*↑13 ↓27*
F_{1b}	(6)	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	**↑3 ↓2**	**↑6 ↓0**	**↑6 ↓0**	*↑15 ↓30*
F_2	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑2 ↓4*	**↑4 ↓2**	**↑5 ↓0**	*↑11 ↓35*
F_3	(6)	*↑0 ↓6*	**↑2 ↓1**	*↑0 ↓4*	*↑0 ↓4*	*↑1 ↓3*	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑19 ↓18**
F_4	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓4*	**↑4 ↓1**	**↑5 ↓0**	*↑10 ↓35*
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓2**	**↑43 ↓3**
F_6	(6)	*↑0 ↓6*	↑2 ↓2	*↑1 ↓2*	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑28 ↓10**
T_1	(7)	*↑1 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑1 ↓4*	*↑2 ↓4*	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑29 ↓20**
T_2	(7)	*↑1 ↓6*	*↑2 ↓4*	*↑1 ↓6*	*↑2 ↓5*	*↑1 ↓5*	↑2 ↓2	**↑6 ↓1**	**↑6 ↓1**	*↑21 ↓30*
T_3	(7)	*↑1 ↓5*	*↑1 ↓4*	*↑1 ↓5*	*↑1 ↓5*	*↑2 ↓4*	↑3 ↓3	**↑4 ↓1**	**↑7 ↓0**	*↑20 ↓27*
T_4	(7)	*↑1 ↓6*	*↑1 ↓5*	*↑1 ↓5*	*↑2 ↓5*	*↑2 ↓5*	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓1**	*↑26 ↓27*
T_5	(7)	*↑1 ↓6*	*↑3 ↓4*	*↑1 ↓4*	*↑1 ↓4*	*↑2 ↓4*	↑3 ↓3	**↑5 ↓2**	**↑5 ↓0**	*↑21 ↓27*
T_6	(7)	*↑1 ↓6*	*↑1 ↓5*	*↑1 ↓5*	*↑1 ↓5*	*↑2 ↓4*	**↑3 ↓2**	**↑6 ↓0**	**↑7 ↓0**	*↑22 ↓27*
All	(54)	*↑6 ↓44*	*↑16 ↓32*	*↑13 ↓34*	*↑10 ↓36*	*↑12 ↓36*	**↑24 ↓21**	**↑36 ↓15**	**↑39 ↓12**	*↑156 ↓230*
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑4 ↓2**	**↑8 ↓1**	**↑7 ↓1**	**↑6 ↓2**	**↑5 ↓2**	**↑4 ↓2**	**↑6 ↓2**	**↑7 ↓1**	**↑47 ↓13**
5	(10)	**↑4 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑4 ↓2**	*↑4 ↓6*	*↑3 ↓6*	*↑3 ↓5*	**↑3 ↓1**	**↑39 ↓25**
10	(10)	**↑5 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑5 ↓4**	↑5 ↓5	*↑4 ↓5*	↑4 ↓4	**↑4 ↓2**	**↑45 ↓25**
20	(10)	**↑6 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑6 ↓2**	**↑5 ↓4**	**↑6 ↓4**	**↑5 ↓4**	**↑5 ↓3**	**↑51 ↓21**
40	(10)	**↑5 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑6 ↓2**	**↑6 ↓4**	**↑6 ↓3**	**↑6 ↓3**	**↑55 ↓17**
80	(10)	*↑3 ↓5*	**↑9 ↓1**	**↑9 ↓0**	**↑9 ↓1**	**↑8 ↓1**	**↑7 ↓2**	**↑7 ↓3**	**↑6 ↓2**	**↑58 ↓15**
C	(30)	**↑11 ↓9**	**↑29 ↓0**	**↑28 ↓0**	**↑14 ↓6**	*↑9 ↓14*	*↑7 ↓17*	*↑8 ↓15*	**↑8 ↓6**	**↑114 ↓67**
S	(30)	**↑16 ↓8**	**↑24 ↓6**	**↑24 ↓5**	**↑24 ↓6**	**↑24 ↓6**	**↑23 ↓6**	**↑23 ↓6**	**↑23 ↓6**	**↑181 ↓49**
		GDBG								
F_{1a}	(30)	*↑1 ↓15*	*↑1 ↓21*	*↑0 ↓26*	*↑0 ↓24*	*↑8 ↓10*	**↑26 ↓0**	**↑27 ↓0**	**↑30 ↓0**	*↑93 ↓96*
F_{1b}	(30)	*↑1 ↓19*	*↑0 ↓26*	*↑0 ↓29*	*↑2 ↓25*	*↑9 ↓13*	**↑25 ↓2**	**↑30 ↓0**	**↑30 ↓0**	*↑97 ↓114*
F_2	(30)	*↑0 ↓24*	*↑0 ↓21*	*↑0 ↓24*	*↑1 ↓29*	*↑5 ↓17*	**↑20 ↓4**	**↑27 ↓2**	**↑29 ↓0**	*↑82 ↓121*
F_3	(30)	*↑0 ↓23*	**↑18 ↓2**	*↑8 ↓10*	**↑15 ↓11**	**↑18 ↓5**	**↑25 ↓0**	**↑27 ↓0**	**↑28 ↓0**	**↑139 ↓51**
F_4	(30)	*↑0 ↓26*	*↑4 ↓20*	*↑0 ↓23*	*↑0 ↓28*	*↑2 ↓22*	**↑17 ↓5**	**↑27 ↓1**	**↑28 ↓0**	*↑78 ↓125*
F_5	(30)	**↑20 ↓7**	**↑16 ↓14**	↑15 ↓15	*↑9 ↓21*	*↑6 ↓21*	*↑8 ↓19*	*↑9 ↓21*	*↑9 ↓20*	*↑92 ↓138*
F_6	(30)	*↑4 ↓20*	**↑9 ↓7**	*↑8 ↓12*	**↑14 ↓6**	**↑20 ↓6**	**↑27 ↓0**	**↑28 ↓1**	**↑27 ↓1**	**↑137 ↓53**
T_1	(35)	*↑3 ↓19*	*↑6 ↓25*	*↑6 ↓26*	*↑6 ↓27*	*↑9 ↓20*	**↑31 ↓3**	**↑32 ↓3**	**↑33 ↓2**	**↑126 ↓125**
T_2	(35)	*↑5 ↓21*	*↑11 ↓16*	*↑5 ↓26*	*↑5 ↓27*	*↑6 ↓18*	**↑20 ↓5**	**↑29 ↓5**	**↑27 ↓4**	*↑108 ↓122*
T_3	(35)	*↑5 ↓19*	*↑8 ↓12*	*↑6 ↓19*	*↑8 ↓22*	*↑10 ↓13*	**↑20 ↓7**	**↑25 ↓5**	**↑31 ↓4**	**↑113 ↓101**
T_4	(35)	*↑3 ↓31*	*↑5 ↓26*	*↑4 ↓28*	*↑6 ↓25*	*↑13 ↓18*	**↑30 ↓3**	**↑32 ↓3**	**↑31 ↓4**	*↑124 ↓138*
T_5	(35)	*↑3 ↓24*	*↑9 ↓14*	*↑3 ↓20*	*↑8 ↓21*	**↑17 ↓12**	**↑24 ↓7**	**↑27 ↓6**	**↑28 ↓4**	**↑119 ↓108**
T_6	(35)	*↑7 ↓20*	*↑9 ↓18*	*↑7 ↓20*	*↑8 ↓22*	↑13 ↓13	**↑23 ↓5**	**↑30 ↓3**	**↑31 ↓3**	**↑128 ↓104**
All	(270)	*↑53 ↓151*	*↑101 ↓117*	*↑83 ↓144*	*↑79 ↓156*	*↑101 ↓114*	**↑178 ↓53**	**↑206 ↓46**	**↑212 ↓33**	**↑1013 ↓814**

than those found for SABrNorRes in all cases. SACDE does, however, have the advantage that the parameters controlling the scale and crossover factors and the Brownian radius are self-adapted and consequently do not have to be fine-tuned. This advantage is achieved by incurring only a relatively small performance penalty.

The performance of SACDE, and the fact that it removes the need to tune several parameters, makes SACDE a viable alternative to its predecessor algorithms. The following research question considers the scalability of the various self-adaptive approaches proposed in this chapter.

6.4.7 Research Question 6

How do the self-adaptive algorithms scale under factors that influence the complexity of a dynamic optimisation problem?

The set of environments used in research questions 2, 4 and 5 was used to observe how jSA2Ran, SABrNorRes, and SACDE scale with respect to change period, number of dimensions, change severity, underlying function, and change type. The results for DynDE and CDE are included to aid the comparisons. The volume of experimental results makes it impossible to plot figures and discuss each of the dynamic environments. The following sections rather focus on determining the general trends that can be observed in the experimental results.

The rest of this section is structured as follows: Section 6.4.7.1 describes the scalability of the algorithms with respect to change period. Section 6.4.7.2 discusses the effect of varying the number of dimensions, while Section 6.4.7.3 describes the effect of varying the change severity. The scalability of the algorithms with respect to the underlying function and the change type is discussed in Sections 6.4.7.4 and 6.4.7.5, respectively. The findings of the scalability study are summarised in Section 6.4.7.6.

6.4.7.1 Trends from Varying the Change Period

The change period is an important consideration for self-adaptive algorithms in dynamic environments. A large change period results in a large number of function evaluations that are available between changes in the environment. A self-adaptive algorithm would consequently have more available function evaluations to adapt its values effectively.

A comparison of the offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE found that jSA2Ran and CDE generally yielded very similar offline errors, while SABrNorRes and SACDE generally yielded similar offline errors. Sections 6.4.3, 6.4.5 and 6.4.6 found that jSA2Ran, SABrNorRes and SACDE were all less effective on low change periods than on high change periods. This trend can be seen in Figure 6.9, which gives the offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE for various settings of change period on the GDBG function F_{1b} with change type T_4 in five dimensions. CDE and jSA2Ran gave the lowest offline errors at low change periods, while SABrNorRes and SACDE gave the lowest offline errors at high change periods. Chapter 4 found that DynDE performed better than CDE at high change periods. This situation is remedied by SABrNorRes and SACDE which generally performed better than DynDE at high change periods.

The trends described above were present in the majority of evironments, but notable exceptions did occur. The previous research questions found a strong correlation between the underlying function and the comparative performance of the self-adaptive algorithms. This dependence on the underlying function is evident when comparing Figure 6.9 to Figure 6.10. Figure 6.10 gives the same information as Figure 6.9, but on the GDBG function F_3. SABrNorRes and SACDE outperformed the other algorithms by a wide margin on function F_3.

The underlying function can also have a detrimental effect on the performances of the self-adaptive algorithms. Figure 6.11 shows the offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on function F_5, with change type T_4, in 10 dimensions. The offline errors of SABrNorRes and SACDE were considerably higher than those of the other algorithms, even at high change periods. At high dimensions on the same problem, shown in Figure 6.12 for 100 dimensions, SABrNorRes and SACDE performed better than the other algorithms at low change periods, but were weaker than jSA2Ran at high change periods.

6.4.7.2 Trends from Varying the Number of Dimensions

Section 4.6.4.3 found that the offline error of DynDE and CDE increased as the number of dimensions was increased. This trend also occurred for jSA2Ran, SABrNorRes and

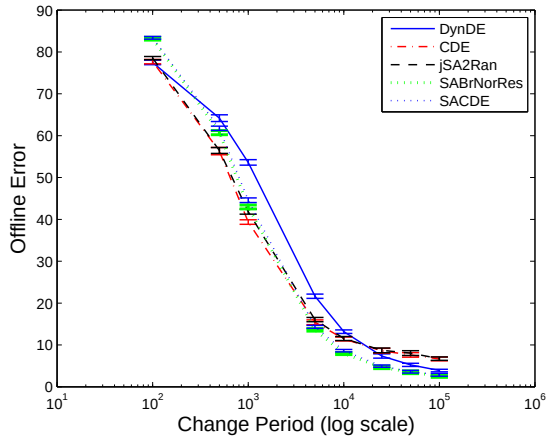

Figure 6.9: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_{1b} and change type T_4 for various settings of change period in 5 dimensions.

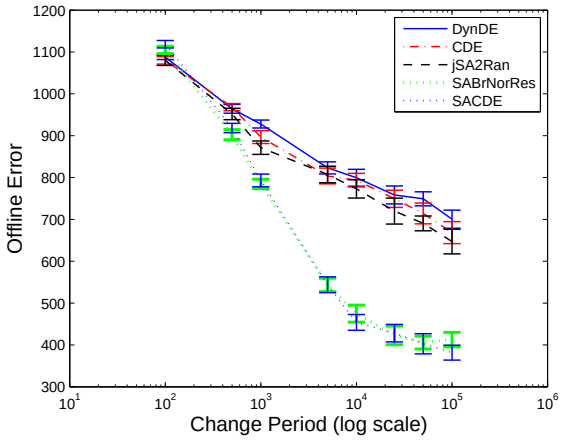

Figure 6.10: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_3 and change type T_4 for various settings of change period in 5 dimensions.

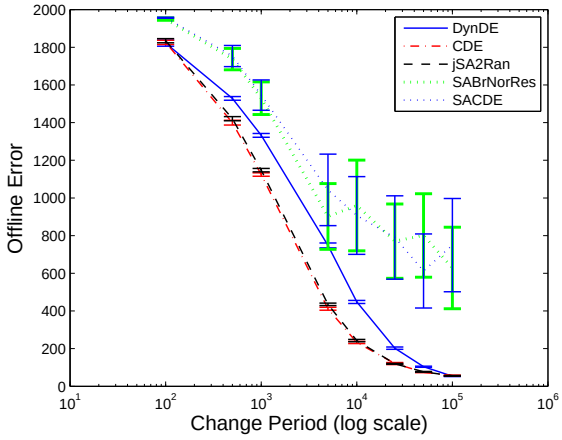

Figure 6.11: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_5 and change type T_4 for various settings of change period in 10 dimensions.

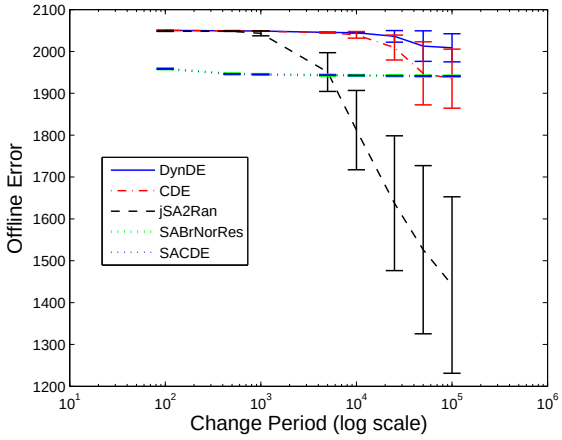

Figure 6.12: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_5 and change type T_4 for various settings of change period in 100 dimensions.

SACDE. The self-adaptive algorithms generally did not perform better than CDE at low change periods, as is evident from Figure 6.13, which gives the offline errors for various settings of number dimensions of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on function F_4, with change type T_1 and a change period of 5 000 function evaluations.

The situation changes at high change periods, where the self-adaptive algorithms performed better, especially in high dimensions. Figure 6.14 gives the same information as Figure 6.13, but with a change period of 100 000 function evaluations. The self-adaptive algorithms, jSA2Ran, SABrNorRes and SACDE, all performed better than CDE in dimensions greater than 10. Large improvements were found. SABrNorRes and SACDE performed better or similar to DynDE in low dimensions, where DynDE generally performed better than CDE.

The scalability of the algorithms with respect to dimension was also found to be dependent on the underlying function, which caused exceptions to the general trends. Figure 6.15 gives the offline errors for various settings of number dimensions, for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on function F_3, with change type T_1, and a change period of 5 000 function evaluations. SABrNorRes and SACDE performed better than the other algorithms on this function, despite the fact that a low change period was used. Note that, at high dimensions, SACDE performed noticeably better than SABrNorRes, which illustrates the functional dependence of the interaction between the two self-adaptive approaches.

The underlying function also proved detrimental to the performance of the algorithms in a minority of the experiments. Figure 6.16 gives the offline errors for various settings of number dimensions for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on function F_5, with change type T_3, and a change period of 100 000 function evaluations. SABrNorRes and SACDE performed worse than the other algorithms at low dimensions, despite the high change period. The performance of SABrNorRes and SACDE recovers at high dimensions where they performed similar to, or better than jSA2Ran.

6.4.7.3 Trends from Varying the Change Severity

The main reason for including Brownian individuals in the sub-populations is to increase diversity, which in turn assists the algorithms to respond effectively to changes in the

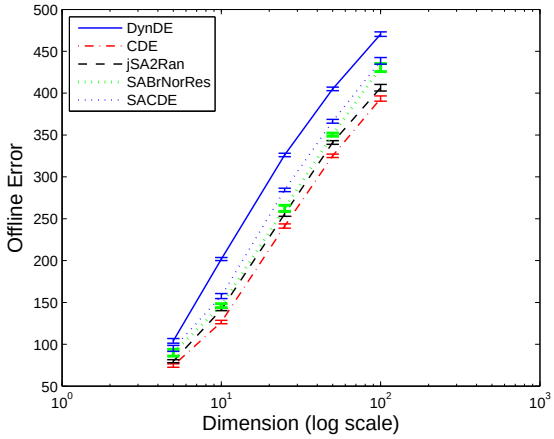

Figure 6.13: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_4 and change type T_1 for various settings of dimension with a change period of 5 000 function evaluations.

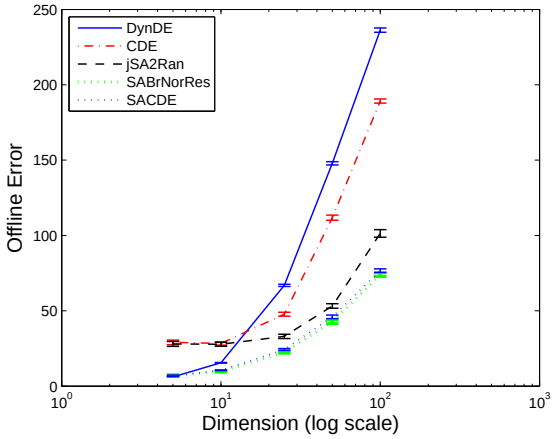

Figure 6.14: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_4 and change type T_1 for various settings of dimension with a change period of 100 000 function evaluations.

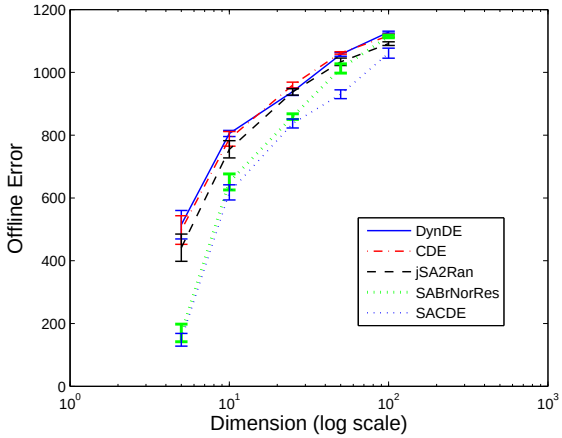

Figure 6.15: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_3 and change type T_1 for various settings of dimension with a change period of 25 000 function evaluations.

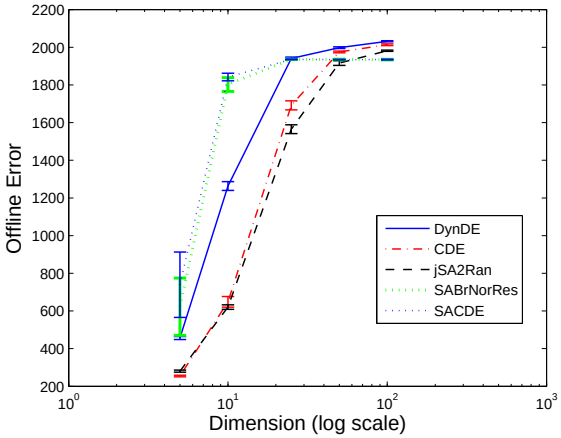

Figure 6.16: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_5 and change type T_3 for various settings of dimension with a change period of 100 000 function evaluations.

environment. MPSO, the PSO based algorithm aimed at DOPs (refer to Section 3.3.3.2), accepts the change severity within the dynamic environment as a parameter. This parameter is used to determine the cloud radius of the quantum particles, so that more diversity is injected into the swarms when changes are more severe. The self-adaptive Brownian radius employed in SABrNorRes and SACDE thus has the potential of achieving lower offline errors than CDE over a range of values for change severity by adapting the Brownian radius to a value appropriate to the environment.

Figure 6.17 gives the offline errors, for various settings of change severity, for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the conical MPB function, in 10 dimensions, and a change period of 500. The figure shows that SABrNorRes and SACDE indeed yielded lower offline errors on this environment in which CDE performed similar to DynDE. The self-adaptive scale and crossover factors of jSA2Ran also yielded improvements on the intermediate range of change severities.

The improvements that were found by using the self-adaptive approaches continued to high change periods, of which Figure 6.18 is an example, as it gives the same information as Figure 6.17, but with a change period of 100 000 function evaluations. The improvements were less pronounced than at low change periods.

A dependence on the underlying function was found with respect the change severity scalability at high dimensions. Figure 6.19 gives the offline errors for various settings of change severity for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the spherical MPB function, in 5 dimensions, and a change period of 5 000. The self-adaptive approaches yielded considerable improvements over DynDE and CDE, as was the case when using the conical function. A large degradation in the performances of SABrNorRes and SACDE were found in 100 dimensions when using the spherical function (refer to Figure 6.20). This poor performance is an exception to the general case where SABrNorRes and SACDE performed better than the other algorithms. The effectiveness of the self-adaptive Brownian radius approach thus depends on the dimension and the underlying function.

6.4.7.4 Trends from Various Functions

The previous sections within this research question have noted the dependence of the self-adaptive approaches on the underlying function. The algorithms described in this chapter

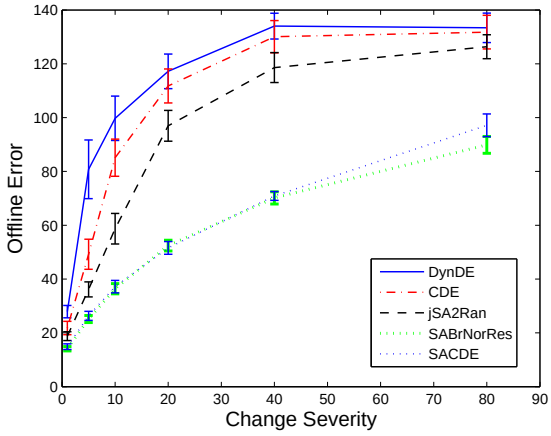

Figure 6.17: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the conical peak function in 10 dimensions for various settings of change severity with a change period of 500 function evaluations.

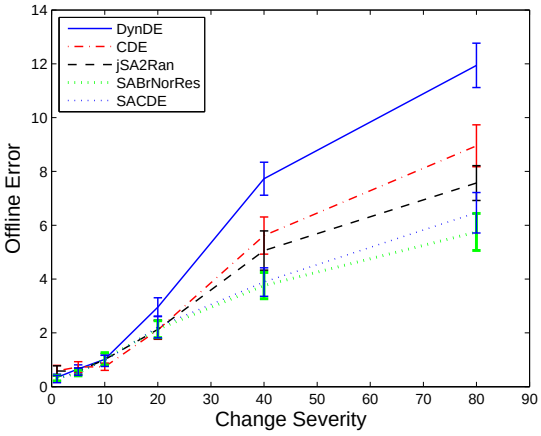

Figure 6.18: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the conical peak function in 10 dimensions for various settings of change severity with a change period of 100 000 function evaluations.

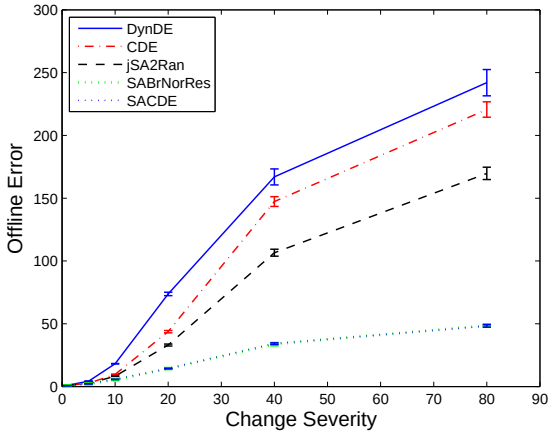

Figure 6.19: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the spherical peak function in 5 dimensions for various settings of change severity with a change period of 5 000 function evaluations.

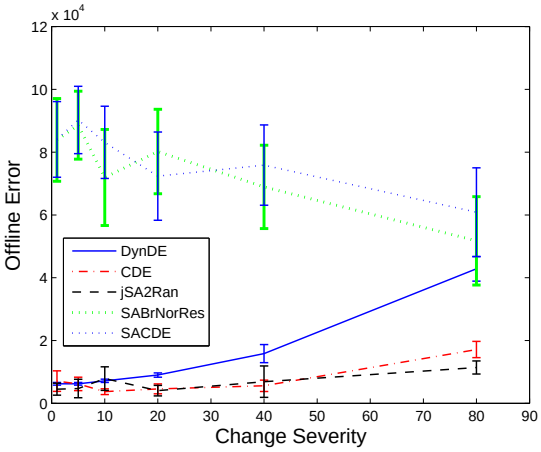

Figure 6.20: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the spherical peak function in 100 dimensions for various settings of change severity with a change period of 5 000 function evaluations.

generally followed similar trends as DynDE and CDE on the various functions that were evaluated, but several noteworthy exceptions are described in this section.

Figure 6.21 gives the offline errors for various underlying GDBG functions for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE, in 5 dimensions, a change period of 50 000 function evaluations and change type T_1. Section 4.6.4.5 found that function F_3 was particularly challenging to DynDE and CDE. Figure 6.21 shows that SABrNorRes and SACDE performed considerably better than the other algorithms on this function. The improved performance was, however, found to be dependent on the number of dimensions. Consider Figure 6.22, which gives the same information as Figure 6.21, but in 10 dimensions. Here the performance difference between SABrNorRes and SACDE, and the other algorithms has diminished.

The self-adaptive scale and crossover approach also suffered from function-dependence. Figure 6.23 gives the offline errors for various underlying GDBG functions for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE, in 50 dimensions, a change period of 50 000 function evaluations and change type T_1. The figure shows that jSA2Ran performed considerably better than the other algorithms on function F_5. This advantage, however, narrows when the number of dimensions is increased to 100, as shown in Figure 6.24.

Cases where the self-adaptive algorithms performed worse than DynDE and CDE were also found to be function-dependent. SABrNorRes and SACDE performed worse than the other algorithms by a wide margin in Figure 6.23, while SABrNorRes and SACDE performed better than the other algorithms on the other functions. SABrNorRes and SACDE recover from poor performance when the number of dimensions was increased to 100 (refer to Figure 6.24).

The scalability of the self-adaptive algorithms with respect to the underlying function is thus dependent on the number of dimensions, although, in general, trends similar to those of DynDE and CDE were found.

6.4.7.5 Trends from Various Change Types

The effect of the change type on the self-adaptive algorithms was found to depend on the underlying function, as was the case with DynDE and CDE. However, the change period did, in a small number of experimental cases, influence the effect of the change type.

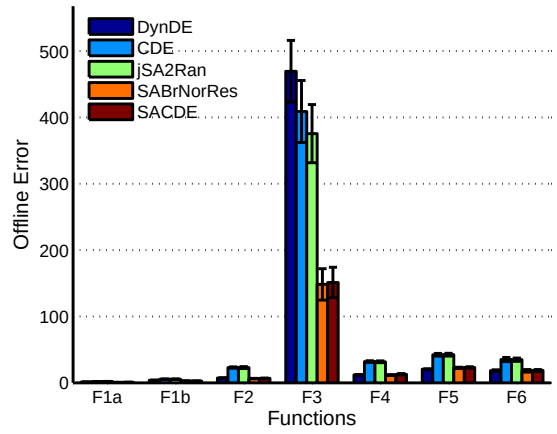

Figure 6.21: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 5 dimensions with a change period of 50 000 function evaluations.

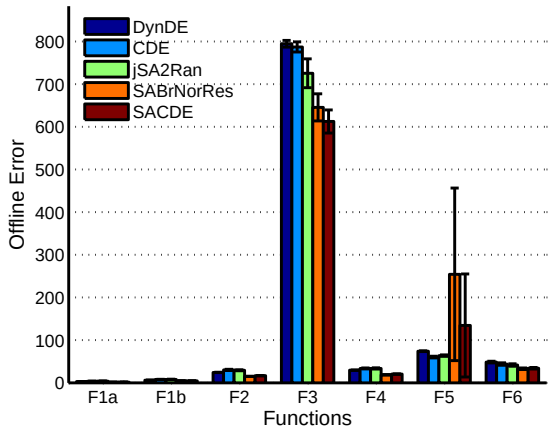

Figure 6.22: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 10 dimensions with a change period of 50 000 function evaluations.

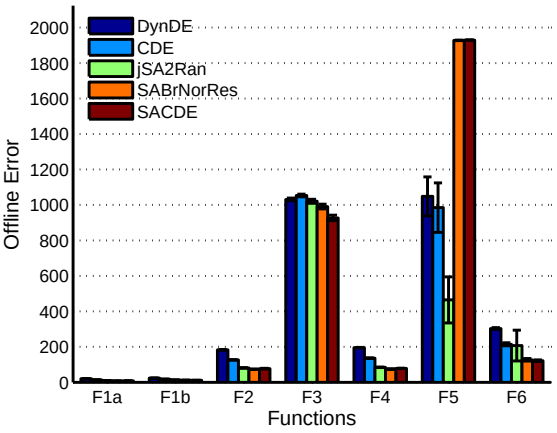

Figure 6.23: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 50 dimensions with a change period of 50 000 function evaluations.

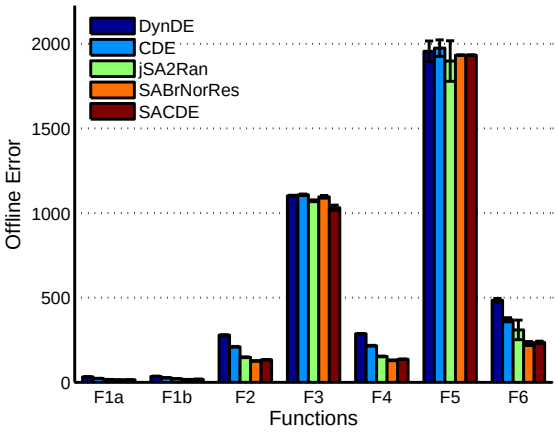

Figure 6.24: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 100 dimensions with a change period of 50 000 function evaluations.

Figure 6.25 gives the offline errors for various change types, for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE, in 5 dimensions, with a change period of 1 000 function evaluations and function F_5 on the GDBG. This is a function that was identified in the previous section as one on which SABrNorRes and SACDE was ineffective. Figure 6.25 show that SABrNorRes and SACDE gave higher offline errors than the other algorithms, but that all algorithms follow similar trends over different change types. Conversely, Figure 6.26, which presents the same results when using a change period of 50 000, shows very different behaviour for SABrNorRes and SACDE when using change types T_3, T_5 and T_6. The inferior performance of SABrNorRes and SACDE on this function is thus clearly linked to the change type.

Figure 6.27 gives the offline errors for various change types, for DynDE, CDE, jSA2Ran, SABrNorRes and SACDE, in 50 dimensions, with a change period of 1 000 function evaluations on function F_2. Once again, all the algorithms exhibited offline errors similar to each other over the different change types. Figure 6.28 gives the same information as Figure 6.27, but when using a change period of 50 000 function evaluations. The self-adaptive algorithms performed noticeably better than DynDE and CDE on change types T_1, T_4 and T_6.

The cases where the self-adaptive algorithms performed better than the other algorithms thus also depend on the change type. Note that SABrNorRes and SACDE performed worse than DynDE and CDE on change type T_6 in Figure 6.26, but performed better than DynDE and CDE on the same change type in Figure 6.28. The influence of the change type is thus strongly linked to the underlying function.

6.4.7.6 Summary for Research Question 6

The scalability study found that in the vast majority of cases, CDE and jSA2Ran scaled similarly with respect to the various benchmark settings, while SABrNorRes and SACDE behaved similarly. The trends that emerged from varying the benchmark settings are summarised below for each setting:

Change Period: Increasing the change period resulted in a reduction of offline error for all algorithms. The self-adaptive algorithms were found to be comparatively more

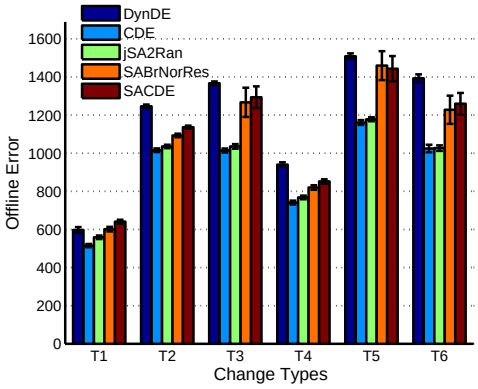

Figure 6.25: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_5 for various change types in 5 dimensions with a change period of 1 000 function evaluations.

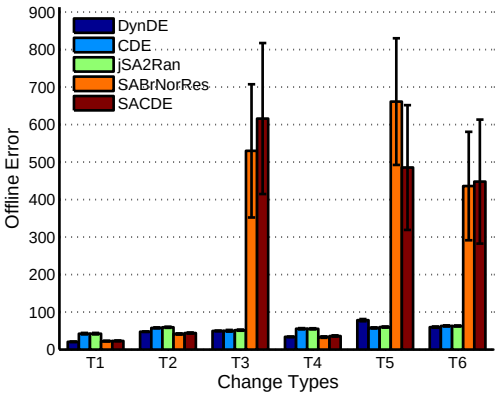

Figure 6.26: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_5 for various change types in 5 dimensions with a change period of 50 000 function evaluations.

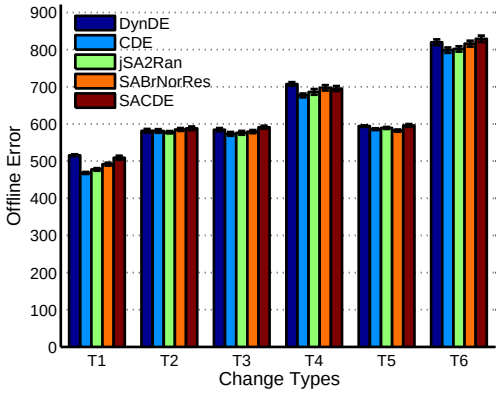

Figure 6.27: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_2 for various change types in 50 dimensions with a change period of 1 000 function evaluations.

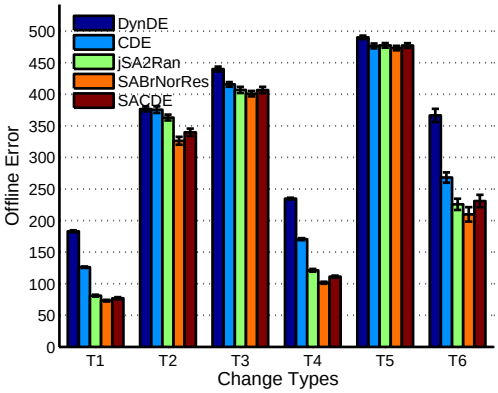

Figure 6.28: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_2 for various change types in 50 dimensions with a change period of 50 000 function evaluations.

effective at high change periods, as more function evaluations are available for the adaptation process.

Number of Dimensions: Larger numbers of dimensions resulted in larger offline errors. SABrNorRes and SACDE were found to be generally inferior to CDE and jSA2Ran at high dimensional problems, especially when a low change period was used.

Change Severity: More severe changes in the environment result in higher offline errors as information gathered before changes becomes less relevant. The self-adaptive algorithms were more effective than DynDE and CDE on environments where the changes were severe.

Function: A strong correlation between the effectiveness of the self-adaptive algorithms and the underlying function was found. SABrNorRes and SACDE outperformed the other algorithms by a wide margin on some functions, notably F_3, but also performed comparatively poorly on other functions, for example, F_5.

Change Type: The effect of the change type was found to be related to the function that was being optimised. Cases where SABrNorRes and SACDE performed much worse than the other algorithms were found to be isolated to specific, function dependent, change types. The self-adaptive algorithms were also found to be more effective on specific change types which depended on the underlying function.

6.4.8 Research Question 7

What are the convergence profiles of the self-adaptive algorithms?

This research question compares the convergence profiles of jSA2Ran, SABrNorRes and SACDE to those of DynDE and CDE to provide insights into the functioning of the algorithms. The algorithms are compared in terms of offline and current errors, and the diversity of the algorithms is compared in terms of overall diversity and average diversity per sub-population (as calculated using equation (4.8) on page 156). The values assumed during the optimisation process for the scale factor, crossover factor and the Brownian radius, are also investigated. All the figures in this section present results averaged over 30 repeats of each experiment.

Figure 6.29 gives the offline and current errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE during the first 10 changes in the environment on the Scenario 2 settings of the MPB. The current errors of the self-adaptive approaches decreased at a faster rate, and reached a lower value than those of DynDE and CDE during the period before the first change in the environment. The current errors of SABrNorRes and SACDE reached lower values than those of jSA2Ran. The fast initial decrease in current error resulted in considerably lower initial offline errors for the self-adaptive algorithms. The self-adaptive algorithms are thus more effective at initially discovering optima in the environment. The performance difference between the algorithms became less noticeable later in the optimisation process, as none of the algorithms clearly performed better than the others (in terms of current error) after the fifth change in the environment.

Section 4.6.6 found that the reason why DynDE performed better than CDE at high change periods was that DynDE's current error eventually reached a lower value than that of CDE, despite the fact that CDE's current error initially decreased faster than that of DynDE. SABrNorRes and SACDE did not suffer from this problem, as can be seen in Figure 6.30, which gives the same information as Figure 6.29, but with a change period of 100 000 function evaluations. The self-adaptive algorithms generally reached lower current errors than CDE, while SABrNorRes and SACDE generally reached current errors as low as those of DynDE. This explains the trend where SABrNorRes and SACDE performed better than the other algorithms at high change periods, which was found in Section 6.4.7.1.

Figure 6.31 gives the diversity and average sub-population diversity of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE during the first 10 changes in the environment on the Scenario 2 settings of the MPB. CDE and jSA2Ran exhibits virtually identical diversity profiles, which explains the similar scaling behaviour observed in the previous section. SABrNorRes and SACDE exhibits similar diversity profiles which, in contrast to the other algorithms, show clear reactions to the changes in the environment. The average sub-population diversity of SABrNorRes and SACDE increased dramatically after a change in the environment, and was mirrored by a smaller increase in overall diversity. This increase in diversity is caused by resetting the value from which the Brownian radius is selected, when a change in the environment occurs. The Brownian individuals which are created

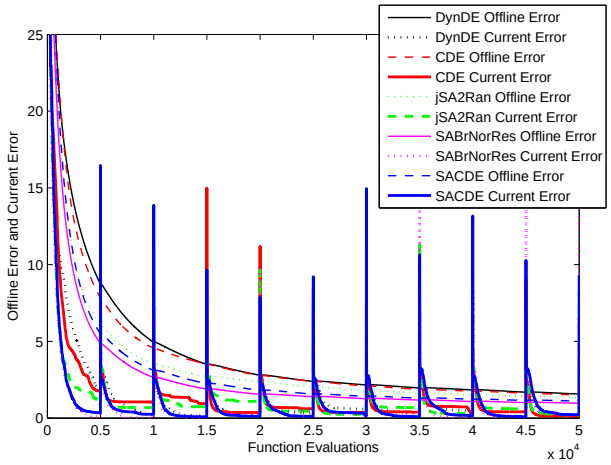

Figure 6.29: Offline and current errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB with the Scenario 2 settings.

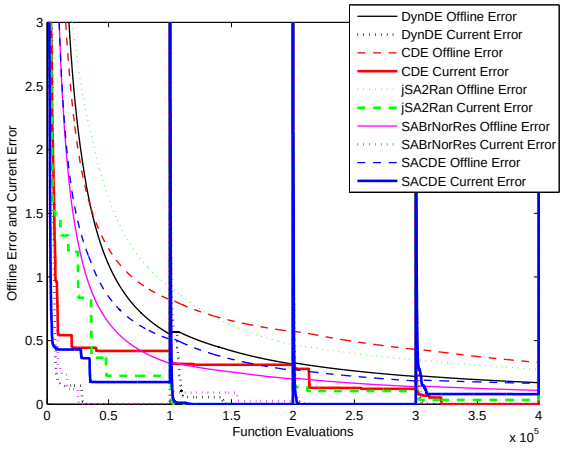

Figure 6.30: Offline and current errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the Scenario 2 settings with a change period of 100 000.

using a large Brownian radius are typically located at a large Euclidean distance from the best individuals in the sub-populations, which results in high diversity. The average sub-population diversity of SABrNorRes and SACDE dropped sharply after a change in the environment, but always maintained a higher value than that of the other algorithms.

Figure 6.32 gives the scale and crossover factors adapted by jSA2Ran, the value from which the Brownian radius is calculated by SABrNorRes, and all three previously mentioned for SACDE during the first 10 changes in the environment on the Scenario 2 settings of the MPB. The scale and crossover factors of jSA2Ran and SACDE followed the same trends as discussed in Section 6.2.2, with the scale factor converging to a value around 0.63 and the crossover factor converging to values around 0.52. The behaviour of the scale and crossover factor adapting process did thus not change noticeably when used in conjunction with the adaptive Brownian radius component.

The value used to select the Brownian radius is reset after changes in the environment, which causes the periodic spikes that are visible in Figure 6.32, for both SABrNorRes and SACDE. A rapid decrease in the Brownian value occurs directly after a change in the environment, but the value never dropped below 1.53. The Brownian radius, as calculated using equation (6.1), is thus likely to be greater than the default value of $r_{brown} = 0.2$, which was used in the previous chapters. This explains the higher overall average diversity

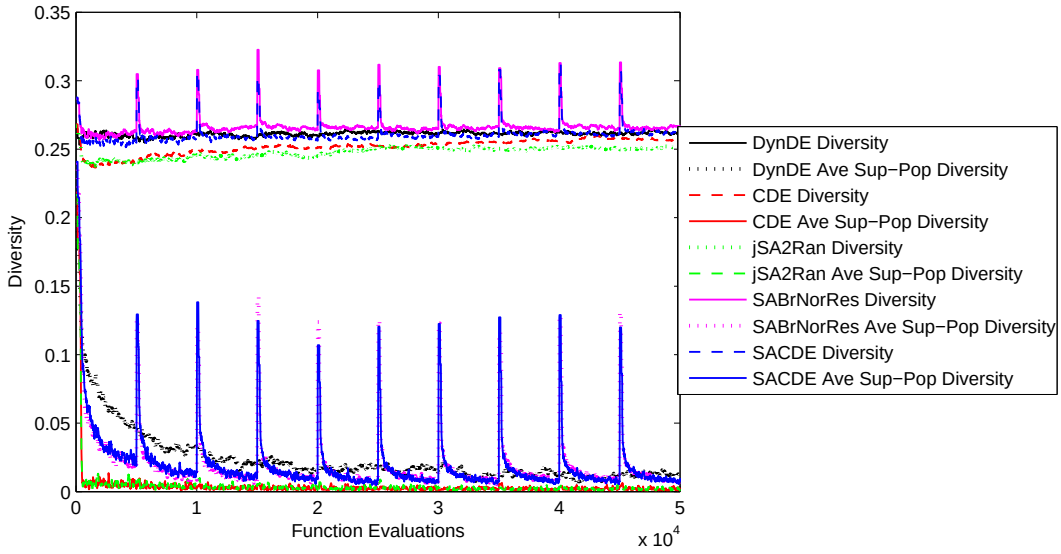

Figure 6.31: Diversity profiles of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB with the Scenario 2 settings.

per sub-populations which was observed for SABrNorRes and SACDE in comparison with the other algorithms.

6.4.9 Research Question 8

How do the self-adaptive components affect the performance of DynPopDE?

The previous sections identified jSA2Ran and SABrNorRes as the more effective of the two self-adaptive approaches. These were combined to form SACDE. This section investigates a new algorithm, SADynPopDE, which is formed by incorporating the self-adaptive scale factor, crossover factor and Brownian radius components of SACDE into DynPopDE.

SADynPopDE is evaluated on three sets of experiments. The performance analysis of SADynPopDE on variations of the standard set, which has been used to evaluate jSA2Ran, SABrNorRes and SACDE in this chapter, is discussed in Section 6.4.9.1. SADynPopDE is evaluated on environments with various numbers of optima in Section 6.4.9.2, and on environments in which the number of optima fluctuates in Section 6.4.9.3. A summary of the findings of this research question is provided in Section 6.4.9.4.

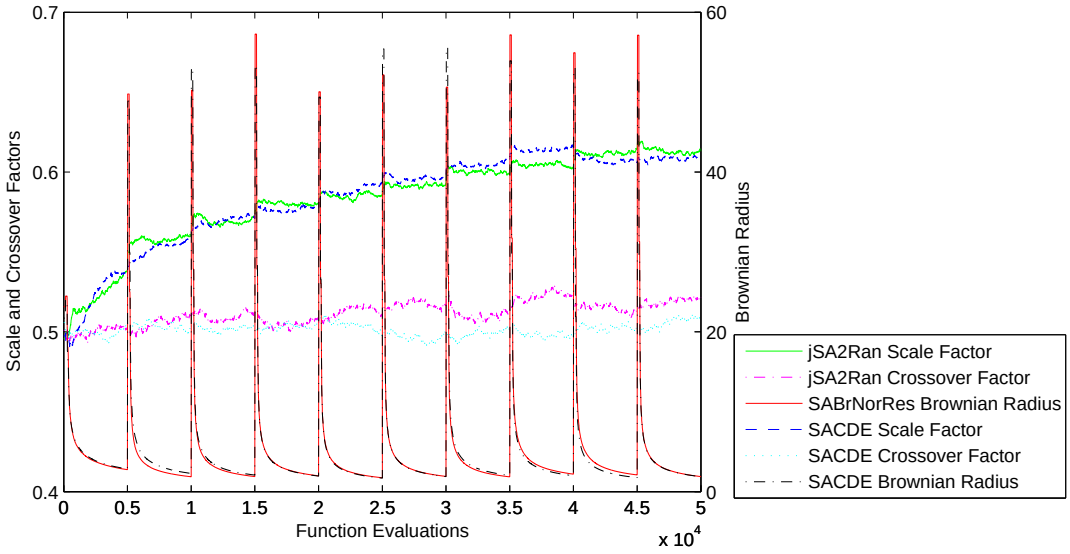

Figure 6.32: Self-adaptive value profiles of jSA2Ran, SABrNorRes and SACDE on the MPB with the Scenario 2 settings.

6.4.9.1 SADynPopDE on Variations of the Standard Set

Chapter 5 found that DynPopDE did not scale well over different functions. DynPopDE performed statistically significantly worse than CDE on 1 257 of the 2 160 variations of the standard set, and better on only 369 environments (refer to Section 5.3.9). SADynPopDE was evaluated on the same set of experimental environment and compared to CDE. SADynPopDE performed slightly better in comparison to CDE; it performed significantly better than CDE in 624 environments, but worse in 1 294 of the 2 160 environments (the analysis is given in Appendix D). The average percentage improvement of -11.50% was found for DynPopDE over CDE, while the average percentage improvement of SADynPopDE over CDE was -6%.

SADynPopDE was compared to SACDE and the detailed results of the performance analysis are given in Appendix D. SADynPopDE performed better than SACDE in 321 of the 2 160 experiments, and worse in 1 414 experiments. Most of the cases where SADynPopDE performed better more often than SACDE, occurred at a change period of 100, where SADynPopDE has an advantage, as it commences with a single sub-population. The average percentage improvement was found to be -13.18%. The margin by which SADynPopDE was inferior to SACDE is thus wider than the margin by which DynPopDE was

inferior to CDE, despite the fact that the incorporation of the self-adaptive components improved the relative performance of DynPopDE with respect to CDE. The incorporation of the self-adaptive components into DynPopDE, thus did not solve the algorithm's scaling problem with respect to the underlying function.

6.4.9.2 SADynPopDE on the n_p Standard Set

DynPopDE showed a considerable improvement over CDE on environments that used various settings for the number of peaks (refer to Section 5.3.4). This section investigates the performance of the self-adaptive algorithms on the n_p standard set to determine whether the self-adaptive components yield improved results.

The results of a performance analysis comparing SADynPopDE to DynPopDE are given in Table 6.9. SADynPopDE performed better than DynPopDE in 83 of the 480 environments, but worse in 287 environments. Environments where SADynPopDE performed better than DynPopDE mainly used very high or very low change periods. The average percentage improvement of SADynPopDE over DynPopDE was found to be -11.49%. These results indicate that the self-adaptive components identified as effective in SACDE do not improve the performance of DynPopDE in environments with unknown numbers of peaks.

SACDE was also evaluated on the n_p standard set in order to determine whether the self-adaptive components are beneficial to CDE over a range of settings for the number of peaks. The analysis which compares the results of SACDE to those of CDE is given in Table 6.10. SACDE was found to perform statistically significantly better than CDE in 244 of the 480 experiments, and worse in only 92 experiments. SACDE performed better more often than CDE, except in 100 dimensions.

The average percentage improvement of SACDE over CDE over all experiments was found to be 12.16%. The APIs per dimension were found to be 2.34%, 16.83%, 34.02%, 37.27% and -29.67% for 5, 10, 25, 50 and 100 dimensions respectively. The percentage improvement thus increases with the number of dimensions, with the exception of the 100 dimensional case, where CDE performed better than SACDE by a wide margin. The APIs per change period were found to be 3.63%, 25.82%, 24.12%, 10.00%, 5.78%, 5.65%, 10.19% and 12.08% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and

Table 6.9: SADynPopDE vs DynPopDE performance analysis on the n_p standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓13
10	(2)	↑0 ↓0	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓9
25	(2)	**↑1 ↓0**	↑0 ↓1	↑0 ↓2	↑1 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑2 ↓8
50	(2)	↑1 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑1 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑2 ↓10
100	(2)	↑1 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑1 ↓11
200	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓13
C	(6)	**↑3 ↓0**	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑3 ↓40
S	(6)	↑0 ↓4	↑0 ↓6	↑0 ↓6	↑1 ↓1	↑1 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑2 ↓24
All	(12)	↑3 ↓4	↑0 ↓11	↑0 ↓12	↑1 ↓7	↑1 ↓8	↑0 ↓8	↑0 ↓7	↑0 ↓7	↑5 ↓64
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑1 ↓1	**↑1 ↓0**	↑3 ↓7
10	(2)	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	↑4 ↓6
25	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑1 ↓1	↑0 ↓1	↑0 ↓0	**↑1 ↓0**	↑3 ↓6
50	(2)	**↑1 ↓0**	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑1 ↓1	↑1 ↓1	**↑2 ↓0**	↑5 ↓7
100	(2)	↑0 ↓0	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓0	↑0 ↓9
200	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑1 ↓11
C	(6)	**↑4 ↓0**	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓4	↑1 ↓1	↑5 ↓33
S	(6)	↑0 ↓1	↑0 ↓5	↑0 ↓4	↑0 ↓1	↑2 ↓2	**↑2 ↓0**	**↑3 ↓0**	**↑4 ↓0**	↑11 ↓13
All	(12)	**↑4 ↓1**	↑0 ↓10	↑0 ↓9	↑0 ↓7	↑2 ↓8	↑2 ↓6	↑3 ↓4	**↑5 ↓1**	↑16 ↓46
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑7 ↓6**
10	(2)	↑0 ↓0	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑6 ↓5**
25	(2)	**↑1 ↓0**	↑0 ↓1	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑6 ↓5**
50	(2)	↑0 ↓0	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑6 ↓7
100	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓1	↑1 ↓1	↑0 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑4 ↓5
200	(2)	↑0 ↓0	↑0 ↓1	↑0 ↓0	↑1 ↓1	↑0 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑4 ↓5
C	(6)	↑0 ↓0	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓4	↑0 ↓1	↑0 ↓32
S	(6)	**↑2 ↓0**	↑0 ↓1	**↑3 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑33 ↓1**
All	(12)	**↑2 ↓0**	↑0 ↓6	↑3 ↓5	↑6 ↓6	↑4 ↓6	**↑6 ↓5**	**↑6 ↓4**	**↑6 ↓1**	↑33 ↓33
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	↑0 ↓0	↑0 ↓2	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑5 ↓7
10	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑5 ↓8
25	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑5 ↓9
50	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑5 ↓8
100	(2)	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑4 ↓9
200	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑5 ↓11
C	(6)	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓1	↑0 ↓40
S	(6)	↑0 ↓3	↑0 ↓6	↑0 ↓3	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑29 ↓12**
All	(12)	↑0 ↓8	↑0 ↓11	↑0 ↓8	↑5 ↓6	↑6 ↓6	↑6 ↓6	↑6 ↓6	**↑6 ↓1**	↑29 ↓52
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓13
10	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓15
25	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓16
50	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓16
100	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓16
200	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓16
C	(6)	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓46
S	(6)	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓5	↑0 ↓46
All	(12)	↑0 ↓11	↑0 ↓12	↑0 ↓12	↑0 ↓12	↑0 ↓12	↑0 ↓12	↑0 ↓11	↑0 ↓10	↑0 ↓92
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	↑2 ↓2	↑0 ↓9	↑1 ↓7	↑2 ↓7	↑2 ↓7	↑2 ↓5	↑3 ↓5	↑3 ↓4	↑15 ↓46
10	(10)	↑0 ↓4	↑0 ↓8	↑1 ↓7	↑2 ↓6	↑3 ↓6	↑3 ↓6	↑3 ↓4	**↑3 ↓2**	↑15 ↓43
25	(10)	↑3 ↓4	↑0 ↓8	↑0 ↓7	↑3 ↓6	↑3 ↓6	↑3 ↓6	↑2 ↓4	↑3 ↓3	↑16 ↓44
50	(10)	↑2 ↓4	↑0 ↓9	↑1 ↓8	↑2 ↓6	↑3 ↓6	↑3 ↓6	↑3 ↓6	**↑4 ↓3**	↑18 ↓48
100	(10)	↑1 ↓5	↑0 ↓7	↑0 ↓9	↑1 ↓6	↑1 ↓7	↑2 ↓7	↑2 ↓6	↑2 ↓3	↑9 ↓50
200	(10)	↑1 ↓5	↑0 ↓9	↑0 ↓8	↑2 ↓7	↑1 ↓8	↑2 ↓7	↑2 ↓7	↑2 ↓5	↑10 ↓56
C	(30)	↑7 ↓10	↑0 ↓26	↑0 ↓27	↑0 ↓30	↑0 ↓30	↑0 ↓29	↑0 ↓26	↑1 ↓13	↑8 ↓191
S	(30)	↑2 ↓14	↑0 ↓24	↑3 ↓19	**↑12 ↓8**	**↑13 ↓10**	**↑14 ↓8**	**↑15 ↓6**	**↑16 ↓7**	↑75 ↓96
All	(60)	↑9 ↓24	↑0 ↓50	↑3 ↓46	↑12 ↓38	↑13 ↓40	↑14 ↓37	↑15 ↓32	↑17 ↓20	↑83 ↓287

100 000 function evaluations, respectively. The APIs, in terms of the number of peaks, were found to be 2.73%, 19.10%, 13.97%, 12.49%, 11.86% and 12.81% for 5, 10, 25, 50, 100 and 200 peaks respectively. The self-adaptive components thus improved the performance of CDE, in general, over different settings for number of peaks.

This section showed that the self-adaptive components proved beneficial to SACDE on the n_p standard set, but not to SADynPopDE. A performance comparison between DynPopDE and SACDE (given in Appendix D) found that DynPopDE performed better than SACDE on 248 of the 480 environments, and worse on 117 environments. DynPopDE is thus still the best performing algorithm on the n_p standard set, despite the fact that SACDE performed better than CDE.

6.4.9.3 SADynPopDE on the $n_p(t)$ Standard Set

The poor performance of SADynPopDE on the n_p standard set suggests that it would also be ineffective on the $n_p(t)$ standard set. This, however, was found not to be the case. Table 6.11 gives the performance analysis of SADynPopDE compared to DynPopDE on the $n_p(t)$ standard set. SADynPopDE performed better than DynPopDE in 1 100 of the 2 000 environments, and worse in only 267. The cases where DynPopDE performed better than SADynPopDE occurred almost exclusively in 100 dimensional environments.

The average percentage improvement of SADynPopDE over DynPopDE over all experiments was found to be 30.51%. The APIs per dimension were found to be 36.08%, 49.02%, 46.61%, 33.83% -13.01% for 5, 10, 25, 50 and 100 dimensions respectively. SADynPopDE is thus superior except in 100 dimensions. The APIs per change period were found to be 13.81%, 30.87%, 35.84%, 35.60%, 34.29%, 34.63%, 29.41% and 29.61% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The improvement over DynPopDE is uniformly large over all change periods. The APIs per setting of maximum number of peaks are 29.71%, 30.19%, 30.17%, 30.84% and 31.63% for values 5, 10, 25, 50, 100 and 200, respectively. The APIs, with regard to percentage change in the number of peaks, were found to be 18.57%, 25.35%, 32.07%, 36.96% and 39.58% for values of 5%, 10%, 20%, 40% and 80%, respectively. The improvement of SADynPopDE over DynPopDE is thus greater when the percentage change in the number of peaks is higher.

Table 6.10: SACDE vs CDE performance analysis on the n_p standard set

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓14*
10	(2)	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑8 ↓4**
25	(2)	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑5 ↓1**
50	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑5 ↓0**
100	(2)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑7 ↓0**
200	(2)	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑8 ↓1**
C	(6)	**↑4 ↓0**	↑1 ↓1	*↑1 ↓3*	**↑3 ↓2**	**↑3 ↓2**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓2**	**↑23 ↓12**
S	(6)	↑0 ↓0	*↑1 ↓2*	**↑2 ↓1**	↑1 ↓1	**↑2 ↓1**	↑1 ↓1	**↑2 ↓1**	↑1 ↓1	**↑10 ↓8**
All	(12)	**↑4 ↓0**	*↑2 ↓3*	*↑3 ↓4*	**↑4 ↓3**	**↑5 ↓3**	**↑5 ↓2**	**↑6 ↓2**	**↑4 ↓3**	**↑33 ↓20**
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑8 ↓6**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑11 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑9 ↓1**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑9 ↓2**
100	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑7 ↓2**
200	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑8 ↓2**
C	(6)	**↑6 ↓0**	**↑5 ↓0**	↑1 ↓1	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑13 ↓13
S	(6)	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑39 ↓0**
All	(12)	**↑8 ↓0**	**↑11 ↓0**	**↑7 ↓1**	**↑6 ↓5**	**↑6 ↓4**	**↑5 ↓1**	**↑4 ↓1**	**↑5 ↓1**	**↑52 ↓13**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑10 ↓4**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑11 ↓0**
50	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓1**
100	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓3**
200	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓2**
C	(6)	*↑0 ↓3*	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓2*	*↑0 ↓3*	*↑0 ↓1*	**↑2 ↓1**	**↑1 ↓0**	**↑15 ↓10**
S	(6)	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑47 ↓0**
All	(12)	**↑5 ↓3**	**↑12 ↓0**	**↑12 ↓0**	**↑6 ↓2**	**↑6 ↓3**	**↑6 ↓1**	**↑8 ↓1**	**↑7 ↓0**	**↑62 ↓10**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓2**
10	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
25	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓1**
50	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓1**
100	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓2**
200	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓2**
C	(6)	*↑0 ↓6*	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑0 ↓3*	**↑1 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑28 ↓9**
S	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓0**
All	(12)	*↑0 ↓6*	**↑12 ↓0**	**↑12 ↓0**	**↑11 ↓0**	**↑6 ↓3**	**↑7 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑70 ↓9**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑5 ↓8*
10	(2)	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑5 ↓8*
25	(2)	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	**↑1 ↓0**	*↑5 ↓7*
50	(2)	*↑0 ↓2*	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑5 ↓2**
100	(2)	*↑0 ↓2*	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑1 ↓0**	*↑3 ↓6*
200	(2)	*↑0 ↓2*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑4 ↓9*
C	(6)	*↑0 ↓4*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑0 ↓0	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑27 ↓4**
S	(6)	*↑0 ↓6*	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓3*	*↑0 ↓36*
All	(12)	*↑0 ↓10*	↑4 ↓4	**↑6 ↓4**	**↑6 ↓5**	*↑0 ↓4*	*↑0 ↓5*	↑5 ↓5	**↑6 ↓3**	*↑27 ↓40*
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	↑2 ↓2	**↑6 ↓3**	**↑6 ↓4**	*↑4 ↓5*	*↑3 ↓6*	*↑3 ↓5*	*↑4 ↓5*	↑4 ↓4	*↑32 ↓34*
10	(10)	**↑4 ↓2**	**↑8 ↓1**	**↑7 ↓2**	**↑6 ↓2**	**↑4 ↓2**	**↑5 ↓1**	**↑7 ↓1**	**↑8 ↓2**	**↑49 ↓13**
25	(10)	↑3 ↓3	**↑7 ↓1**	**↑8 ↓1**	**↑5 ↓2**	**↑3 ↓1**	**↑4 ↓1**	**↑7 ↓1**	**↑5 ↓0**	**↑42 ↓10**
50	(10)	*↑3 ↓4*	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑41 ↓6**
100	(10)	*↑3 ↓4*	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓3**	**↑5 ↓3**	**↑3 ↓1**	**↑5 ↓1**	**↑5 ↓0**	**↑39 ↓13**
200	(10)	*↑2 ↓4*	**↑7 ↓2**	**↑7 ↓1**	**↑6 ↓2**	↑4 ↓4	**↑4 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑41 ↓16**
C	(30)	*↑10 ↓13*	**↑22 ↓1**	**↑20 ↓4**	**↑14 ↓9**	*↑3 ↓12*	**↑5 ↓3**	**↑16 ↓3**	**↑16 ↓3**	**↑106 ↓48**
S	(30)	**↑7 ↓6**	**↑19 ↓6**	**↑20 ↓5**	**↑19 ↓6**	**↑20 ↓5**	**↑18 ↓6**	**↑18 ↓6**	**↑17 ↓4**	**↑138 ↓44**
All	(60)	*↑17 ↓19*	**↑41 ↓7**	**↑40 ↓9**	**↑33 ↓15**	**↑23 ↓17**	**↑23 ↓9**	**↑34 ↓9**	**↑33 ↓7**	**↑244 ↓92**

Table 6.11: SADynPopDE vs DynPopDE performance analysis on the $n_p(t)$ standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 10	(10)	↑**10** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**4** ↓**0**	↑**4** ↓**1**	↑**3** ↓**1**	↑**3** ↓**0**	↑**3** ↓**0**	↑**43** ↓**2**
25	(10)	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑**4** ↓**1**	↑**3** ↓**0**	↑**50** ↓**2**
50	(10)	↑**9** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**54** ↓**0**
100	(10)	↑**7** ↓**0**	↑**8** ↓**1**	↑**9** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**3** ↓**0**	↑**49** ↓**1**
200	(10)	↑**7** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑**4** ↓**1**	↑**3** ↓**0**	↑**46** ↓**2**
pc 5	(10)	↑**8** ↓**0**	↑**5** ↓**1**	↑**6** ↓**0**	↑0 ↓0	↑**1** ↓**0**	*↑0 ↓2*	*↑0 ↓1*	↑0 ↓0	↑**20** ↓**4**
10	(10)	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**3** ↓**0**	↑1 ↓1	*↑0 ↓1*	↑1 ↓1	↑0 ↓0	↑**25** ↓**3**
20	(10)	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**4** ↓**0**	↑**5** ↓**0**	↑**2** ↓**0**	↑**52** ↓**0**
40	(10)	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**68** ↓**0**
80	(10)	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**77** ↓**0**
C	(25)	↑**25** ↓**0**	↑**24** ↓**0**	↑**25** ↓**0**	↑**17** ↓**0**	↑**15** ↓**1**	↑**14** ↓**3**	↑**13** ↓**2**	↑**12** ↓**0**	↑**145** ↓**6**
S	(25)	↑**18** ↓**0**	↑**18** ↓**1**	↑**17** ↓**0**	↑**11** ↓**0**	↑**13** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**5** ↓**0**	↑**97** ↓**1**
All	(50)	↑**43** ↓**0**	↑**42** ↓**1**	↑**42** ↓**0**	↑**28** ↓**0**	↑**28** ↓**1**	↑**21** ↓**3**	↑**21** ↓**2**	↑**17** ↓**0**	↑**242** ↓**7**
Set.	**Max**	**10 Dimensions**								
n_p 10	(10)	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**73** ↓**0**
25	(10)	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**66** ↓**0**
50	(10)	↑**6** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**58** ↓**0**
100	(10)	↑**9** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**63** ↓**0**
200	(10)	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**55** ↓**0**
pc 5	(10)	↑**10** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**3** ↓**0**	↑**2** ↓**0**	↑**3** ↓**0**	↑**5** ↓**0**	↑**39** ↓**0**
10	(10)	↑**6** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**52** ↓**0**
20	(10)	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**67** ↓**0**
40	(10)	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**79** ↓**0**
80	(10)	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**78** ↓**0**
C	(25)	↑**18** ↓**0**	↑**22** ↓**0**	↑**24** ↓**0**	↑**22** ↓**0**	↑**20** ↓**0**	↑**17** ↓**0**	↑**17** ↓**0**	↑**18** ↓**0**	↑**158** ↓**0**
S	(25)	↑**23** ↓**0**	↑**20** ↓**0**	↑**21** ↓**0**	↑**18** ↓**0**	↑**17** ↓**0**	↑**18** ↓**0**	↑**18** ↓**0**	↑**22** ↓**0**	↑**157** ↓**0**
All	(50)	↑**41** ↓**0**	↑**42** ↓**0**	↑**45** ↓**0**	↑**40** ↓**0**	↑**37** ↓**0**	↑**35** ↓**0**	↑**35** ↓**0**	↑**40** ↓**0**	↑**315** ↓**0**
Set.	**Max**	**25 Dimensions**								
n_p 10	(10)	↑**5** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**70** ↓**0**
25	(10)	↑**5** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**65** ↓**0**
50	(10)	↑**5** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**63** ↓**0**
100	(10)	↑**4** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**58** ↓**0**
200	(10)	↑**4** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**56** ↓**0**
pc 5	(10)	↑**4** ↓**0**	↑**4** ↓**0**	↑**4** ↓**0**	↑**4** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**43** ↓**0**
10	(10)	↑**4** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**54** ↓**0**
20	(10)	↑**5** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**68** ↓**0**
40	(10)	↑**5** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**74** ↓**0**
80	(10)	↑**5** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**73** ↓**0**
C	(25)	↑**1** ↓**0**	↑**16** ↓**0**	↑**21** ↓**0**	↑**18** ↓**0**	↑**19** ↓**0**	↑**19** ↓**0**	↑**12** ↓**0**	↑**20** ↓**0**	↑**126** ↓**0**
S	(25)	↑**22** ↓**0**	↑**21** ↓**0**	↑**21** ↓**0**	↑**22** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**186** ↓**0**
All	(50)	↑**23** ↓**0**	↑**37** ↓**0**	↑**42** ↓**0**	↑**40** ↓**0**	↑**44** ↓**0**	↑**44** ↓**0**	↑**37** ↓**0**	↑**45** ↓**0**	↑**312** ↓**0**
Set.	**Max**	**50 Dimensions**								
n_p 10	(10)	↑**1** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**48** ↓**0**
25	(10)	↑0 ↓0	↑**4** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**7** ↓**1**	↑**9** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**44** ↓**1**
50	(10)	↑0 ↓0	↑**3** ↓**1**	↑**3** ↓**0**	↑**7** ↓**0**	↑**5** ↓**1**	↑**9** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**41** ↓**2**
100	(10)	*↑0 ↓1*	↑**4** ↓**1**	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**41** ↓**4**
200	(10)	↑**1** ↓**0**	↑**3** ↓**1**	↑**4** ↓**0**	↑**6** ↓**1**	↑**4** ↓**0**	↑**7** ↓**1**	↑**8** ↓**0**	↑**9** ↓**0**	↑**42** ↓**3**
pc 5	(10)	↑**1** ↓**0**	↑**3** ↓**1**	↑**2** ↓**0**	↑**4** ↓**1**	↑**5** ↓**2**	↑**6** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**32** ↓**5**
10	(10)	↑0 ↓0	↑**3** ↓**0**	↑**4** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**7** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**33** ↓**1**
20	(10)	↑1 ↓1	↑**3** ↓**1**	↑**5** ↓**1**	↑**10** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**47** ↓**3**
40	(10)	↑0 ↓0	↑**5** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**50** ↓**1**
80	(10)	↑0 ↓0	↑**5** ↓**0**	↑**5** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**54** ↓**0**
C	(25)	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓1*	↑**8** ↓**1**	↑**4** ↓**3**	↑**15** ↓**1**	↑**13** ↓**0**	↑**11** ↓**0**	↑**51** ↓**10**
S	(25)	↑**2** ↓**0**	↑**19** ↓**0**	↑**21** ↓**0**	↑**24** ↓**0**	↑**24** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**165** ↓**0**
All	(50)	↑**2** ↓**1**	↑**19** ↓**3**	↑**21** ↓**1**	↑**32** ↓**1**	↑**28** ↓**3**	↑**40** ↓**1**	↑**38** ↓**0**	↑**36** ↓**0**	↑**216** ↓**10**
Set.	**Max**	**100 Dimensions**								
n_p 10	(10)	*↑0 ↓6*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓7*	*↑0 ↓8*	*↑0 ↓2*	*↑0 ↓63*
25	(10)	*↑0 ↓8*	*↑0 ↓7*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓5*	*↑0 ↓4*	↑**2** ↓**1**	*↑2 ↓55*
50	(10)	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓7*	*↑0 ↓5*	*↑1 ↓4*	*↑1 ↓3*	*↑2 ↓52*
100	(10)	*↑0 ↓5*	*↑0 ↓8*	*↑0 ↓10*	*↑0 ↓6*	*↑0 ↓5*	*↑1 ↓5*	*↑0 ↓3*	↑**3** ↓**2**	*↑4 ↓44*
200	(10)	*↑0 ↓6*	*↑0 ↓8*	*↑0 ↓9*	*↑0 ↓4*	*↑1 ↓2*	*↑2 ↓4*	↑1 ↓1	↑**3** ↓**2**	*↑7 ↓36*
pc 5	(10)	*↑0 ↓7*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓8*	*↑1 ↓9*	*↑0 ↓7*	*↑1 ↓7*	*↑1 ↓5*	*↑3 ↓63*
10	(10)	*↑0 ↓7*	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓2*	*↑0 ↓54*
20	(10)	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓9*	*↑0 ↓8*	*↑0 ↓6*	*↑0 ↓3*	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓43*
40	(10)	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓9*	*↑0 ↓9*	*↑0 ↓5*	*↑1 ↓4*	*↑0 ↓3*	↑**3** ↓**0**	*↑4 ↓42*
80	(10)	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓10*	*↑0 ↓8*	*↑0 ↓8*	*↑2 ↓5*	*↑1 ↓3*	↑**5** ↓**0**	*↑8 ↓48*
C	(25)	*↑0 ↓7*	*↑0 ↓15*	*↑0 ↓23*	*↑0 ↓19*	*↑1 ↓16*	*↑0 ↓19*	*↑0 ↓13*	↑**7** ↓**4**	*↑8 ↓116*
S	(25)	*↑0 ↓25*	*↑0 ↓25*	*↑0 ↓25*	*↑0 ↓21*	*↑0 ↓18*	*↑3 ↓7*	*↑2 ↓7*	*↑2 ↓6*	*↑7 ↓134*
All	(50)	*↑0 ↓32*	*↑0 ↓40*	*↑0 ↓48*	*↑0 ↓40*	*↑1 ↓34*	*↑3 ↓26*	*↑2 ↓20*	*↑9 ↓10*	*↑15 ↓250*
Set.	**Max**	**All Dimensions**								
n_p 10	(50)	↑**25** ↓**6**	↑**33** ↓**10**	↑**33** ↓**10**	↑**32** ↓**10**	↑**29** ↓**11**	↑**30** ↓**8**	↑**26** ↓**8**	↑**26** ↓**2**	↑**234** ↓**65**
25	(50)	↑**25** ↓**8**	↑**30** ↓**7**	↑**33** ↓**10**	↑**28** ↓**10**	↑**29** ↓**11**	↑**28** ↓**6**	↑**26** ↓**5**	↑**28** ↓**1**	↑**227** ↓**58**
50	(50)	↑**20** ↓**7**	↑**28** ↓**8**	↑**28** ↓**9**	↑**28** ↓**10**	↑**28** ↓**8**	↑**30** ↓**5**	↑**27** ↓**4**	↑**29** ↓**3**	↑**218** ↓**54**
100	(50)	↑**20** ↓**6**	↑**26** ↓**10**	↑**29** ↓**11**	↑**26** ↓**6**	↑**27** ↓**6**	↑**27** ↓**5**	↑**28** ↓**3**	↑**32** ↓**2**	↑**215** ↓**49**
200	(50)	↑**19** ↓**6**	↑**23** ↓**9**	↑**27** ↓**9**	↑**26** ↓**5**	↑**25** ↓**2**	↑**28** ↓**6**	↑**26** ↓**2**	↑**32** ↓**2**	↑**206** ↓**41**
pc 5	(50)	↑**23** ↓**7**	↑**18** ↓**12**	↑**18** ↓**10**	↑**12** ↓**9**	↑**17** ↓**11**	↑**15** ↓**10**	↑**14** ↓**8**	↑**20** ↓**5**	↑**137** ↓**72**
10	(50)	↑**17** ↓**7**	↑**24** ↓**9**	↑**27** ↓**10**	↑**21** ↓**7**	↑**18** ↓**8**	↑**19** ↓**8**	↑**17** ↓**7**	↑**21** ↓**2**	↑**164** ↓**58**
20	(50)	↑**21** ↓**7**	↑**31** ↓**8**	↑**35** ↓**10**	↑**35** ↓**8**	↑**31** ↓**6**	↑**30** ↓**3**	↑**26** ↓**1**	↑**25** ↓**3**	↑**234** ↓**46**
40	(50)	↑**25** ↓**5**	↑**34** ↓**8**	↑**35** ↓**9**	↑**35** ↓**9**	↑**36** ↓**5**	↑**37** ↓**4**	↑**36** ↓**3**	↑**37** ↓**0**	↑**275** ↓**43**
80	(50)	↑**23** ↓**7**	↑**33** ↓**7**	↑**35** ↓**10**	↑**37** ↓**8**	↑**36** ↓**8**	↑**42** ↓**5**	↑**40** ↓**3**	↑**44** ↓**0**	↑**290** ↓**48**
C	(125)	↑**44** ↓**8**	↑**62** ↓**18**	↑**70** ↓**24**	↑**65** ↓**20**	↑**59** ↓**20**	↑**65** ↓**23**	↑**55** ↓**15**	↑**68** ↓**4**	↑**488** ↓**132**
S	(125)	↑**65** ↓**25**	↑**78** ↓**26**	↑**80** ↓**25**	↑**75** ↓**21**	↑**79** ↓**18**	↑**78** ↓**7**	↑**78** ↓**7**	↑**79** ↓**6**	↑**612** ↓**135**
All	(250)	↑**109** ↓**33**	↑**140** ↓**44**	↑**150** ↓**49**	↑**140** ↓**41**	↑**138** ↓**38**	↑**143** ↓**30**	↑**133** ↓**22**	↑**147** ↓**10**	↑**1100** ↓**267**

The self-adaptive components also improved CDE's performance on the $n_p(t)$ standard set. A performance analysis that compared SACDE to CDE on the $n_p(t)$ standard set (given in Appendix D) found that SACDE performed statistically significantly better than CDE on 1 403 of the 2 000 experimental environments, and worse on only 244. The average percentage improvement of SACDE over CDE was 36.26%.

The results presented in this section shows that SADynPopDE and SACDE greatly benefitted from their self-adaptive components on problems where the number of optima fluctuates over time. Despite the large improvement of SACDE over CDE, a performance analysis of SADynPopDE compared to SACDE (refer to Appendix D) found that SADynPopDE performed better more often than SACDE, in the 2 000 environments (856 versus 603). SADynPopDE is thus the most effective algorithm on the $n_p(t)$ standard set.

6.4.9.4 Summary for Research Question 8

This research question investigated the effect of the self-adaptive approaches on environments in which the number of optima are unknown or fluctuating. SADynPopDE performed better than DynPopDE on variations of the standard set, but was still inferior to CDE and SACDE.

SADynPopDE performed worse than DynPopDE over a range of values for the number of optima. The self-adaptive components thus did not improve the DynPopDE on problems when the number of optima are unknown. However, large improvements were found for SADynPopDE over DynPopDE, and SACDE over CDE, on problems where the number of optima fluctuates over time. The self-adaptive components were thus especially beneficial on these problems.

6.5 Comparison to Other Approaches

The experimental results of this chapter showed that, in general, the self-adaptive approaches incorporated into SACDE, resulted in an improvement over DynDE and CDE. This section compares SACDE to the algorithms created by other researchers that were used for comparisons in Chapters 4 and 5. Section 6.5.1 compares SACDE to *jDE* on variations of the standard set. Section 6.5.2 compares SACDE to the published results of

several algorithms aimed at DOPs.

6.5.1 SACDE Compared to jDE

jDE is one of the state-of-the-art algorithms aimed at dynamic environments. The comparison of CDE to *jDE* in Section 4.7.1 found that CDE performed better more often than *jDE* on variations of the standard set. SACDE was consequently compared to *jDE* to determine whether it constitutes a further improvement.

The performance analysis of SACDE compared to *jDE* is given in Tables 6.12 and 6.13. SACDE performed better than *jDE* in 1 524 of the 2 160 experimental environments, and worse in only 428 environments. SACDE thus outperformed *jDE* more often than CDE, and was outperformed by *jDE* less often than CDE. The average percentage improvement of CDE over *jDE* was 19.4%, while the API of SACDE over *jDE* was 25.16%. SACDE is thus not only more effective than *jDE*, but also more effective than CDE in comparison to *jDE*.

jDE performed better more often than SACDE when a change period of 100 function evaluations was used, as was the case when comparing to CDE. *jDE* also performed better than SACDE on specific functions, for example, the GDBG function F_3, and the spherical peak function of the MPB in 100 dimensions. Despite these exceptions, the experimental results prove that SACDE is generally a more effective optimisation algorithm for dynamic environments than *jDE*.

6.5.2 SACDE Compared to Other Algorithms

This section compares SACDE to the published results of other algorithms on variations of the Scenario 2 settings of the MPB. The four change detection strategies which were discussed in Section 4.7.2 were incorporated into SACDE to facilitate a fair comparison of SACDE and the other algorithms.

Table 6.14 lists the offline errors of SACDE using the automatic detection strategy (which uses no function evaluations and always works perfectly), the Det_{best} strategy, the Det_{local} strategy, the Det_{n_k-best} strategy, and the $Det_{n_k-local}$ strategy. Cases where SACDE performed statistically significantly worse when using a detection strategy are

Table 6.12: SACDE vs *jDE* performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**1**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
20	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
40	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**0**
80	(2)	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**1**
C	(6)	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**45** ↓**0**
S	(6)	↑*1* ↓*2*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**2**
		GDBG								
F_{1a}	(6)	↑1 ↓1	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**1**
F_{1b}	(6)	↑2 ↓2	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**2**	↑**41** ↓**4**
F_2	(6)	↑*0* ↓*2*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**40** ↓**2**
F_3	(6)	↑**1** ↓**0**	↑**4** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑**22** ↓**18**
F_4	(6)	↑*0* ↓*2*	↑**4** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**39** ↓**3**
F_5	(6)	↑*0* ↓*3*	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**3** ↓**1**	↑**3** ↓**1**	↑3 ↓3	↑**30** ↓**9**
F_6	(6)	↑**3** ↓**1**	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**1**
T_1	(7)	↑**1** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**5** ↓**2**	↑**45** ↓**3**
T_2	(7)	↑**1** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**47** ↓**3**
T_3	(7)	↑**3** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**45** ↓**6**
T_4	(7)	↑*0* ↓*6*	↑**3** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑**41** ↓**10**
T_5	(7)	↑*0* ↓*4*	↑*2* ↓*3*	↑**5** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**5** ↓**2**	↑**5** ↓**1**	↑**5** ↓**2**	↑**35** ↓**12**
T_6	(7)	↑**2** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**45** ↓**4**
All	(54)	↑*11* ↓*13*	↑**45** ↓**3**	↑**52** ↓**0**	↑**54** ↓**0**	↑**52** ↓**0**	↑**45** ↓**6**	↑**44** ↓**7**	↑**43** ↓**11**	↑**346** ↓**40**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**1**
5	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**0**
10	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
20	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**0**
40	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
80	(2)	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**1**
C	(6)	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**1**
S	(6)	↑*1* ↓*3*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**3**
		GDBG								
F_{1a}	(6)	↑**2** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**44** ↓**0**
F_{1b}	(6)	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**2**	↑**40** ↓**3**
F_2	(6)	↑*1* ↓*3*	↑**2** ↓**1**	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**38** ↓**5**
F_3	(6)	↑*1* ↓*2*	↑*2* ↓*3*	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑*1* ↓*2*	↑*0* ↓*6*	↑*0* ↓*6*	↑19 ↓19
F_4	(6)	↑*1* ↓*2*	↑1 ↓1	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**37** ↓**4**
F_5	(6)	↑2 ↓2	↑*1* ↓*4*	↑*2* ↓*3*	↑*2* ↓*3*	↑3 ↓3	↑*2* ↓*3*	↑*2* ↓*3*	↑*2* ↓*3*	↑*16* ↓*24*
F_6	(6)	↑**1** ↓**0**	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**39** ↓**0**
T_1	(7)	↑*0* ↓*1*	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑**45** ↓**5**
T_2	(7)	↑*1* ↓*4*	↑**3** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**43** ↓**7**
T_3	(7)	↑*1* ↓*3*	↑**3** ↓**2**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑**4** ↓**2**	↑**36** ↓**13**
T_4	(7)	↑1 ↓1	↑2 ↓2	↑**5** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**4** ↓**2**	↑**35** ↓**7**
T_5	(7)	↑1 ↓1	↑*2* ↓*3*	↑3 ↓3	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**4** ↓**2**	↑**32** ↓**14**
T_6	(7)	↑**4** ↓**0**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**42** ↓**9**
All	(54)	↑*9* ↓*14*	↑**34** ↓**9**	↑**45** ↓**5**	↑**50** ↓**3**	↑**51** ↓**3**	↑**45** ↓**5**	↑**44** ↓**9**	↑**40** ↓**11**	↑**318** ↓**59**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
5	(2)	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**1**
10	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
20	(2)	↑*0* ↓*1*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**1**
40	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
80	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
C	(6)	↑*0* ↓*6*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**6**
S	(6)	↑*0* ↓*4*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**4**
		GDBG								
F_{1a}	(6)	↑*1* ↓*2*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**41** ↓**2**
F_{1b}	(6)	↑*0* ↓*2*	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**41** ↓**2**
F_2	(6)	↑*0* ↓*5*	↑*1* ↓*2*	↑**4** ↓**2**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**35** ↓**9**
F_3	(6)	↑*2* ↓*4*	↑*0* ↓*5*	↑*0* ↓*5*	↑**5** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑*0* ↓*5*	↑*0* ↓*6*	↑*17* ↓*26*
F_4	(6)	↑*1* ↓*5*	↑*1* ↓*3*	↑**3** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**35** ↓**9**
F_5	(6)	↑**6** ↓**0**	↑**4** ↓**2**	↑**3** ↓**2**	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*13* ↓*34*
F_6	(6)	↑*1* ↓*3*	↑**3** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**38** ↓**3**
T_1	(7)	↑*1* ↓*3*	↑**5** ↓**2**	↑**5** ↓**2**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**39** ↓**14**
T_2	(7)	↑*1* ↓*4*	↑**4** ↓**1**	↑**4** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**37** ↓**13**
T_3	(7)	↑*1* ↓*4*	↑*2* ↓*3*	↑**4** ↓**2**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**35** ↓**16**
T_4	(7)	↑*2* ↓*4*	↑*1* ↓*3*	↑**5** ↓**2**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**5** ↓**2**	↑**35** ↓**17**
T_5	(7)	↑*2* ↓*4*	↑*2* ↓*3*	↑3 ↓3	↑**5** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**4** ↓**2**	↑**32** ↓**17**
T_6	(7)	↑**4** ↓**2**	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**42** ↓**8**
All	(54)	↑*11* ↓*31*	↑**30** ↓**12**	↑**39** ↓**10**	↑**47** ↓**6**	↑**48** ↓**6**	↑**46** ↓**7**	↑**42** ↓**11**	↑**41** ↓**12**	↑**304** ↓**95**

Table 6.13: SACDE vs *jDE* performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓2	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓2**
5	(2)	↑0 ↓2	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓2**
10	(2)	↑0 ↓2	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓2**
20	(2)	↑0 ↓2	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓2**
40	(2)	↑0 ↓2	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓2**
80	(2)	↑0 ↓2	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓2**
C	(6)	↑0 ↓6	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓6**
S	(6)	↑0 ↓6	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓6**
		GDBG								
F_{1a}	(6)	↑1 ↓3	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓3**
F_{1b}	(6)	↑0 ↓4	**↑3 ↓2**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓6**
F_2	(6)	↑0 ↓5	↑1 ↓1	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑34 ↓6**
F_3	(6)	↑1 ↓4	↑0 ↓6	↑0 ↓6	↑1 ↓5	**↑3 ↓0**	**↑4 ↓0**	↑1 ↓2	↑0 ↓6	↑10 ↓29
F_4	(6)	↑0 ↓5	↑2 ↓2	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑35 ↓7**
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	↑3 ↓3	↑1 ↓4	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑22 ↓25
F_6	(6)	↑0 ↓5	↑2 ↓4	**↑3 ↓2**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑34 ↓11**
T_1	(7)	↑1 ↓5	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑42 ↓13**
T_2	(7)	↑1 ↓5	↑2 ↓2	**↑6 ↓1**	**↑5 ↓2**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑36 ↓15**
T_3	(7)	↑1 ↓4	↑1 ↓4	↑2 ↓2	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑32 ↓15**
T_4	(7)	↑2 ↓5	**↑4 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑36 ↓15**
T_5	(7)	↑2 ↓5	↑2 ↓3	**↑5 ↓2**	**↑5 ↓1**	**↑6 ↓0**	**↑6 ↓1**	**↑5 ↓2**	**↑4 ↓2**	**↑35 ↓16**
T_6	(7)	↑1 ↓2	↑2 ↓4	**↑3 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑32 ↓13**
All	(54)	↑8 ↓38	**↑29 ↓15**	**↑39 ↓8**	**↑45 ↓8**	**↑46 ↓4**	**↑46 ↓6**	**↑43 ↓8**	**↑41 ↓12**	**↑297 ↓99**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓9
5	(2)	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓9
10	(2)	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓9
20	(2)	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓9
40	(2)	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓9
80	(2)	↑0 ↓2	↑0 ↓1	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑6 ↓7
C	(6)	↑0 ↓6	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓6**
S	(6)	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓46
		GDBG								
F_{1a}	(6)	↑0 ↓2	**↑2 ↓1**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑36 ↓3**
F_{1b}	(6)	↑0 ↓4	↑2 ↓3	**↑3 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑35 ↓8**
F_2	(6)	↑0 ↓6	↑2 ↓3	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓9**
F_3	(6)	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑1 ↓1	**↑6 ↓0**	**↑5 ↓0**	↑0 ↓4	↑12 ↓28
F_4	(6)	↑0 ↓6	↑1 ↓2	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑35 ↓8**
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓2**	↑3 ↓3	↑1 ↓4	↑0 ↓6	**↑31 ↓15**
F_6	(6)	↑0 ↓6	↑1 ↓4	↑2 ↓2	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑32 ↓12**
T_1	(7)	↑1 ↓5	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑42 ↓13**
T_2	(7)	↑1 ↓4	↑1 ↓2	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑36 ↓11**
T_3	(7)	↑1 ↓4	↑1 ↓5	**↑3 ↓2**	**↑6 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓2**	**↑35 ↓14**
T_4	(7)	↑1 ↓5	**↑3 ↓2**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑35 ↓14**
T_5	(7)	↑1 ↓5	↑2 ↓3	**↑3 ↓2**	**↑6 ↓1**	**↑6 ↓1**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓2**	**↑37 ↓14**
T_6	(7)	↑1 ↓6	↑1 ↓6	↑2 ↓2	**↑6 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑4 ↓1**	**↑33 ↓17**
All	(54)	↑6 ↓41	↑19 ↓25	**↑30 ↓14**	**↑41 ↓11**	**↑41 ↓9**	**↑45 ↓9**	**↑42 ↓10**	**↑35 ↓16**	**↑259 ↓135**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑0 ↓8	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑63 ↓15**
5	(10)	↑1 ↓5	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑64 ↓12**
10	(10)	↑3 ↓6	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑66 ↓13**
20	(10)	↑1 ↓5	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑64 ↓12**
40	(10)	↑0 ↓8	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑63 ↓15**
80	(10)	↑0 ↓8	**↑8 ↓1**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑62 ↓13**
C	(30)	↑3 ↓19	**↑29 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑212 ↓19**
S	(30)	↑2 ↓21	**↑24 ↓6**	**↑24 ↓5**	**↑24 ↓5**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑170 ↓61**
		GDBG								
F_{1a}	(30)	↑5 ↓8	**↑21 ↓1**	**↑27 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑203 ↓9**
F_{1b}	(30)	↑2 ↓13	**↑22 ↓5**	**↑27 ↓1**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑26 ↓4**	**↑196 ↓23**
F_2	(30)	↑1 ↓21	**↑10 ↓7**	**↑24 ↓3**	**↑29 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑184 ↓31**
F_3	(30)	↑5 ↓15	↑6 ↓21	↑8 ↓17	**↑18 ↓11**	**↑22 ↓1**	**↑15 ↓8**	↑6 ↓19	↑0 ↓28	↑80 ↓120
F_4	(30)	↑2 ↓20	↑9 ↓9	**↑20 ↓2**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑181 ↓31**
F_5	(30)	**↑20 ↓5**	**↑22 ↓7**	**↑23 ↓5**	**↑16 ↓12**	↑12 ↓15	↑8 ↓19	↑6 ↓20	↑5 ↓24	**↑112 ↓107**
F_6	(30)	↑5 ↓15	**↑14 ↓8**	**↑22 ↓4**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑25 ↓0**	**↑186 ↓27**
T_1	(35)	↑4 ↓14	**↑31 ↓4**	**↑31 ↓4**	**↑32 ↓3**	**↑32 ↓3**	**↑30 ↓4**	**↑28 ↓6**	**↑25 ↓10**	**↑213 ↓48**
T_2	(35)	↑5 ↓17	**↑17 ↓6**	**↑29 ↓3**	**↑31 ↓4**	**↑32 ↓2**	**↑30 ↓4**	**↑28 ↓6**	**↑27 ↓7**	**↑199 ↓49**
T_3	(35)	↑7 ↓16	↑14 ↓14	**↑21 ↓7**	**↑31 ↓4**	**↑30 ↓2**	**↑30 ↓4**	**↑26 ↓7**	**↑24 ↓10**	**↑183 ↓64**
T_4	(35)	↑6 ↓21	**↑13 ↓8**	**↑27 ↓4**	**↑29 ↓4**	**↑30 ↓3**	**↑27 ↓6**	**↑26 ↓7**	**↑24 ↓10**	**↑182 ↓63**
T_5	(35)	↑6 ↓19	↑10 ↓15	**↑19 ↓10**	**↑29 ↓4**	**↑30 ↓3**	**↑28 ↓5**	**↑27 ↓7**	**↑22 ↓10**	**↑171 ↓73**
T_6	(35)	**↑12 ↓10**	**↑19 ↓11**	**↑24 ↓4**	**↑31 ↓4**	**↑30 ↓3**	**↑28 ↓4**	**↑26 ↓6**	**↑24 ↓9**	**↑194 ↓51**
All	(270)	↑45 ↓137	**↑157 ↓64**	**↑205 ↓37**	**↑237 ↓28**	**↑238 ↓22**	**↑227 ↓33**	**↑215 ↓45**	**↑200 ↓62**	**↑1524 ↓428**

printed in italics. The Det_{local} detection strategy caused SACDE to perform worse in several instances, but the results from the other detection strategies differed from the automatic detection strategy in a minority of cases. SACDE performed significantly better when using the Det_{best} strategy in one instance, which is printed in boldface.

Table 6.14: SACDE using various detection strategies

Settings	Automatic	Det_{best}	p-val	Det_{local}	p-val	Det_{n_k-best}	p-val	$Det_{n_k-local}$	p-val
C_s 1	0.99 ± 0.08	0.97 ± 0.07	0.924	*1.33 ± 0.12*	0.000	1.06 ± 0.07	0.26	1.01 ± 0.08	0.82
C_s 2	1.65 ± 0.1	1.56 ± 0.1	0.197	*2.28 ± 0.14*	0.000	1.61 ± 0.12	0.504	1.6 ± 0.11	0.358
C_s 4	2.51 ± 0.15	*2.76 ± 0.14*	0.043	*3.58 ± 0.19*	0.000	*2.89 ± 0.18*	0.006	*2.79 ± 0.16*	0.016
C_s 6	3.72 ± 0.23	3.81 ± 0.2	0.273	*4.88 ± 0.29*	0.000	3.86 ± 0.27	0.35	3.85 ± 0.23	0.458
Cp 500	7.26 ± 0.38	6.98 ± 0.31	0.415	*8.16 ± 0.47*	0.016	7.71 ± 0.26	0.109	*8.06 ± 0.42*	0.008
Cp 1000	4 ± 0.15	3.94 ± 0.14	0.843	*5.06 ± 0.23*	0.000	*4.35 ± 0.18*	0.003	4.19 ± 0.15	0.063
Cp 2500	1.9 ± 0.08	1.97 ± 0.11	0.293	*2.54 ± 0.13*	0.000	2.03 ± 0.1	0.088	*2.02 ± 0.08*	0.034
Cp 10000	0.45 ± 0.05	0.59 ± 0.16	0.068	*0.79 ± 0.16*	0.000	0.51 ± 0.06	0.155	0.56 ± 0.15	0.374
n_d 10	2.26 ± 0.16	2.27 ± 0.13	0.786	*3.09 ± 0.17*	0.000	2.37 ± 0.21	0.423	2.3 ± 0.17	0.775
n_d 50	12.06 ± 0.77	12.75 ± 0.96	0.213	*16.66 ± 1.34*	0.000	12.55 ± 0.79	0.495	13.09 ± 0.74	0.072
n_d 100	23.81 ± 1.16	*25.84 ± 1.18*	0.015	*35.63 ± 2.05*	0.000	24.99 ± 1.33	0.254	*26.35 ± 1.20*	0.003
n_p 1	4.6 ± 0.47	**3.9 ± 0.44**	0.034	5.03 ± 0.73	0.843	4.39 ± 0.52	0.552	3.97 ± 0.55	0.134
n_p 5	2.1 ± 0.14	2.1 ± 0.17	0.994	*2.60 ± 0.16*	0.000	2.09 ± 0.16	0.73	2.13 ± 0.15	0.775
n_p 20	2.38 ± 0.16	2.42 ± 0.19	0.775	2.59 ± 0.2	0.106	2.18 ± 0.14	0.182	2.17 ± 0.18	0.088
n_p 30	2.84 ± 0.24	2.8 ± 0.14	0.959	3.08 ± 0.19	0.072	2.64 ± 0.15	0.248	2.61 ± 0.16	0.159
n_p 40	2.85 ± 0.19	2.8 ± 0.18	0.495	3.01 ± 0.17	0.273	2.79 ± 0.17	0.504	2.81 ± 0.14	0.775
n_p 50	2.9 ± 0.2	2.96 ± 0.17	0.654	*3.29 ± 0.18*	0.008	2.91 ± 0.19	0.924	2.91 ± 0.28	0.476
n_p 100	2.71 ± 0.15	*2.98 ± 0.17*	0.034	*3.19 ± 0.16*	0.000	*3.03 ± 0.16*	0.006	2.91 ± 0.15	0.106
n_p 200	2.57 ± 0.12	2.54 ± 0.1	0.947	*2.88 ± 0.12*	0.000	2.67 ± 0.12	0.219	2.6 ± 0.13	0.82

Table 6.15 lists the published results of 14 of the algorithms that were discussed in Section 3.3 on the variations of the MPB Scenario 2. The 95% confidence intervals were calculated from the reported standard errors or standard deviations in cases where the confidence interval was not reported. Each result was compared to the relevant SACDE result to determine which is better. Results were considered to be similar if the confidence intervals overlapped, i.e. neither algorithm was considered better than the other. Offline errors that are higher than SACDE's (i.e. SACDE performed better) are printed in boldface in shaded cells. Offline errors that are lower than the corresponding SACDE results are printed in italics.

The comparison of SACDE with each of the tabulated algorithms is briefly discussed below:

MMEO [Moser and Hendtlass, 2007] MMEO algorithm detects changes by re-evaluating all the best solutions in the fitness landscape and is thus comparable to the Det_{local} and $Det_{n_k-local}$ detection strategies. MMEO yielded a lower offline error than SACDE on experiments with change severity of 1, but confidence intervals of the two algorithms overlap (when comparing with $Det_{n_k-local}$). Therefore, MMEO can-

Table 6.15: Reported offline errors of various algorithms on variations of Scenario 2 of the MPB

	MMEO	HJEO	LSEO	CESO	ESCA	MPSO	SPSO
C_s 1	0.66 ± 0.39	*0.25 ± 0.20*	*0.25 ± 0.16*	**1.38 ± 0.04**	**1.53 ± 0.02**	**1.75 ± 0.12**	N/A
C_s 2	*0.86 ± 0.41*	*0.52 ± 0.27*	*0.47 ± 0.24*	**1.78 ± 0.04**	1.57 ± 0.02	**2.4 ± 0.12**	N/A
C_s 4	*0.97 ± 0.41*	*0.64 ± 0.31*	*0.53 ± 0.25*	*2.23 ± 0.10*	1.72 ± 0.06	**3.59 ± 0.20**	N/A
C_s 6	*1.09 ± 0.43*	*0.9 ± 0.33*	*0.77 ± 0.47*	*2.74 ± 0.20*	1.79 ± 0.06	**4.79 ± 0.20**	N/A
n_d 10	2.44 ± 1.51	2.17 ± 1.57	2.25 ± 1.67	**2.51 ± 0.08**	N/A	**4.17 ± 0.29**	N/A
n_d 50	**206.3 ± 69.97**	*5.79 ± 2.74*	*6.22 ± 3.14*	*6.81 ± 0.14*	N/A	N/A	N/A
n_d 100	**480.5 ± 137.39**	16.5 ± 10.86	17.8 ± 13.52	24.6 ± 0.49	N/A	N/A	N/A
n_p 1	11.30 ± 6.98	7.08 ± 3.90	7.47 ± 3.88	*1.04 ± 0.00*	*0.98 ± 0.00*	**5.07 ± 0.33**	N/A
n_p 5	N/A	N/A	N/A	N/A	N/A	*1.81 ± 0.14*	*1.98 ± 0.05*
n_p 20	*0.90 ± 0.31*	*0.39 ± 0.20*	*0.40 ± 0.22*	*1.72 ± 0.04*	*1.89 ± 0.08*	2.42 ± 0.14	N/A
n_p 30	*1.06 ± 0.27*	*0.49 ± 0.18*	*0.49 ± 0.20*	*1.24 ± 0.02*	*1.52 ± 0.04*	2.48 ± 0.14	N/A
n_p 40	*1.18 ± 0.31*	*0.56 ± 0.18*	*0.56 ± 0.18*	*1.30 ± 0.04*	*1.61 ± 0.04*	2.55 ± 0.14	N/A
n_p 50	*1.23 ± 0.22*	*0.58 ± 0.18*	*0.59 ± 0.20*	*1.45 ± 0.02*	*1.67 ± 0.04*	*2.50 ± 0.12*	**3.47 ± 0.06**
n_p 100	*1.38 ± 0.18*	*0.66 ± 0.14*	*0.66 ± 0.14*	*1.28 ± 0.04*	*1.61 ± 0.06*	*2.36 ± 0.08*	**3.60 ± 0.07**
n_p 200	N/A	N/A	N/A	N/A	N/A	*2.26 ± 0.06*	**3.47 ± 0.04**
	CPSO	**MSOD**	**HMSO**	**MSO**	**Cellular DE**	**Cellular PSO**	**MPSO2**
C_s 1	1.06 ± 0.07	1.06 ± 0.03	**1.42 ± 0.04**	**1.51 ± 0.04**	**1.64 ± 0.03**	1.14 ± 0.13	N/A
C_s 2	*1.17 ± 0.06*	1.51 ± 0.04	N/A	N/A	N/A	N/A	N/A
C_s 4	*1.38 ± 0.08*	N/A	N/A	N/A	N/A	N/A	N/A
C_s 6	*1.53 ± 0.08*	N/A	N/A	N/A	N/A	N/A	N/A
Cp 500	N/A	N/A	7.56 ± 0.27	*5.95 ± 0.09*	N/A	*1.44 ± 0.13*	N/A
Cp 1000	N/A	*3.58 ± 0.05*	**4.61 ± 0.07**	3.94 ± 0.08	*3.98 ± 0.03*	*1.33 ± 0.11*	N/A
Cp 2500	N/A	N/A	**2.39 ± 0.16**	N/A	**2.42 ± 0.02**	*1.08 ± 0.09*	N/A
Cp 10^4	0.625 ± N/A	N/A	**0.94 ± 0.09**	**0.97 ± 0.04**	N/A	**1.1 ± 0.18**	N/A
n_p 1	*0.14 ± 0.03*	N/A	*0.87 ± 0.05*	*0.56 ± 0.04*	*1.53 ± 0.07*	**5.23 ± 0.47**	N/A
n_p 5	*0.72 ± 0.08*	N/A	*1.18 ± 0.04*	*1.06 ± 0.06*	*1.50 ± 0.04*	*1.09 ± 0.22*	*1.77 ± 0.05*
n_p 20	*1.59 ± 0.06*	N/A	*1.50 ± 0.06*	*1.89 ± 0.04*	**2.46 ± 0.05**	2.20 ± 0.12	N/A
n_p 30	*1.58 ± 0.05*	N/A	*1.65 ± 0.04*	*2.03 ± 0.06*	2.62 ± 0.05	2.67 ± 0.13	N/A
n_p 40	*1.51 ± 0.03*	N/A	*1.65 ± 0.05*	*2.04 ± 0.06*	2.76 ± 0.05	2.70 ± 0.13	N/A
n_p 50	*1.54 ± 0.03*	N/A	*1.66 ± 0.02*	*2.08 ± 0.02*	2.75 ± 0.05	2.77 ± 0.13	N/A
n_p 100	*1.41 ± 0.02*	N/A	*1.68 ± 0.03*	*2.14 ± 0.02*	2.73 ± 0.03	2.91 ± 0.14	N/A
n_p 200	*1.24 ± 0.02*	N/A	*1.71 ± 0.02*	*2.11 ± 0.03*	2.61 ± 0.02	**3.14 ± 0.12**	*2.37 ± 0.03*

not conclusively be considered superior to SACDE. Change severities of 2, 4 and 6 yielded MMEO results that are clearly superior to SACDE's results. Overlapping confidence intervals were found in 10 dimensions, but SACDE clearly performed better than MMEO in 50 and 100 dimensions. SACDE performed worse than MMEO on experiments with various numbers of peaks, with the exception of the experiment using one peak, where the confidence intervals overlapped.

HJEO [Moser, 2007] HJEO performed better than SACDE on all reported cases, except 10 and 100 dimensions where the confidence intervals overlapped. SACDE performed worse than HJEO on experiments with various numbers of peaks, with the exception of the experiment using one peak, where the confidence intervals overlapped.

LSEO [Moser and Chiong, 2010] LSEO performed better than SACDE on all reported cases, except in 10 and 100 dimensions, where the confidence intervals overlapped. SACDE performed worse than LSEO on experiments with various numbers

of peaks, with the exception of the experiment using one peak, where the confidence intervals overlapped.

CESO [Lung and Dumitrescu, 2007] CESO detects changes by re-evaluating the best individual in the DE population and is thus comparable to the Det_{best} detection strategy. SACDE performed better than CESO on experiments with a change severity of 1 and 2, but CESO performed better than SACDE on experiments with change severities of 4, and 6. SACDE performed better than CESO in 10 dimensions but worse in 50 dimensions. Overlapping confidence intervals were found in 100 dimensions. CESO performed better than SACDE on all settings of number of peaks, except 10.

ESCA [Lung and Dumitrescu, 2010] SACDE performed better than ESCA on a change severity of 1, but worse on change severities of 2, 4 and 6. ESCA performed better than SACDE on all settings of number of peaks, except 10.

MPSO [Blackwell and Branke, 2006] MPSO uses the Det_{local} detection strategy, and is consequently comparable to SACDE using the $Det_{n_k-local}$ detection strategy. SACDE performed better than MPSO on all variations of change severity and dimension. SACDE performed better than MPSO when one peak was used, but worse in experiments with 5, 50, 100 and 200 peaks. Overlapping confidence intervals were found when using 20, 30 and 40 peaks.

SPSO [Li *et al.*, 2006] SPSO detects changes by re-evaluating the five best particles, and is thus comparable to SACDE using the $Det_{n_k-local}$ strategy. SACDE performed worse than SPSO when five peaks were used, but better than SPSO in all other reported cases.

CPSO [Yang and Li, 2010] CPSO detects changes using the Det_{best} detection strategy and is thus roughly comparable to SACDE using the Det_{n_k-best} strategy. Overlapping confidence intervals were found when using a change severity of 1. CPSO performed better than SACPE with change severities of 2, 4 and 6. SACPE performed better than CPSO when a change period of 10 000 was used, but as Yang and Li [2010] did not report the confidence interval for this experiment, SACPE's

superiority cannot be confirmed for this experiment. CPSO performed better than SACDE on all settings of number of peaks, except 10.

MPSOD [Novoa-Hernández *et al.*, 2011] MPSOD is compared to SACDE using the $Det_{n_k-local}$ detection strategy. Overlapping confidence intervals were found for variations of change severity. MPSOD performed better than SACDE when a change period of 1 000 was used.

HMSO [Kamosi *et al.*, 2010a] HMSO uses the Det_{best} change detection strategy, and, as not all sub-populations are evolved per generation, it can be compared to SACDE using Det_{best}. SACDE performed better than HMSO on all variations of change period, with the exception of the experiment where 500 function evaluations were used, where overlapping confidence intervals were found. HMSO performed better than SACDE on all settings of number of peaks, except 10.

MSO [Kamosi *et al.*, 2010b] Kamosi *et al.* [2010b] do not specify the change detection strategy that is used in MSO, but it is assumed that the Det_{best} strategy that was used in HMSO (which was developed by the same authors) is also used in MSO. SACDE performed better than MSO when a change severity of 1 was used, and overlapping confidence intervals were found when a change period of 1 000 was used. SACDE performed better than MSO when a change period of 10 000 was used. MSO performed better than SACDE in all other reported cases.

Cellular DE [Noroozi *et al.*, 2011] Cellular DE employs the Det_{local} change detecting strategy and is thus comparable to SACDE using the $Det_{n_k-local}$ strategy. SACDE performed better than Cellular DE when a change period of 2 500 and 5 000 were used, but worse when a change period of 1 000 was used. Cellular DE performed better than SACDE when 1 and 5 peaks were present, but worse than SACDE when 20 peaks were present. Overlapping confidence intervals were found when more than 20 peaks were present in the environment.

Cellular PSO [Hashemi and Meybodi, 2009a] Cellular PSO uses the Det_{local} change detecting strategy and is thus comparable to SACDE using the $Det_{n_k-local}$ strategy. Cellular PSO performed better than SACDE when using change periods of

500, 1 000 and 2 500, but overlapping confidence intervals were found when using a change period of 5 000. SACDE performed better than Cellular PSO when using a change period of 10 000. SACDE performed better than Cellular PSO when a single peak was present, but worse than Cellular PSO when 5 peaks were present. Overlapping confidence intervals were found for all settings for the number of peaks greater than 5, except when 200 peaks were present, where SACDE performed better than Cellular PSO.

MPSO2 [Blackwell, 2007] MPSO2 detects changes using the Det_{local} strategy, which makes it comparable to SACDE using the $Det_{n_k-local}$ strategy. SACDE performed worse than MPSO2 in both the reported environments.

The analysis in this section found that SACDE performed similar to, or better than the majority of the algorithms on at least one environment. SACDE did not, however, compare as favourably to the algorithms as was the case with CDE in Section 4.7.2 and DynPopDE in Section 5.4. Although CDE and DynPopDE performed better in comparison with published results of the algorithms, it should be noted that the experimental set used in this section is relatively small, and that it has been shown in this chapter that SACDE is a better algorithm than CDE (and by extension, DynPopDE) over a wide range of experimental environments. The default values used by CDE for the scale and crossover factors and the Brownian radius are thus effective for the small set of environments used in this section (the MPB was used by the researchers who originally tuned the default values). CDE thus performed comparatively better SACDE on these environments, while SACDE performed better than CDE on the large set of environments used in the previous sections. The appropriate parameters are thus problem dependant and the default parameters were accordingly less effective over a broad set of environments. SACDE has the further benefit of having fewer parameters to tune than CDE, DynPopDE, and the majority of the algorithms used for comparison in this section.

6.6 Conclusions

This chapter investigated the incorporation of self-adaptive control parameters into CDE and DynPopDE to form SACDE and SADynPopDE. The self-adaptive approaches focused on adapting the scale and crossover factors and the Brownian radius.

Three approaches to self-adapting the scale and crossover factors were selected from the literature. Alterations of these approaches were investigated, including the use of different DE schemes, random initial values and resetting the values when a change in the environment occurs. A comparison of these approaches found that, in conjunction with CDE, the approach of Brest *et al.* [2009] performed the best, but when using the DE/best/2/bin scheme with initial values of the scale and crossover factors selected from a normal distribution. This algorithm is referred to as jSA2Ran.

A comparative performance analysis of jSA2Ran and CDE found that jSA2Ran performed better more often than CDE over a wide range of benchmark instances. The effectiveness of jSA2Ran was found to be function-dependent, and more pronounced at high change periods.

A novel approach to self-adapting the Brownian radius was proposed in this chapter. The use of different random distributions and resetting the radius when changes occur, was investigated. An experimental comparison of these approaches found that the most effective approach uses a normal distribution to select the Brownian radius, and resets the radius to initial values when changes occur. This algorithm is referred to as SABrNorRes.

SABrNorRes was compared to CDE on variations of the standard set. The results showed that SABrNorRes performed better more often than CDE on the benchmark instances. Improvements were more pronounced at high change periods and were found to be function-dependent. A weakness of the SABrNorRes is that it performed comparatively poorly on 100 dimensional problems.

SABrNorRes and jSA2Ran were combined to form SACDE. A performance analysis which compared the performance of SACDE to CDE found that SACDE performed better more often than CDE, but not more often than SABrNorRes performed better than CDE. A scalability study found that SACDE generally scaled similar to SABrNorRes, while jSA2Ran generally scaled similar to CDE.

The scalability study confirmed that the self-adaptive algorithms scale better to high change periods than CDE when compared to DynDE. SACDE and SABrNorRes did not scale well to high dimensions, especially on the MPB functions. This trend was found to be function-dependent which caused exceptions to the general rule. SACDE and SABrNorRes scaled well with respect to the change severity, as the adaptive Brownian radius was capable of maintaining problem-appropriate diversity. A strong correlation was observed between the performance of the self-adaptive algorithms and the underlying functions and change types.

The convergence profiles of the algorithms showed that the self-adaptive algorithms generally show more rapid improvements in errors after a change in the environment. SACDE and SABrNorRes reached equally low errors as DynDE at high change periods, which explains why SACDE and SABrNorRes performed better than CDE on these environments. The diversity profiles showed that, while jSA2Ran mirrored the low diversity values found for CDE. SACDE and SABrNorRes maintained a higher average sub-population diversity over the course of the optimisation process. The effect of resetting the Brownian radius after changes in the environment was a temporary increase in overall and average sub-populations diversity.

The self-adaptive components that were used to create SACDE were incorporated into DynPopDE to form SADynPopDE. An experimental comparison found that the self-adaptive components do not compensate for DynPopDE's poor scalability with respect to the underlying function. SADynPopDE performed worse than DynPopDE over various settings of the number of peaks, which suggests that the self-adaptive components are not beneficial to the DynPopDE algorithm. However, it was shown that SADynPopDE was superior to DynPopDE in situations where the number of optima in the dynamic environment is fluctuating. Large improvements were found in several experiments, although diminished performance was observed in 100 dimensions. The self-adaptive components thus improve DynPopDE on problems where the number of optima fluctuates.

A comparison of SACDE with other algorithms from the literature found that SACDE compares favourably to the other algorithms. SACDE did not outperform as many algorithms as CDE on the benchmark results reported by other researchers, despite the fact that this chapter has shown that SACDE, in general, is a more effective algorithm than

CDE.

The experimental results presented in this chapter showed that CDE was improved by the incorporation of self-adaptive control parameters which formed SACDE. SACDE has three fewer parameters to tune than CDE, and yields lower offline errors in the majority of the benchmark instances that were investigated. DynPopDE does not benefit from the incorporation of self-adaptive control parameters on problems where the number of optima is unknown, but SADynPopDE yielded lower offline error on problems where the number of optima fluctuates.

Chapter 7

CONCLUSIONS

This thesis reviewed algorithms aimed at solving dynamic optimisation problems. New DE based algorithms for dynamic environments, which were shown to be improvements over previous DE-based algorithms, were presented. The principal findings of this thesis are summarised in this chapter. Section 7.1 gives an overview of the core content of each of the chapters. The primary objective of this study was to improve on current DE-based algorithms for optimising dynamic environments. Section 7.2 explains how the primary and the sub-objectives were achieved. Future work arising from this study is described in Section 7.3.

7.1 Summary

Chapter 2 briefly described the concepts of optimisation and optimisation algorithms. Genotypic diversity and measures of diversity were outlined. The chapter reviewed DE and approaches to self-adapting control parameters. Dynamic optimisation environments were described and it was concluded that the three major considerations when classifying dynamic environments are: the fitness landscape composition, the change types, and the change pervasiveness (refer to Section 2.5.2). The chapter argued that the three factors that influence the ability of an optimisation algorithm to optimise a dynamic environment are: the hardness of the fitness landscape, the frequency of changes, and the severity of changes.

The moving peaks benchmark (MPB) and the generalised dynamic benchmark gen-

erator (GDBG) were critically discussed. An extension to the MPB, which allows the investigation of environments in which the number of optima fluctuates over time, was presented. The discussion concluded that both benchmarks are appropriate for investigating the effect of varying the change period and number of dimensions. The MPB is best suited to investigating the effect of varying the change severity and number of optima, while the GDBG is best suited to investigating the effect of different change types and underlying functions. Chapter 2 concluded with a description of performance measures for dynamic environments. The offline error was identified as the most appropriate performance measure for use in this thesis.

Chapter 3 reviewed algorithms aimed at optimisation in dynamic environments. The two main problems associated with using DE in dynamic environments, namely loss of diversity and outdated information, were described. A literature survey of related work identified three broad categories of approaches that can be used to categorise strategies for adapting algorithms for dynamic optimisation: increasing diversity, using memory, and employing parallel searching. Seven specific strategies were identified that are commonly used in dynamic optimisation algorithms. These strategies are: change control parameters, custom individuals, refresh information, sentinels, selection, history, and multiple populations. Algorithms aimed at solving dynamic optimisation problems were discussed in terms of their base algorithms: GA, PSO, DE, and other algorithms. The strategies that are used by each of the algorithms were indicated. Two DE-based algorithms, DynDE and *jDE* were discussed in detail. Section 3.4.1 showed that the subcomponents of DynDE allow it to maintain much higher diversity than the normal DE algorithm.

Chapters 4 to 6 presented new algorithms. These are tabulated in Table 7.1, along with their respective advantages and disadvantages to aid the discussion in the rest of this section.

Chapter 4 commenced with a description of the exclusion threshold approach that is used in this thesis in the absence of knowledge of the number of optima in the environment. The appropriate number of individuals per sub-population and the appropriate number of Brownian individuals were experimentally determined for DynDE. The results of the experiments showed that using a small sub-population size gives better results in the majority of investigated cases. While this result may seem counterintuitive, it can be

Table 7.1: Summary of novel algorithms

Description	Advantages	Disadvantages
CPE is aimed at lowering the error faster after changes occur. Only the best performing sub-population is evolved at a time.	Performed better than DynDE in the majority of experimental environments.	Did not perform better than DynDE at high change period in low dimensions.
RMC evaluates the midpoint value between the best individuals in sub-populations flagged for exclusion. Sub-populations are not reinitialised if they are located on separate optima.	Better results than those of DynDE were achieved in low dimensions.	RMC generally had a very small impact on the algorithm's performance.
CDE is the combination of CPE and RMC.	Performed better than DynDE in the majority of experimental environments. CDE also resulted in significant improvements over its predecessor algorithms. The general applicability of CDE was illustrated by the incorporation of its sub-components into *jDE*.	Did not perform better than DynDE at high change period in low dimensions.
DynPopDE is an extension of CDE which adapts the number of sub-populations during the optimisation process. DynPopDE is aimed at situations where the number of optima is unknown or fluctuating.	DynPopDE performed better than DynDE and CDE on MPB instances with various settings for the number of optima. Improvements were also found when the number of optima fluctuates.	DynPopDE did not scale well to other functions, and was generally inferior to CDE.
jSA2Ran is an adaptation of the approach of Brest *et al.* [2009] to self-adapt the scale and crossover factors.	jSA2Ran generally performed better than CDE over a wide range of benchmark instances. jSA2Ran has fewer parameters to tune than CDE.	The magnitude of improvements over CDE was relatively small.
SABrNorRes self-adapts the Brownian radius.	SABrNorRes performed better more often on the experimental environments than CDE. Large improvements were found at high change periods and low dimensions. SABrNorRes has fewer parameters to tune than CDE.	SABrNorRes proved to be inferior to CDE in 100 dimensions.
SACDE is the combination of jSA2Ran and SABrNorRes.	SACDE generally performed better than CDE over a wide range of benchmark instances. SACDE has fewer parameters to set than its predecessor algorithms.	SACDE did not perform better than CDE more often than SABrNorRes. SACDE was inferior to CDE in 100 dimensions.
SADynPopDE was created by incorporating the self-adaptive components of SACDE into DynPopDE.	SADynPopDE performed better than DynPopDE on environments where the number of optima is fluctuating. DynPopDE has fewer parameters to set than its predecessor algorithms.	SADynPopDE was inferior to DynPopDE on various settings of the number of optima on the MPB. SADynPopDE did not scale well to other functions, and was generally inferior to SACDE.

explained by the fact that smaller sub-population sizes allow the optimisation algorithms to perform more generations between changes in the environment. The benefit of greater diversity that can be achieved by using larger sub-population sizes is thus offset by the loss of generations that could have been used to converge to optima.

DynDE was extended by incorporating two novel approaches to form the competi-

tive population evaluation (CPE) and reinitialisation midpoint check (RMC) algorithms. CPE utilises a custom performance value to evolve selectively only one sub-population at a time. This results in optima being discovered in sequence rather than in parallel. The global optimum is thus discovered earlier, which reduces the average error of the algorithm. DynDE employs exclusion, a technique which reinitialises a sub-population when it converges to an optimum occupied by another sub-population. This dual convergence to a single optimum is detected by determining whether the best individuals of each sub-population are located within a threshold distance of each other. A disadvantage of the exclusion approach is that some optima are at times located within the exclusion threshold of another, and sub-populations should consequently not be reinitialised. This weakness is partially remedied by RMC's checking if a valley exists between the best individuals in each sub-population. CPE and RMC are combined to form competitive differential evolution (CDE).

CPE, RMC and CDE were evaluated on a wide range of benchmark instances in Chapter 4. Scalability studies showed that the offline errors of DynDE, CPE, RMC and CDE increase when changes in the environment become more frequent or more severe, and when the number of dimensions is increased. It was also shown that the peak function can have a considerable influence on the performance of dynamic optimisation algorithms.

Experimental evidence was presented that shows that RMC yields minor improvements, localised in low dimensional problems, over DynDE. This observation was experimentally explained by showing that situations where optima are located within the exclusion threshold of each other are much more likely to occur in low dimensional rather than in high dimensional cases.

CPE was found to perform better than DynDE in a wide range of experiments, but especially in high dimensions. No benefit of using CPE over DynDE was found on problems where changes occur very infrequently. This result is expected since CPE improves DynDE by locating better optima earlier, after changes in the environment. In the absence of frequent changes in the environment, the benefit of locating better optima early is reduced.

CDE was shown to be a slight improvement over its constituent parts, CPE and RMC, but was found to behave very similar to CPE in all cases when high numbers of dimensions were used. An analysis of the convergence profiles of DynDE and CDE found that CDE's

current error reduces faster than that of DynDE after changes in the environment. The diversity profile of CDE was found to be very similar to that of DynDE, but the average sub-population diversity of CDE was found to decrease at a faster rate than that of DynDE at the commencement of the optimisation process.

CDE and DynDE were compared in terms of average lowest error found just before changes in the environment to illustrate that CDE does not merely exploit the offline error performance measure. The experimental results showed that CDE performed better more often, in general, than DynDE on the benchmark environments. The improvements of CDE over DynDE were more pronounced at low change periods and high dimensions.

CDE was shown to compare favourably to the reported results on the MPB of other state-of-the-art algorithms. The general applicability of the competitive population evaluation and reinitialisation midpoint check approaches were illustrated by their incorporation into *jDE*. The adapted *jDE* algorithm was shown to outperform the normal *jDE* algorithm in the majority of the experimental environments.

Chapter 5 presented and motivated the dynamic population DE (DynPopDE) which was created by adapting CDE for dynamic problems in which the number of optima is unknown or fluctuates over time. DynPopDE dynamically introduces a new sub-population when stagnation occurs for current sub-populations, and removes sub-populations that are not converging to new optima in the environment.

Experiments with DynDE and CDE on problems with large numbers of optima showed that it is not an effective strategy to use the same number of sub-populations as the number of optima. Considerably better results were found when a constant number of 10 sub-populations was used in these algorithms.

A scalability study found that increasing the number of optima in a dynamic environment generally results in higher offline errors for DynDE, CDE and DynPopDE. CDE was found to perform considerably better than DynDE on problems where the number of optima is unknown. DynPopDE in turn performed better than CDE when larger numbers of optima are present. However, the improvement was more pronounced in low dimensional cases. An analysis of DynPopDE's convergence profiles concluded that the reason for DynPopDE's sub-optimal performance in high dimensions is that the population spawning and removal processes are ineffective in high dimensions. This causes DynPopDE to

be unable to distinguish between environments with large numbers of optima and environments with small numbers of optima. DynPopDE thus creates either too many or too few sub-populations in high dimensions. DynPopDE was shown to compare favourably with the reported results of other algorithms focusing on unknown numbers of optima.

Dynamic optimisation problems where the number of optima fluctuate over time have not been widely investigated by other researchers. To address this gap in the research, DynDE, CDE and DynPopDE were studied using an extension of the MPB that simulates these problems. Comparisons between DynDE, CDE and DynPopDE on problems where the number of optima fluctuates over time showed that CDE performed considerably better than DynDE. DynPopDE was found to be marginally more effective than CDE, but considerable improvements of DynPopDE over CDE were found when low change periods were used.

DynPopDE's process of spawning and removing sub-populations was validated by incorporating the spawning and removal process used in MPSO2 into CDE. The performance of the resulting algorithm, MPSO2CDE, was compared to that of DynPopDE. DynPopDE performed better than MPSO2CDE in the majority of the environments that were investigated.

A major weakness of DynPopDE was identified when evaluating its performance using various underlying functions. DynPopDE proved to be inferior to CDE on the majority of the functions of the GDBG. The GDBG benchmark instances all have different numbers of optima. If DynPopDE was truly effective at adjusting the number of sub-populations to suit the environment, it should have performed better than CDE on the GDBG environments. The set of environments on which DynPopDE can be used effectively is thus severely limited due to the strong function-dependence of DynPopDE.

Chapter 6 investigated the incorporation of self-adaptive control parameters into CDE. Strategies were considered to self-adapt the scale and crossover factors, as well as the Brownian radius. Three existing approaches were considered to self-adapt the scale and crossover factors: SDE of Omran *et al.* [2005a], Barebones DE of Omran *et al.* [2009], and the approach of Brest *et al.* [2006][Brest *et al.*, 2009] which was also used in the successful *jDE* algorithm. Since CDE uses DE/best/2/bin while both the approaches of Omran *et al.* [2005a] and Brest *et al.* [2006] use DE/rand/1/bin, these two approaches

were also investigated using DE/best/2/bin. Further alterations that were investigated included resetting the scale and crossover factors to their initial values when a change in the environment occurs, and selecting the initial values from a normal distribution.

An experimental comparison found that the approach of Brest *et al.* [2009], but in conjunction with DE/best/2/bin and with initial values selected from a normal distribution, is the most effective algorithm for incorporation into CDE. This algorithm was referred to as jSA2Ran. A comparison to CDE showed that jSA2Ran performed better on the experimental environments more often than CDE, and that improvements were more pronounced at high change periods.

Four approaches were investigated for self-adapting the Brownian radius. All the approaches involved randomly selecting the Brownian radius from a distribution controlled by the average of a history of successful Brownian radius values. The initial value which controlled the Brownian radius was set to the radius of the first sub-population that was evaluated. Two distributions were tested, i.e. Gaussian and Cauchy. Additionally, the effect of resetting the Brownian radius after changes in the environment, was investigated. An experimental comparison found that SABrNorRes, the approach that utilises a normal distribution with resetting after changes, performed the best. SABrNorRes was experimentally compared to CDE. The results showed that SABrNorRes generally performed better than CDE, especially at high change periods. SABrNorRes performed better than DynDE on environments with a high change period where DynDE generally performed better than CDE. However, it was found that SABrNorRes was inferior to CDE in the majority of 100 dimensional experiments.

SABrNorRes and jSA2Ran were combined to from SACDE. SACDE performed better more often on the benchmark environments than CDE. A scalability study found that SACDE and SABrNorRes generally exhibited very similar scaling behaviour. Despite this fact, SABrNorRes performed better than CDE more often than SACDE. SACDE does, however, have the advantage of having fewer parameters than SABrNorRes.

An investigation into the convergence profiles of the new algorithms found that the reason why the self-adaptive algorithms performed better than CDE at high change periods is that the current errors of the self-adaptive algorithms reached lower values than those of CDE. The self-adaptive algorithms thus not only have the advantage of a fast reduction

in current error after a change in the environment, but also in the low current error value that is achieved.

The diversity profiles of the self-adaptive algorithms showed that jSA2Ran's diversity did not meaningfully differ from that of CDE, but that SABrNorRes and SACDE exhibited different behaviour. The overall diversity and average sub-population diversity of SABrNorRes and SACDE sharply increased after changes in the environment. The average sub-population diversity of SABrNorRes and SACDE was generally higher than that of CDE.

Incorporating the self-adaptive approach used in SACDE into DynPopDE to form SADynPopDE did not result in a more effective algorithm on problems where the number of optima is unknown, although SADynPopDE did perform slightly better than DynPopDE over a range of functions. Large improvements of SADynPopDE over DynPopDE were found on environments where the number of optima fluctuates over time. SACDE also showed considerable improvements over CDE on these environments, but was inferior to SADynPopDE. A comparison to other algorithms found that SACDE compares favourably to state-of-the-art algorithms aimed at dynamic environments.

7.2 Achievement of Objectives

The objectives of this study were outlined in Section 1.2. The primary objective of this thesis was to create improved DE-based algorithms for dynamic environments. Eight new algorithms were created and were shown to yield better results than existing DE-based algorithms. The sub-objectives were achieved as follows:

- Existing algorithms aimed at solving dynamic optimisation problems, specifically focusing on algorithms based on differential evolution, were identified and reviewed.
- Three broad categories of approaches to solve dynamic optimisation problems were found in the literature. Seven general strategies used to achieve the three approaches were identified.
- DynDE was extended to form CDE and DynPopDE, and hybridised with *jDE* to form SACDE and SADynPopDE.

- Dynamic optimisation problem environments that have not been thoroughly investigated by other researchers, namely environments in which the number of optima fluctuate over time, were investigated. Scalability studies were also conducted using larger experimental sets than those generally used by other researchers.

- The use of adaptive control parameters to reduce the number of parameters that must be manually tuned was investigated. The number of sub-populations is adapted in DynPopDE, while SACDE self-adapts the DE scale and crossover factors and the Brownian radius.

- Extensive scalability studies were performed on existing and newly created algorithms.

- The performances of the new algorithms were compared to the reported results of other algorithms.

All the objectives of this study were achieved.

7.3 Future Work

The components that make up CDE, namely competing populations and reinitialisation midpoint check, do not depend on any DE specific behaviour. Future studies could investigate the general applicability of these components on PSO and GA-based algorithms for dynamic environments. Section 3.3 highlighted the many similarities of DE, PSO and GA-based approaches to solving DOPs, for example, the use of multiple populations or swarms. Hybrid algorithms can be formed by incorporating facets of the approaches presented in this thesis into existing PSO and GA-based algorithms. The resulting hybrid algorithms may be an improvement on current algorithms.

The approach to spawning and removing sub-populations, which was introduced in Chapter 5, was found to be beneficial on specific environments, but did not scale well to different functions. This limits the current applicability of DynPopDE. Future work can include the improvement of DynPopDE to extend its usefulness to more environments. DynPopDE does have the advantage of not having to tune the number of sub-populations

parameter. Research that results in an improved technique of spawning and removing sub-populations may enable DynPopDE to be as effective as CDE over a wide range of problems, while still retaining DynPopDE's advantages.

jDE is currently one of the state-of-the-art DE algorithms aimed at dynamic environments. SACDE was shown generally to outperform *jDE* on a wide range of dynamic environments. However *jDE* did perform better than SACDE on several benchmark instances. The self-adaptive scale and crossover component of *jDE* was incorporated into SACDE and showed improved results. Future work could include further hybridisations of the two algorithms which could result in further improvements. For example, the aging metaphor used by *jDE*, which results in increased diversity, could potentially improve the performance of SACDE on DOPs with large change periods by preventing the algorithm from stagnating.

The scope of this thesis was delineated to exclude environments in which the number of dimensions changes. The algorithms developed in this study are consequently not suited to environments with a dynamic number of dimensions. Future work could include the application of DE to problems with dynamically changing constraints. The resulting algorithms would be more generally applicable than CDE, DynPopDE, SACDE and SADynPopDE.

Chapter 8

Bibliography

H. A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 831–836. IEEE, 2002.

H. A. Abbass and R. Sarker. The Pareto Differential Evolution Algorithm. *International Journal on Artificial Intelligence Tools*, 11(4):531–552, World Scientific, 2002.

M. M. Ali and A. Törn. Population set-based global optimization algorithms: some modifications and numerical studies. *Computers and Operations Research*, 31(10):1703–1725, Elsevier, September 2004. ISSN 0305-0548.

P. J. Angeline. Tracking extrema in dynamic environments. In *Proceedings of the Conference on Evolutionary Programming*, pages 335–345. Springer, 1997.

P. J. Angeline, D. B. Fogel, and L. J. Fogel. A comparison of self-adaptation methods for finite state machines in dynamic environments. In *Proceedings of the Conference on Evolutionary Programming*, pages 441–449. Springer, 1996.

D. V. Arnold and H.-G. Beyer. Random Dynamics Optimum Tracking with Evolution Strategies. In *Parallel Problem Solving from Nature*, pages 3–12. Springer, 2002.

D. Ayvaz, H. Topcuoglu, and F. Gurgen. Performance evaluation of evolutionary heuristics in dynamic environments. *Applied Intelligence*, pages 1–15, Springer, 2011.

T. Bäck. On the behavior of evolutionary algorithms in dynamic environments. In *Pro-*

ceedings of the IEEE International Conference on Evolutionary Computation, pages 446–451. IEEE, 1998.

T. Bäck, U. Hammel, and H. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, IEEE, 1997.

N.A. Barricelli. Symbiogenetic evolution processes realized by artificial methods. *Methodos*, 9:143–182, 1957.

H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. M. Gambardella. Results of the first international contest on evolutionary optimisation. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 611–615. IEEE, 1996.

S. Bird and X. Li. Informative performance metrics for dynamic optimisation problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 18–25. ACM, 2007.

T. Blackwell. Particle swarm optimization in dynamic environments. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 29–49. Springer, 2007. ISBN 978-3-540-49772-1.

T. Blackwell and J. Branke. Multiswarm optimization in dynamic environments. *Applications of Evolutionary Computing*, 3005:489–500, Springer, 2004.

T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, IEEE, 2006.

T. Blackwell, J. Branke, and X. Li. Particle swarms for dynamic optimization problems. In *Swarm Intelligence*, Natural Computing Series, pages 193–217. Springer, 2008. ISBN 978-3-540-74089-6.

T. M. Blackwell and P. J. Bentley. Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 19–26. Morgan Kaufmann, 2002.

S. Boettcher and A. G. Percus. Extremal optimization: Methods derived from co-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 825–832, Orlando, Florida, USA, 1999. Morgan Kaufmann.

J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002. ISBN 0-7923-7631-5.

J. Branke. *The moving peaks benchmark*. http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/, Accessed February 2007.

J. Branke, T. Kaußler, C. Schmidt, and H. Schmeck. A multi-population approach to dynamic optimization problems. In *Adaptive Computing in Design and Manufacturing*, pages 299–308. Springer, 2000.

J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. In *Advances in evolutionary computing: theory and applications*, pages 239–262. Springer, 2003. ISBN 3-540-43330-9.

J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. S. Maucec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing*, 11(7): 617–629, Springer, 2007.

J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. S. Maucec. High-dimensional real-parameter optimization using Self-Adaptive Differential Evolution algorithm with population size reduction. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2032–2039. IEEE, 2008.

J. Brest, S. Greiner, B. Boškovic, M. Mernik, and V. Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006. ISSN 1089-778X.

J. Brest, A. Zamuda, B. Boškovic, M. S. Maučec, and V. Žumer. Dynamic optimization using self-adaptive differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 415–422. IEEE, 2009.

R.L. Burden and J.D. Faires. *Numerical Analysis*, chapter 10, pages 628–633. Brooks/Cole, 7th edition, 2001. ISBN 0-534-38216-9.

E.K. Burke, S. Gustafson, and G. Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, IEEE, 2004.

A. Carlisle and G. Dozier. Adapting particle swarm optimisation to dynamic environments. In *Proceedings of the International Conference on Artificial Intelligence*, pages 429–434, 2000.

A. Carlisle and G. Dozier. Tracking changing extrema with adaptive particle swarm optimizer. In *Proceedings of the IEEE World Automation Congress*, pages 265–270. IEEE, 2002.

C.S. Chang and D. Du. Differential evolution based tuning of fuzzy automatic train operation for mass rapid transit system. In *Proceedings of Electric Power Applications*, pages 206–212. IEEE, 2000.

C.W. Chen, D.Z. Chen, and G.Z. Cao. An improved differential evolution algorithm in training and encoding prior knowledge into feedforward networks with application in chemistry. *Chemometrics and Intelligent Laboratory Systems*, 64(1):2743, Elsevier, 2002.

J.P. Chiou and F.S. Wang. A hybrid method of differential evolution with application to optimal control problems of a bioprocess system. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 627–632. IEEE, 1998.

M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6 (1):58–73, IEEE, 2002.

H. G. Cobb. *An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments.* Navy Center for Applied Research in Artificial Intelligence, Technical Report AIC-90-001, 1990.

C. Cruz, J.R. Gonzàlez, and D.A. Pelta. Optimization in dynamic environments: A survey on problems, methods and measures. *Soft Computing*, 15(7):1427–1448, Springer, 2011.

I.L. López Cruz, L.G. van Willigenburg, and G. van Straten. Efficient differential evolution algorithms for multimodal optimal control problems. *Applied Soft Computing*, 3(2):97–122, Springer, 2003.

C. Darwin. *The origin of species*. 1859.

S. Das, A. Konar, and U.K. Chakraborty. Two improved differential evolution schemes for faster global search. In *Genetic and Evolutionary Computation Conference, GECCO*, pages 991–998. ACM, 2005. ISBN 1-59593-010-8.

S. Das and P.N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4 –31, IEEE, 2011. ISSN 1089-778X.

F. O. De França and F. J. Von Zuben. A dynamic artificial immune algorithm applied to challenging benchmarking problems. In *Proceedigs of the IEEE Congress on Evolutionary Computation*, pages 423–430. IEEE, 2009.

K.A. De Jong. Evolving in a changing world. In *Foundations of Intelligent Systems, 11th International Symposium*, volume 1609 of *Lecture Notes in Computer Science*, pages 512–519. Springer, 1999. ISBN 3-540-65965-X.

I. G. del Amo, D. A. Pelta, and J. R. González. Using heuristic rules to enhance a multiswarm PSO for dynamic environments. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

M.C. du Plessis and A.P. Engelbrecht. Improved differential evolution for dynamic optimization problems. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 229–234, IEEE, 2008.

M.C. du Plessis and A.P. Engelbrecht. Differential evolution for dynamic environments with unknown numbers of optima. *Journal of Global Optimization*, pages 1–27, Springer, 2012a. ISSN 0925-5001.

M.C. du Plessis and A.P. Engelbrecht. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research*, 218(1):7–20, Elsevier, 2012b.

A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3:124–141, IEEE, 2000.

A. P. Engelbrecht. Particle swarm optimization: Where does it belong? In *IEEE Swarm Intelligence Symposium*, pages 48–54. IEEE, 2006.

A. P. Engelbrecht. *Computational Intelligence An Introduction*. John Wiley and Sons, 2nd edition, 2007. ISBN 978-0-470-03561-0.

R. Feynman, R. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume 1. Addison-Wesley, Boston, second edition, 1963.

L.J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962. ISSN 1432-7643.

A.S. Fraser. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 10:484–491, CSIRO Publishing, 1957.

T.B. Gibbon, L. Wu, D.W. Waswa, A.B. Conibear, and A.W.R. Leitch. Polarization mode dispersion compensation for the south african optical-fibre telecommunication network. *South African Journal of Science*, 104:119, 2008.

D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the International Conference on Genetic algorithms*, pages 41–49. L. Erlbaum Associates Inc., 1987.

J. J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 2031–2038. IEEE, 1999.

A.B. Hashemi and M.R. Meybodi. Cellular PSO: A PSO for Dynamic Environments. In *Advances in Computation and Intelligence*, Lecture Notes in Computer Science, pages 422–433. Springer, 2009a.

A.B. Hashemi and M.R. Meybodi. A multi-role cellular PSO for dynamic environments. In *CSI Computer Conference*, pages 412–417, 2009b.

J. He, C. Reeves, C. Witt, and X. Yao. A Note on Problem Difficulty Measures in Black-Box Optimization: Classification, Realizations and Predictability. *Evolutionary Computation*, 15(4):435–443, MIT Press, 2007.

T. Hendtlass. A combined swarm differential evolution algorithm for optimization problems. In *Proceedings of the International conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 11–18. Springer, 2001.

J. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.

R. Hooke and T. A. Jeeves. " Direct Search" solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229, ACM, 1961. ISSN 0004-5411.

X. Hu and R.C. Eberhart. Tracking dynamic systems with PSO: Where's the cheese? In *Proceedings of the Workshop on Particle Swarm Optimization*, pages 80–83, 2001.

X. Hu and R.C. Eberhart. Adaptive particle swarm optimisation: detection and response to dynamic systems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1666–1670. IEEE, 2002.

J. Ilonen, J. Kamarainen, and J. Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105, Kluwer Academic Publishers, 2003. ISSN 1370-4621.

S. Janson and M. Middendorf. A hierachical particle swarm optimizer. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 770–776. IEEE, 2003.

S. Janson and M. Middendorf. A hierachical particle swarm optimizer for dynamc optimization problems. In *Applications of evolutionary computing*, volume 3005 of *LNCS*, pages 513–524. Springer, 2004.

Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, IEEE, 2005.

T. Jones and S. Forrest. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In *Proceedings of the International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

M. Kamosi, A. B. Hashemi, and M. R. Meybodi. A hibernating multi-swarm optimization algorithm for dynamic environments. In *Proceedings of the World Congress on Nature & Biologically Inspired Computing*, pages 363–369. IEEE, 2010a.

M. Kamosi, A. B. Hashemi, and M. R. Meybodi. A new particle swarm optimization algorithm for dynamic environments. In *Proceedings of the Swarm, Evolutionary, and Memetic Computing Conference*, Lecture Notes in Computer Science, pages 129–138. Springer, 2010b.

D. Karaboga and S. Okdem. A simple and global optimization algorithm for engineering problems: Differential evolution algorithm. *Turkish Journal of Electrical Engineering and Computer Sciences*, 12(1):53–60, Scientific and Technological Research Council of Turkey, 2004.

J. Kennedy. Bare bones particle swarms. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 80 – 87. IEEE, 2003.

J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, pages 1942–1948. IEEE, 1995.

B. Kiraz, S. Uyar, and E. Özcan. An investigation of selection hyper-heuristics in dynamic environments. In *Applications of Evolutionary Computation*, Lecture Notes in Computer Science, pages 314–323. Springer, 2011. ISBN 978-3-642-20524-8.

P. Korošec and J. Šilc. The differential ant-stigmergy algorithm applied to dynamic optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 407–414. IEEE, 2009.

J.R. Koza. *On the programming of Computers by Means of Natural Selection*. MIT Press, 1992.

T. Krink, J.S. Vesterstrom, and J. Riget. Particle swarm optimisation with spatial particle extension. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1474 –1479. IEEE, 2002.

B.J. Leonard, A.P. Engelbrecht, and A.B. van Wyk. Heterogeneous particle swarms in dynamic environments. In *Proceedings of the IEEE Symposium on Swarm Intelligence*, pages 1–8. IEEE, 2011.

C. Li and S. Yang. A generalized approach to construct benchmark problems for dynamic optimization. In *Proceedings of the International Conference on Simulated Evolution and Learning*, pages 391–400. Springer, 2008.

C. Li and S. Yang. A clustering particle swarm optimizer for dynamic optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 439–446. IEEE, 2009.

C. Li and S. Yang. A general framework of multi-population methods with clustering in undetectable dynamic environments. *IEEE Transactions on Evolutionary Computation*, IEEE, 2012.

C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H. g. Beyer, and P. N. Suganthan. *Benchmark Generator for CEC2009 Competition on Dynamic Optimization*. University of Leicester, University of Birmingham, Nanyang Technological University, Technical Report, 2008.

C. Li, S. Yang, and D. A. Pelta. *Benchmark Generator for the IEEE WCCI-2012 Competition on Evolutionary Computation for Dynamic Optimization Problems*. http://www.ieee-wcci2012.org/ieee-wcci2012/, 2011.

X. Li, J. Branke, and T. Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 51–58. ACM, 2006.

X. Li and K. H. Dam. Comparing particle swarms for tracking extrema in dynamic environments. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 1772– 1779. IEEE, 2003.

Y. Li, L. Rao, R. He, G. Xu, Q. Wu, M. Ge, and W. Tan. Image reconstruction of eit using differential evolution algorithm. In *Proceedings of the IEEE International Conference on Engineering in Medicine and Biology Society*, pages 1011–1014. IEEE, 2003.

Y.C. Lin, K.S. Hwang, and F.S. Wang. Plant scheduling and planning using mixed-integer hybrid differential evolution with multiplier updating. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 593–600. IEEE, 2000.

J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9(6):448–462, Springer, 2005. ISSN 1432-7643.

M. Locatelli. A Note on the Griewank Test Function. *Journal of Global Optimization*, 25: 169–174, Kluwer Academic Publishers, 2003.

M. Lunacek and D. Whitley. The dispersion metric and the CMA evolution strategy. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 477–484. ACM, 2006.

R.I. Lung and D. Dumitrescu. A new collaborative evolutionary-swarm optimization technique. In *Proceedings of the Conference on Genetic and evolutionary computation*, pages 2817–2820. ACM, 2007.

R.I. Lung and D. Dumitrescu. Evolutionary swarm cooperative optimization in dynamic environments. *Natural Computing: an international journal*, 9(1):83–94, Kluwer Academic Publishers, 2010. ISSN 1567-7818.

G.D. Magoulas, V.P. Plagianakos, and M.N. Vrahatis. Neural network-based colonoscopic diagnosis using on-line learning and differential evolution. *Applied Soft Computing*, 4 (4):369–379, Elsevier, 2004.

K.M. Malan and A. P. Engelbrecht. Quantifying ruggedness of continuous landscapes using entropy. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1440–1447. IEEE, 2009.

K.M. Malan and A. P. Engelbrecht. Techniques for characterising fitness landscapes: How they have evolved and a way forward. In *Proceedings of the International Conference on Metaheuristics and Nature Inspired Computation*, 2010.

R. Mendes and A. Mohais. DynDE: a differential evolution for dynamic optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2808–2815. IEEE, 2005.

E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello. A comparative study of differential evolution variants for global optimization. In *Proceedings of the Conference on Genetic and evolutionary computation*, pages 485–492. ACM, 2006.

S. Moalla, A.M. Alimi, and N. Derbel. Design of beta neural systems using differential evolution. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2002.

N. Mori, S. Imanishi, H. Kita, and Y. Nishikawa. Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. In T. Bäck, editor, *Proceedings of the International Conference on Genetic Algorithms*, pages 299–306. Morgan Kaufmann, 1997.

N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In *Parallel Problem Solving from Nature*, number 1141 in LNCS, pages 513–522. Springer, 1996.

N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In *Parallel Problem Solving from Nature*, number 1498 in LNCS, pages 149–158. Springer, 1998.

R. W. Morrison. Performance measurement in dynamic environments. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 5–8, 2003.

R.W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Springer, 2004. ISBN 3540212310.

I. Moser. Personal communication. May 2007.

I. Moser. Hooke-jeeves revisited. In *Proceedings IEEE Congress on Evolutionary Computation*, pages 2670–2676. IEEE, 2009. ISBN 978-1-4244-2958-5.

I. Moser and R. Chiong. Dynamic function optimisation with hybridised extremal dynamics. *Memetic Computing*, 2(2):137–148, 2010.

I. Moser and T. Hendtlass. A simple and efficient multi-component algorithm for solving dynamic function optimisation problems. In *IEEE Congress on Evolutionary Computation*, pages 252–259. IEEE, 2007.

T.T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, Elsevier, 2012.

V. Noroozi, A.B. Hashemi, and M. R. Meybodi. CellularDE: a cellular based differential evolution for dynamic optimization problems. In *Proceedings of International conference on Adaptive and natural computing algorithms - Volume Part I*, pages 340–349. Springer, 2011.

P. Novoa-Hernández, C. Corona, and D. Pelta. Efficient multi-swarm PSO algorithms for dynamic environments. *Memetic Computing*, 3:163–174, Springer, 2011. ISSN 1865-9284.

O. Olorunda and A. P. Engelbrecht. Measuring exploration/exploitation in particle swarms using swarm diversity. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1128–1134. IEEE, 2008.

M. Omran, A. Salman, and A. Engelbrecht. Self-adaptive differential evolution. In *Proceedings of the International Conference on Computational Intelligence and Security*, pages 192–199, 2005a.

M.G.H. Omran, A.P. Engelbrecht, and A. Salman. Differential evolution methods for unsupervised image classification. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 966–973. IEEE, 2005b.

M.G.H. Omran, A.P. Engelbrecht, and A. Salman. Bare bones differential evolution. *European Journal of Operational Research*, 196(1):128 – 139, Elsevier, 2009. ISSN 0377-2217.

F. Oppacher and M. Wineberg. The shifting balance genetic algorithm: Improving the ga in a dynamic environment. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 504 – 510. Morgan Kaufmann, 1999.

D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 98–103. IEEE, 2004.

D. Parrott and X. Li. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computing*, pages 440–458, IEEE, 2006.

M.E.H. Pedersen. Good Parameters for Differential Evolution. Technical Report HL1002, Hvass Laboratories, 2010.

K. Price, R. Storn, and J. Lampinen. *Differential evolution - A practical approach to global optimization*. Springer, 2005. ISBN 3-540-20950-6.

A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1785–1791. IEEE, 2005.

B. Qu and P. N. Suganthan. Novel multimodal problems and differential evolution with ensemble of restricted tournament selection. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2010.

A. Rae and S. Parameswaran. Application-specific heterogeneous multiprocessor synthesis using differential evolution. In *Proceedings of the International Symposium On System Synthesis*, page 8388. ACM, 1998.

C. L. Ramsey and J. J. Grefenstette. Case-based initialization of genetic algorithms. In *International Conference on Genetic Algorithms*, pages 84–91. Morgan Kaufmann, 1993.

I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution*. Frammann-Holzboog Verlag, 1973.

R.G. Reynolds and W. Sverdlik. Problem solving using cultural algorithms. In *Proceedings of the International Conference on Evolutionary Computation*, pages 1004–1008. IEEE, 1994.

H. Richter. Detecting change in dynamic fitness landscapes. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1613–1620. IEEE, 2009.

J. Riget and J.S. Vesterstrøm. A diversity-guided particle swarm optimizer - the ARPSO. Technical report, EVALife, 2002.

J. Rönkkönen, S. Kukkonen, and K. V. Price. Real-parameter optimization with differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 506–513. IEEE, 2005.

K. Rzadca and F. Seredynski. Heterogeneous multiprocessor scheduling with differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2840–2847. IEEE, 2005.

A. Salman, A. P. Engelbrecht, and M.G.H. Omran. Empirical analysis of self-adaptive differential evolution. *European Journal of Operational Research*, 183(2):785 – 804, Elsevier, 2007. ISSN 0377-2217.

B. Sareni and L. Krahenbuhl. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation*, 2(3):97–106, IEEE, 1998.

A. Simões and E. Costa. The influence of population and memory sizes on the evolutionary algorithm's performance for dynamic environments. In *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, pages 705–714. Springer, 2009a. ISBN 978-3-642-01128-3.

A. Simões and E. Costa. Prediction in evolutionary algorithms for dynamic environments using markov chains and nonlinear regression. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 883–890. ACM, 2009b.

R. Storn. On the usage of differential evolution for function optimization. In *Proceedings of the Conference of the North American Fuzzy Information Processing Society*, pages 519–523. IEEE, 1996.

R. Storn and K. Price. Minimizing the real functions of the ICEC96 contest by differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 842–844. IEEE, 1996.

R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, Springer, 1997.

J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, 10(8):673–686, Springer, 2006. ISSN 1432-7643.

R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 382–1389. IEEE, 2004.

I.C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317325, Elsevier, 2003.

K. Trojanowski. B-cell algorithm as a parallel approach to optimization of moving peaks benchmark tasks. In *Proceedings of the International Conference on Computer Information Systems and Industrial Management Applications*, pages 143–148. CISIM, 2007.

K. Trojanowski and Z. Michalewicz. Evolutionary algorithms for non-stationary environments. In *Proceedings of the 8th Workshop: Intelligent Information Systems*, pages 229–240. ICS PAS Press, 1999a.

K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1843–1850. IEEE, 1999b.

R. K. Ursem. Multinational GA optimization techniques in dynamic environments. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 19–26. Morgan Kaufmann, 2000.

R. K. Ursem. Diversity-guided evolutionary algorithms. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pages 462–474. Springer, 2002.

F. van den Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971, Elsevier, 2006.

F. Vavak, K. Jukes, and T. C. Fogarty. Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search. In T. Bäck, editor, *Proceedings of the International Conference on Genetic Algorithms*, pages 719–726. Morgan Kaufmann, 1997.

J. Vesterstrøm and R. Thomsen. A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1980–1987. IEEE, 2004.

K. Weicker. Performance measures for dynamic environments. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pages 64–73. Springer, 2002.

K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2039–2046. IEEE, 1999.

D. H. Wolpert and W. G. Macready. No Free Lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, IEEE, 1997.

X. Xu and R.D. Dony. Differential evolution with Powells direction set method in medical image registration. In *Proceedings of the IEEE International Symposium on Biomedical Imaging*, pages 732–735. IEEE, 2004.

F. Xue, A. C. Sanderson, and R. J. Graves. Pareto-based multi-objective differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 862–869. IEEE, 2003.

S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. In *Proceedings of the Conference on Genetic and evolutionary computation*, pages 1115–1122. ACM, 2005.

S. Yang and C. Li. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 14 (6):959 –974, IEEE, 2010. ISSN 1089-778X.

D. Zaharie. Critical values for the control parameters of differential evolution algorithms. In *Proceedings of the International Conference on Soft Computing*, pages 62–67, 2002a.

D. Zaharie. Parameter adaptation in differential evolution by controlling the population diversity. In *The International Workshop on Symbolic and Numeric Algorithms for Scientific Computing*, volume 2, Analele University, Timisoara, 2002b.

D. Zaharie and F. Zamfirache. Diversity enhancing mechanisms for evolutionary optimization in static and dynamic environments. In *Proceedings of the Romanian-Hungarian Joint Symposium on Applied Computational Intelligence*, pages 460–471, 2006.

J. Zhang and A. C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, IEEE, 2009.

W. Zhang and X. Xie. DEPSO: Hybrid particle swarm with differential evolution operator. In *Proceedings of the IEEE International Conference on System, Man, and Cybernetics*, pages 3816–3821. IEEE, 2003.

T. Zhu, W. Luo, and Z. Li. An adaptive strategy for updating the memory in evolutionary algorithms for dynamic optimization. In *Proceedings of the IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, pages 8–15. IEEE, 2011.

Appendix A

Parameter dependence of jDE

Section 3.4.2.6 concluded that a disadvantage of the *jDE* algorithm of Brest *et al.* [2009] is that it contains 16 parameters which may have to be fine-tuned for optimal results. This appendix empirically shows that *jDE* is sensitive to at least two of these parameters.

An experiment was designed to determine how various combinations of values for τ_3 and τ_5 affect the offline error of *jDE*. The first parameter, τ_3, is the age that must be reached by the best individual in a sub-population before the sub-population may be reinitialised. The second parameter, τ_5, is the age that must be reached by an individual in a sub-population before the individual may be reinitialised (refer to Section 3.4.2.3). Brest *et al.* [2009] suggested that $\tau_3 = 30$ and $\tau_5 = 25$ be used as defaults.

Four benchmark instances were utilised in these experiments: GDBG function F_3 using change periods of 5 000 and 100 000 function evaluations, and GDBG F_5 using change periods of 5 000 and 100 000 function evaluations. Change type T_1 was used in 5 dimensions for all four environments. All the combinations of values for parameters τ_3 and τ_5, which were increased in increments of 5 from 5 to 50, were evaluated. Results were averaged over 30 repeats of each of the experiments.

Figures A.1 to A.4 give the offline errors of *jDE* on F_3 with a change period of 5 000, F_3 with a change period of 100 000, F_5 with a change period of 5 000, and F_5 with a change period of 100 000, respectively. The settings for τ_3 and τ_5 had a large impact on the offline error, especially when using a change period of 5 000.

The optimal settings for τ_3 and τ_5 on F_3 with a change period of 5 000 were found to

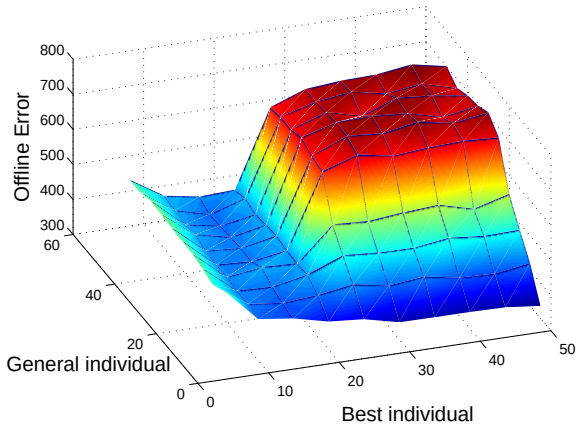

Figure A.1: Offline errors of *jDE* on the GDBG function F_3 using change type T_1 in 5 dimensions with a change period of 5 000 function evaluations for various combinations of settings for the parameters τ_3 and τ_5.

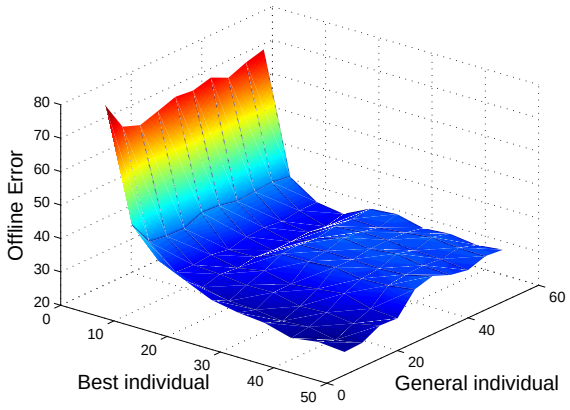

Figure A.2: Offline errors of *jDE* on the GDBG function F_3 using change type T_1 in 5 dimensions with a change period of 100 000 function evaluations for various combinations of settings for the parameters τ_3 and τ_5.

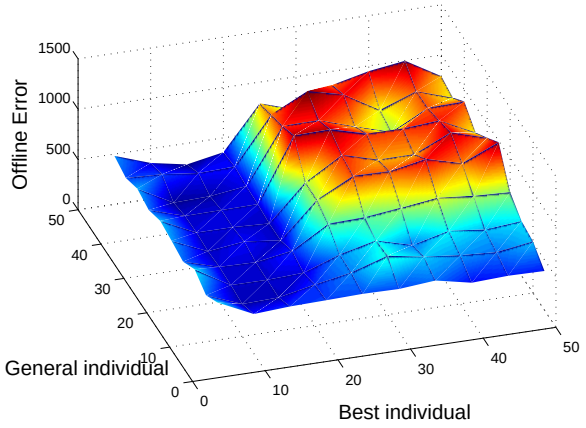

Figure A.3: Offline errors of *jDE* on the GDBG function F_5 using change type T_1 in 5 dimensions with a change period of 5 000 function evaluations for various combinations of settings for the parameters τ_3 and τ_5.

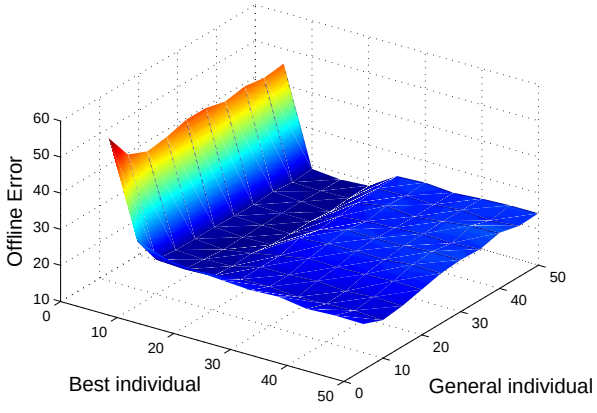

Figure A.4: Offline errors of *jDE* on the GDBG function F_5 using change type T_1 in 5 dimensions with a change period of 100 000 function evaluations for various combinations of settings for the parameters τ_3 and τ_5.

be $\tau_3 = 50$ and $\tau_5 = 5$. The values $\tau_3 = 15$ and $\tau_5 = 50$ were found to be the optimal settings on F_3 with a change period of 100 000. The optimal settings for τ_3 and τ_5 on F_5 with a change period of 5 000 were found be $\tau_3 = 45$ and $\tau_5 = 10$. The optimal settings for τ_3 and τ_5 on F_5 with a change period of 100 000 were found be $\tau_3 = 20$ and $\tau_5 = 40$. The average offline errors produced by the optimal settings were found to be statistically significantly better than the average offline errors produced by the default settings in each of the four environments. *jDE* is thus clearly sensitive to values of the parameters τ_3 and τ_5.

Appendix B

Additional Results - Chapter 4

This appendix contains results that were omitted from Chapter 4 due to space constraints. Tables B.1 and B.2 give the performance analysis of CDE compared to CPE on variations of the standard set of environments. Tables B.3 and B.4 give the performance analysis of CDE compared to RMC. The comparative performance analysis of CDE and CjDE is given in Tables B.5 and B.6.

Table B.1: CDE vs CPE performance analysis - Part 1

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑4 ↓0**
5	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑8 ↓0**
10	(2)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑4 ↓1**
20	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑5 ↓0**
40	(2)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑4 ↓0**
C	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑3 ↓0**
S	(6)	*↑0 ↓2*	↑0 ↓0	**↑1 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑24 ↓2**
		GBG								
F_{1a}	(6)	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑4 ↓0**
F_{1b}	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑2 ↓2
F_2	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑4 ↓0**
F_3	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**
F_4	(6)	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑5 ↓0**
F_5	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*
F_6	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑1 ↓3*
T_1	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1
T_2	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1	**↑2 ↓1**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑4 ↓2**
T_3	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1
T_4	(7)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑3 ↓0**
T_5	(7)	**↑3 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑6 ↓0**
T_6	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑1 ↓0**	**↑3 ↓2**
All	(54)	**↑4 ↓2**	**↑2 ↓0**	**↑3 ↓0**	**↑7 ↓4**	**↑10 ↓1**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓0**	**↑45 ↓8**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
10	(2)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
20	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
C	(6)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓1**
S	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
		GBG								
F_{1a}	(6)	↑1 ↓1	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑3 ↓2**
F_{1b}	(6)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑1 ↓2*
F_2	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑1 ↓1
F_3	(6)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	↑1 ↓1	**↑4 ↓1**
F_4	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑1 ↓1
F_5	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**
F_6	(6)	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1
T_1	(7)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**
T_2	(7)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	**↑5 ↓1**
T_3	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**
T_4	(7)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1
T_5	(7)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑2 ↓2
T_6	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	*↑2 ↓3*
All	(54)	**↑2 ↓1**	**↑3 ↓1**	*↑1 ↓2*	↑1 ↓1	*↑1 ↓2*	**↑1 ↓0**	**↑4 ↓0**	↑2 ↓2	**↑15 ↓9**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
20	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
40	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
C	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
S	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
		GBG								
F_{1a}	(6)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓2*
F_{1b}	(6)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
F_2	(6)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
F_3	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
F_4	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑1 ↓3*
F_5	(6)	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑2 ↓2
F_6	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	↑1 ↓1
T_1	(7)	*↑0 ↓2*	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	↑0 ↓0	*↑1 ↓5*
T_2	(7)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓3*
T_3	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**
T_4	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*
T_5	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
T_6	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑1 ↓1
All	(54)	*↑0 ↓2*	↑1 ↓1	*↑0 ↓1*	*↑1 ↓2*	↑0 ↓0	**↑1 ↓0**	*↑1 ↓2*	*↑0 ↓2*	*↑4 ↓10*

Table B.2: CDE vs CPE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**
10	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
20	(2)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
40	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
C	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
S	(6)	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑2 ↓2
		GBG								
F_{1a}	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
F_{1b}	(6)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
F_2	(6)	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓1**
F_3	(6)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓1**
F_4	(6)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓3*
F_5	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
F_6	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
T_1	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	**↑2 ↓1**
T_2	(7)	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*
T_3	(7)	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓2*
T_4	(7)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑1 ↓2*
T_5	(7)	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**
T_6	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**
All	(54)	*↑0 ↓3*	**↑3 ↓1**	*↑0 ↓1*	**↑2 ↓1**	*↑0 ↓1*	*↑0 ↓2*	**↑1 ↓0**	**↑3 ↓0**	↑9 ↓9
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**
5	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
20	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
40	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
C	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**
S	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
		GBG								
F_{1a}	(6)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	**↑1 ↓0**	↑1 ↓1	**↑3 ↓2**
F_{1b}	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	*↑0 ↓3*
F_2	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑1 ↓1
F_3	(6)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓0**
F_4	(6)	↑0 ↓0	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓1**
F_5	(6)	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	↑2 ↓2
F_6	(6)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑2 ↓2
T_1	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓1**	↑0 ↓0	*↑1 ↓2*	↑3 ↓3
T_2	(7)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑2 ↓2
T_3	(7)	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑3 ↓2**
T_4	(7)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*
T_5	(7)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑1 ↓3*
T_6	(7)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑3 ↓0**
All	(54)	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓1**	↑1 ↓1	*↑0 ↓2*	**↑4 ↓1**	↑2 ↓2	*↑1 ↓4*	**↑13 ↓11**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑5 ↓0**
5	(10)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑9 ↓0**
10	(10)	*↑0 ↓1*	**↑1 ↓0**	*↑0 ↓1*	**↑2 ↓1**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑6 ↓3**
20	(10)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑6 ↓1**
40	(10)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑3 ↓1**
80	(10)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑4 ↓0**
C	(6)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑7 ↓1**
S	(6)	*↑0 ↓3*	**↑1 ↓0**	**↑1 ↓0**	**↑4 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑26 ↓4**
		GBG								
F_{1a}	(30)	**↑3 ↓2**	**↑2 ↓0**	↑0 ↓0	↑1 ↓1	*↑0 ↓2*	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓1**	**↑10 ↓6**
F_{1b}	(30)	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓2*	*↑1 ↓3*	↑1 ↓1	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	*↑3 ↓9*
F_2	(30)	*↑0 ↓1*	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**	**↑2 ↓0**	**↑2 ↓1**	↑0 ↓0	**↑8 ↓4**
F_3	(30)	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑3 ↓0**	**↑3 ↓1**	**↑10 ↓2**
F_4	(30)	**↑2 ↓0**	*↑0 ↓1*	**↑3 ↓1**	*↑0 ↓1*	↑1 ↓1	**↑2 ↓1**	↑1 ↓1	*↑0 ↓2*	**↑9 ↓8**
F_5	(30)	↑1 ↓1	**↑3 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	*↑0 ↓1*	*↑0 ↓2*	*↑1 ↓2*	**↑8 ↓6**
F_6	(30)	↑0 ↓0	**↑3 ↓0**	*↑0 ↓1*	*↑1 ↓3*	*↑0 ↓1*	↑0 ↓0	↑1 ↓1	*↑0 ↓1*	*↑5 ↓7*
T_1	(35)	*↑0 ↓2*	**↑2 ↓0**	*↑0 ↓1*	*↑1 ↓2*	**↑1 ↓0**	↑2 ↓2	*↑1 ↓2*	↑2 ↓2	*↑9 ↓11*
T_2	(35)	**↑1 ↓0**	↑2 ↓2	**↑2 ↓1**	↑2 ↓2	↑2 ↓2	↑0 ↓0	**↑2 ↓1**	*↑0 ↓1*	**↑11 ↓9**
T_3	(35)	↑1 ↓1	↑0 ↓0	*↑0 ↓2*	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑6 ↓5**
T_4	(35)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑5 ↓5
T_5	(35)	**↑4 ↓2**	**↑3 ↓1**	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	↑1 ↓1	**↑12 ↓6**
T_6	(35)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	*↑1 ↓2*	↑1 ↓1	**↑4 ↓0**	↑1 ↓1	↑2 ↓2	**↑10 ↓6**
All	(270)	↑8 ↓8	**↑10 ↓3**	**↑6 ↓5**	**↑12 ↓9**	**↑11 ↓6**	**↑13 ↓3**	**↑14 ↓5**	**↑12 ↓8**	**↑86 ↓47**

Table B.3: CDE vs RMC performance analysis - Part 1

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑5 ↓0**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑12 ↓0**
10	(2)	↑0 ↓0	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓1**
20	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑10 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑7 ↓1**
80	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑6 ↓0**
C	(6)	↑0 ↓0	**↑2 ↓1**	**↑3 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑3 ↓0**	↑1 ↓1	**↑22 ↓2**
S	(6)	↑0 ↓0	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑30 ↓0**
		GBG								
F_{1a}	(6)	*↑0 ↓2*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓3*	*↑0 ↓6*	*↑0 ↓6*	**↑23 ↓17**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	**↑23 ↓17**
F_2	(6)	*↑0 ↓1*	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓1**	↑3 ↓3	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑17 ↓23*
F_3	(6)	↑0 ↓0	**↑2 ↓0**	**↑3 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑6 ↓0**
F_4	(6)	*↑0 ↓1*	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	*↑0 ↓6*	*↑0 ↓6*	**↑24 ↓14**
F_5	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	*↑1 ↓4*	*↑0 ↓6*	**↑29 ↓12**
F_6	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑4 ↓1**	*↑1 ↓2*	**↑32 ↓4**
T_1	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	**↑22 ↓20**
T_2	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	*↑1 ↓5*	*↑0 ↓5*	**↑28 ↓12**
T_3	(7)	*↑0 ↓1*	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑3 ↓1**	*↑1 ↓4*	*↑1 ↓5*	**↑28 ↓12**
T_4	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑1 ↓4*	*↑0 ↓5*	*↑0 ↓6*	**↑25 ↓16**
T_5	(7)	*↑0 ↓1*	**↑3 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑3 ↓2**	*↑2 ↓4*	*↑0 ↓5*	**↑23 ↓14**
T_6	(7)	↑0 ↓0	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	↑3 ↓3	*↑1 ↓5*	*↑0 ↓5*	**↑28 ↓13**
All	(54)	*↑0 ↓4*	**↑31 ↓1**	**↑46 ↓0**	**↑46 ↓1**	**↑40 ↓3**	**↑22 ↓18**	*↑13 ↓29*	*↑8 ↓33*	**↑206 ↓89**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	*↑3 ↓6*
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑6 ↓0**
10	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
20	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓0**
C	(6)	↑0 ↓0	**↑2 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑31 ↓0**
S	(6)	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓1**	**↑21 ↓6**
		GBG								
F_{1a}	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑0 ↓6*	*↑0 ↓5*	**↑29 ↓12**
F_{1b}	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	*↑0 ↓5*	*↑0 ↓5*	**↑27 ↓11**
F_2	(6)	↑0 ↓0	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑4 ↓1**	↑3 ↓3	*↑1 ↓4*	*↑0 ↓6*	**↑22 ↓14**
F_3	(6)	↑0 ↓0	**↑2 ↓0**	**↑5 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑11 ↓1**
F_4	(6)	**↑1 ↓0**	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑3 ↓1**	*↑2 ↓4*	*↑0 ↓6*	**↑23 ↓12**
F_5	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	**↑40 ↓2**
F_6	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑40 ↓1**
T_1	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑2 ↓4*	*↑0 ↓6*	**↑33 ↓10**
T_2	(7)	**↑1 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑4 ↓1**	*↑2 ↓4*	*↑2 ↓4*	**↑33 ↓9**
T_3	(7)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑3 ↓1**	*↑2 ↓3*	↑2 ↓2	**↑32 ↓6**
T_4	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓2**	*↑1 ↓5*	**↑33 ↓7**
T_5	(7)	*↑0 ↓1*	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓2**	**↑4 ↓2**	*↑2 ↓4*	*↑2 ↓4*	**↑27 ↓13**
T_6	(7)	*↑0 ↓1*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	*↑2 ↓5*	**↑34 ↓8**
All	(54)	*↑1 ↓2*	**↑33 ↓1**	**↑43 ↓1**	**↑48 ↓1**	**↑43 ↓3**	**↑35 ↓5**	**↑24 ↓19**	*↑17 ↓27*	**↑244 ↓59**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑8 ↓1**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑7 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
80	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
C	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑38 ↓0**
S	(6)	↑0 ↓0	**↑1 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓1**	**↑5 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑25 ↓1**
		GBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	**↑38 ↓2**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	*↑1 ↓3*	**↑35 ↓3**
F_2	(6)	↑0 ↓0	**↑3 ↓0**	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑30 ↓0**
F_3	(6)	↑0 ↓0	↑0 ↓0	**↑5 ↓0**	**↑4 ↓0**	**↑2 ↓1**	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓4*	**↑11 ↓9**
F_4	(6)	↑0 ↓0	**↑3 ↓0**	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑31 ↓0**
F_5	(6)	↑1 ↓1	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓1**
F_6	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑39 ↓0**
T_1	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑42 ↓5**
T_2	(7)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓1**	↑2 ↓2	**↑36 ↓3**
T_3	(7)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑3 ↓1**	**↑36 ↓2**
T_4	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓3**	**↑41 ↓3**
T_5	(7)	**↑1 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑34 ↓0**
T_6	(7)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓2**	**↑35 ↓2**
All	(54)	↑1 ↓1	**↑30 ↓0**	**↑43 ↓0**	**↑49 ↓0**	**↑47 ↓2**	**↑47 ↓1**	**↑37 ↓3**	**↑33 ↓9**	**↑287 ↓16**

Table B.4: CDE vs RMC performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑12 ↓0**
10	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
20	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
40	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
C	(6)	**↑1 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑38 ↓0**
S	(6)	↑0 ↓0	**↑1 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑35 ↓0**
		GBG								
F_{1a}	(6)	**↑1 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑40 ↓0**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑40 ↓0**
F_2	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑37 ↓2**
F_3	(6)	*↑0 ↓1*	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓5*	**↑15 ↓13**
F_4	(6)	*↑0 ↓1*	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑36 ↓1**
F_5	(6)	↑0 ↓0	**↑2 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑3 ↓1**	**↑28 ↓1**
F_6	(6)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑36 ↓0**
T_1	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑41 ↓4**
T_2	(7)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑38 ↓3**
T_3	(7)	*↑0 ↓1*	**↑4 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑40 ↓2**
T_4	(7)	*↑0 ↓1*	**↑3 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑41 ↓4**
T_5	(7)	*↑0 ↓1*	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓1**	**↑36 ↓3**
T_6	(7)	**↑1 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑5 ↓0**	**↑4 ↓0**	**↑36 ↓1**
All	(54)	*↑2 ↓3*	**↑30 ↓0**	**↑44 ↓0**	**↑51 ↓0**	**↑53 ↓0**	**↑46 ↓3**	**↑44 ↓4**	**↑35 ↓7**	**↑305 ↓17**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑9 ↓5**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	↑1 ↓1	*↑0 ↓1*	**↑9 ↓3**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑2 ↓0**	**↑11 ↓3**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑12 ↓1**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
C	(6)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑38 ↓0**
S	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	↑3 ↓3	**↑4 ↓2**	*↑2 ↓4*	**↑4 ↓2**	**↑30 ↓12**
		GBG								
F_{1a}	(6)	↑0 ↓0	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
F_{1b}	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓1**
F_2	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑40 ↓1**
F_3	(6)	↑0 ↓0	*↑0 ↓4*	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	↑0 ↓0	*↑0 ↓5*	**↑19 ↓9**
F_4	(6)	↑0 ↓0	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑40 ↓1**
F_5	(6)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓1**	**↑4 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑20 ↓1**
F_6	(6)	↑0 ↓0	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑32 ↓3**
T_1	(7)	↑0 ↓0	**↑5 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑39 ↓2**
T_2	(7)	↑0 ↓0	**↑6 ↓1**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓3**	**↑44 ↓4**
T_3	(7)	*↑0 ↓1*	**↑2 ↓1**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓2**	**↑37 ↓4**
T_4	(7)	↑0 ↓0	**↑5 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑44 ↓2**
T_5	(7)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑35 ↓4**
T_6	(7)	↑0 ↓0	**↑3 ↓0**	**↑3 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑33 ↓0**
All	(54)	*↑0 ↓1*	**↑35 ↓4**	**↑44 ↓1**	**↑51 ↓1**	**↑47 ↓3**	**↑46 ↓2**	**↑39 ↓5**	**↑38 ↓11**	**↑300 ↓28**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑0 ↓0	**↑7 ↓1**	**↑6 ↓1**	**↑6 ↓2**	**↑5 ↓3**	**↑5 ↓2**	**↑4 ↓1**	**↑5 ↓2**	**↑38 ↓12**
5	(10)	↑0 ↓0	**↑5 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑7 ↓1**	**↑9 ↓0**	**↑8 ↓1**	**↑4 ↓1**	**↑50 ↓3**
10	(10)	**↑1 ↓0**	**↑4 ↓1**	**↑8 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑57 ↓1**
20	(10)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓0**	**↑56 ↓3**
40	(10)	↑0 ↓0	**↑1 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑8 ↓1**	**↑9 ↓1**	**↑53 ↓2**
80	(10)	↑0 ↓0	**↑2 ↓0**	**↑7 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑54 ↓0**
C	(6)	**↑1 ↓0**	**↑15 ↓1**	**↑25 ↓0**	**↑29 ↓0**	**↑27 ↓0**	**↑27 ↓0**	**↑25 ↓0**	**↑18 ↓1**	**↑167 ↓2**
S	(6)	↑0 ↓0	**↑8 ↓1**	**↑19 ↓1**	**↑23 ↓2**	**↑20 ↓5**	**↑24 ↓3**	**↑21 ↓4**	**↑26 ↓3**	**↑141 ↓19**
		GBG								
F_{1a}	(30)	*↑1 ↓3*	**↑26 ↓0**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑23 ↓3**	**↑18 ↓12**	**↑14 ↓13**	**↑170 ↓31**
F_{1b}	(30)	*↑0 ↓2*	**↑29 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑21 ↓5**	**↑16 ↓11**	*↑11 ↓14*	**↑166 ↓32**
F_2	(30)	*↑0 ↓2*	**↑21 ↓0**	**↑23 ↓0**	**↑29 ↓1**	**↑25 ↓4**	**↑21 ↓9**	**↑15 ↓10**	*↑12 ↓14*	**↑146 ↓40**
F_3	(30)	*↑0 ↓1*	↑4 ↓4	**↑21 ↓0**	**↑19 ↓0**	**↑14 ↓1**	↑4 ↓4	*↑0 ↓7*	*↑0 ↓15*	**↑62 ↓32**
F_4	(30)	*↑1 ↓2*	**↑17 ↓0**	**↑24 ↓0**	**↑30 ↓0**	**↑28 ↓1**	**↑25 ↓2**	**↑17 ↓10**	*↑12 ↓13*	**↑154 ↓28**
F_5	(30)	↑1 ↓1	**↑19 ↓0**	**↑23 ↓1**	**↑25 ↓0**	**↑27 ↓0**	**↑25 ↓2**	**↑21 ↓4**	**↑16 ↓9**	**↑157 ↓17**
F_6	(30)	↑0 ↓0	**↑20 ↓0**	**↑27 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑26 ↓1**	**↑24 ↓2**	**↑22 ↓5**	**↑179 ↓8**
T_1	(35)	*↑0 ↓2*	**↑29 ↓1**	**↑32 ↓0**	**↑30 ↓0**	**↑29 ↓2**	**↑23 ↓8**	**↑19 ↓12**	*↑15 ↓16*	**↑177 ↓41**
T_2	(35)	**↑1 ↓0**	**↑25 ↓1**	**↑32 ↓0**	**↑34 ↓0**	**↑32 ↓0**	**↑26 ↓3**	**↑17 ↓11**	*↑12 ↓16*	**↑179 ↓31**
T_3	(35)	*↑0 ↓3*	**↑19 ↓1**	**↑30 ↓0**	**↑34 ↓0**	**↑33 ↓1**	**↑24 ↓2**	**↑18 ↓8**	**↑15 ↓11**	**↑173 ↓26**
T_4	(35)	*↑0 ↓2*	**↑26 ↓1**	**↑32 ↓0**	**↑33 ↓0**	**↑32 ↓0**	**↑24 ↓5**	**↑21 ↓8**	↑16 ↓16	**↑184 ↓32**
T_5	(35)	*↑1 ↓3*	**↑20 ↓0**	**↑25 ↓1**	**↑30 ↓1**	**↑27 ↓3**	**↑22 ↓4**	**↑16 ↓10**	**↑14 ↓12**	**↑155 ↓34**
T_6	(35)	↑1 ↓1	**↑17 ↓0**	**↑25 ↓0**	**↑32 ↓0**	**↑30 ↓0**	**↑26 ↓4**	**↑20 ↓7**	**↑15 ↓12**	**↑166 ↓24**
All	(270)	*↑4 ↓11*	**↑159 ↓6**	**↑220 ↓2**	**↑245 ↓3**	**↑230 ↓11**	**↑196 ↓29**	**↑157 ↓60**	**↑131 ↓87**	**↑1342 ↓209**

Table B.5: CDE vs CjDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*1*	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**13** ↓**1**
5	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**12** ↓**4**
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**4**
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**6**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑*6* ↓*8*
80	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑**2** ↓**0**	↑*4* ↓*9*
C	(6)	↑*0* ↓*5*	↑*1* ↓*5*	↑*2* ↓*3*	↑**4** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**30** ↓**13**
S	(6)	↑*0* ↓*5*	↑*0* ↓*5*	↑*2* ↓*3*	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**1**	↑**5** ↓**1**	↑**6** ↓**0**	↑**25** ↓**19**
		GBG								
F_{1a}	(6)	↑**2** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**0**
F_{1b}	(6)	↑**2** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**44** ↓**1**
F_2	(6)	↑0 ↓0	↑**2** ↓**1**	↑**5** ↓**1**	↑2 ↓2	↑**4** ↓**2**	↑**4** ↓**1**	↑**4** ↓**1**	↑**5** ↓**0**	↑**26** ↓**8**
F_3	(6)	↑**1** ↓**0**	↑*0* ↓*2*	↑**3** ↓**1**	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*4* ↓*33*
F_4	(6)	↑**2** ↓**0**	↑*1* ↓*4*	↑**4** ↓**1**	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**36** ↓**6**
F_5	(6)	↑*2* ↓*3*	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**3**
F_6	(6)	↑1 ↓1	↑**3** ↓**1**	↑**5** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**3** ↓**0**	↑**3** ↓**0**	↑**3** ↓**1**	↑**27** ↓**4**
T_1	(7)	↑**4** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**47** ↓**5**
T_2	(7)	↑1 ↓1	↑**4** ↓**1**	↑**5** ↓**0**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**39** ↓**7**
T_3	(7)	↑*0* ↓*1*	↑**4** ↓**1**	↑**7** ↓**0**	↑**5** ↓**2**	↑**5** ↓**2**	↑**4** ↓**1**	↑**4** ↓**1**	↑**5** ↓**1**	↑**34** ↓**9**
T_4	(7)	↑**5** ↓**0**	↑**3** ↓**2**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**44** ↓**8**
T_5	(7)	↑0 ↓0	↑*1* ↓*4*	↑**3** ↓**2**	↑3 ↓3	↑**4** ↓**3**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**1**	↑**23** ↓**17**
T_6	(7)	↑*0* ↓*3*	↑**4** ↓**0**	↑**7** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**36** ↓**9**
All	(54)	↑*10* ↓*15*	↑**23** ↓**18**	↑**39** ↓**9**	↑**38** ↓**11**	↑**41** ↓**11**	↑**41** ↓**8**	↑**42** ↓**8**	↑**44** ↓**7**	↑**278** ↓**87**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**11** ↓**5**
5	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**11** ↓**5**
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**6**
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**10** ↓**6**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**9** ↓**7**
80	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*3* ↓*12*
C	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*4*	↑**5** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**30** ↓**17**
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑**4** ↓**2**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑24 ↓24
		GBG								
F_{1a}	(6)	↑1 ↓1	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**41** ↓**2**
F_{1b}	(6)	↑**2** ↓**1**	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**43** ↓**2**
F_2	(6)	↑**3** ↓**0**	↑*1* ↓*3*	↑**3** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**35** ↓**6**
F_3	(6)	↑**2** ↓**0**	↑*0* ↓*6*	↑*1* ↓*2*	↑2 ↓2	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*5* ↓*34*
F_4	(6)	↑**1** ↓**0**	↑*1* ↓*2*	↑**3** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**35** ↓**3**
F_5	(6)	↑**3** ↓**2**	↑*2* ↓*4*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**41** ↓**6**
F_6	(6)	↑2 ↓2	↑*1* ↓*3*	↑**5** ↓**1**	↑**5** ↓**0**	↑**5** ↓**0**	↑**3** ↓**1**	↑3 ↓3	↑3 ↓3	↑**27** ↓**13**
T_1	(7)	↑**2** ↓**0**	↑**6** ↓**1**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**44** ↓**6**
T_2	(7)	↑*0* ↓*2*	↑**3** ↓**1**	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**40** ↓**7**
T_3	(7)	↑*0* ↓*2*	↑*1* ↓*3*	↑**5** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**34** ↓**11**
T_4	(7)	↑**4** ↓**0**	↑*2* ↓*5*	↑**4** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**40** ↓**11**
T_5	(7)	↑**3** ↓**0**	↑*0* ↓*7*	↑*3* ↓*4*	↑**4** ↓**1**	↑**4** ↓**2**	↑**5** ↓**2**	↑**5** ↓**2**	↑**5** ↓**2**	↑**29** ↓**20**
T_6	(7)	↑**5** ↓**2**	↑*2* ↓*3*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**40** ↓**11**
All	(54)	↑*14* ↓*18*	↑*14* ↓*32*	↑**32** ↓**15**	↑**45** ↓**6**	↑**44** ↓**8**	↑**44** ↓**8**	↑**44** ↓**10**	↑**44** ↓**10**	↑**281** ↓**107**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**2** ↓**0**	↑*6* ↓*9*
5	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**8** ↓**7**
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑7 ↓7
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**8** ↓**7**
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**9** ↓**6**
80	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑7 ↓*8*
C	(6)	↑*0* ↓*5*	↑*0* ↓*6*	↑*1* ↓*4*	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**30** ↓**16**
S	(6)	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑*1* ↓*4*	↑**3** ↓**2**	↑**5** ↓**0**	↑**6** ↓**0**	↑*15* ↓*28*
		GBG								
F_{1a}	(6)	↑**5** ↓**0**	↑**4** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**44** ↓**1**
F_{1b}	(6)	↑**4** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**45** ↓**0**
F_2	(6)	↑**3** ↓**1**	↑*1* ↓*3*	↑1 ↓1	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**32** ↓**7**
F_3	(6)	↑2 ↓2	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*3*	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*4* ↓*39*
F_4	(6)	↑**3** ↓**0**	↑*1* ↓*2*	↑1 ↓1	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**32** ↓**5**
F_5	(6)	↑**2** ↓**0**	↑*1* ↓*5*	↑3 ↓3	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**4** ↓**1**	↑**31** ↓**10**
F_6	(6)	↑2 ↓2	↑*1* ↓*5*	↑**3** ↓**2**	↑**5** ↓**1**	↑**5** ↓**1**	↑**3** ↓**1**	↑3 ↓3	↑*2* ↓*3*	↑**24** ↓**18**
T_1	(7)	↑0 ↓0	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**42** ↓**7**
T_2	(7)	↑*2* ↓*3*	↑*2* ↓*4*	↑**4** ↓**1**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**38** ↓**11**
T_3	(7)	↑*1* ↓*2*	↑*1* ↓*3*	↑2 ↓2	↑**7** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**5** ↓**2**	↑**5** ↓**2**	↑**32** ↓**12**
T_4	(7)	↑**6** ↓**0**	↑*2* ↓*5*	↑**3** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**41** ↓**11**
T_5	(7)	↑**6** ↓**0**	↑*0* ↓*6*	↑*2* ↓*5*	↑*3* ↓*4*	↑*3* ↓*4*	↑**3** ↓**2**	↑**5** ↓**2**	↑**4** ↓**2**	↑**26** ↓**25**
T_6	(7)	↑**6** ↓**0**	↑*2* ↓*3*	↑*2* ↓*3*	↑**6** ↓**0**	↑**5** ↓**1**	↑**4** ↓**1**	↑**4** ↓**3**	↑**4** ↓**3**	↑**33** ↓**14**
All	(54)	↑**21** ↓**15**	↑*13* ↓*34*	↑*20* ↓*23*	↑**40** ↓**12**	↑**39** ↓**11**	↑**39** ↓**9**	↑**43** ↓**10**	↑**42** ↓**10**	↑**257** ↓**124**

Table B.6: CDE vs CjDE performance analysis - Part 2

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*5* ↓*11*
5	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*5* ↓*9*
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*5* ↓*10*
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑2 ↓0**	↑*6* ↓*9*
40	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑9 ↓6**
80	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑9 ↓6**
C	(6)	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*4*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑30 ↓14**
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*4*	↑*2* ↓*4*	↑*1* ↓*4*	↑*2* ↓*4*	↑3 ↓3	↑*9* ↓*37*
		GBG								
F_{1a}	(6)	**↑3 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑41 ↓0**
F_{1b}	(6)	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑47 ↓0**
F_2	(6)	↑*2* ↓*3*	**↑1 ↓0**	**↑3 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑33 ↓6**
F_3	(6)	↑*2* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*43*
F_4	(6)	↑*2* ↓*3*	↑1 ↓1	**↑3 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑31 ↓7**
F_5	(6)	↑1 ↓1	↑*0* ↓*6*	↑*0* ↓*5*	↑2 ↓2	**↑3 ↓1**	**↑3 ↓1**	**↑4 ↓1**	**↑3 ↓2**	↑*16* ↓*19*
F_6	(6)	↑2 ↓2	↑*0* ↓*5*	↑*2* ↓*3*	**↑5 ↓1**	**↑5 ↓1**	↑3 ↓3	↑3 ↓3	↑*1* ↓*3*	↑21 ↓21
T_1	(7)	↑*1* ↓*4*	**↑4 ↓2**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑4 ↓1**	**↑38 ↓12**
T_2	(7)	↑*1* ↓*4*	↑*2* ↓*3*	**↑3 ↓2**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑35 ↓14**
T_3	(7)	↑*1* ↓*4*	↑*1* ↓*3*	↑*2* ↓*3*	**↑5 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑4 ↓2**	**↑29 ↓18**
T_4	(7)	**↑6 ↓0**	↑*2* ↓*3*	**↑4 ↓2**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑37 ↓11**
T_5	(7)	**↑2 ↓0**	↑*1* ↓*4*	↑*2* ↓*5*	↑*2* ↓*5*	↑*2* ↓*4*	↑2 ↓2	**↑4 ↓2**	**↑4 ↓2**	↑*19* ↓*24*
T_6	(7)	**↑6 ↓0**	↑*2* ↓*3*	↑3 ↓3	**↑5 ↓1**	**↑5 ↓1**	**↑4 ↓3**	**↑4 ↓3**	**↑4 ↓3**	**↑33 ↓17**
All	(54)	↑*17* ↓*22*	↑*12* ↓*30*	↑*19* ↓*26*	**↑36 ↓14**	**↑38 ↓13**	**↑35 ↓14**	**↑38 ↓14**	**↑35 ↓14**	**↑230 ↓147**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*5* ↓*11*
5	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*5* ↓*11*
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*5* ↓*11*
20	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	↑*5* ↓*7*
40	(2)	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑8 ↓6**
80	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓6**
C	(6)	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑30 ↓17**
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*3*	↑*2* ↓*3*	↑*2* ↓*3*	↑*1* ↓*4*	↑*1* ↓*4*	↑*8* ↓*35*
		GBG								
F_{1a}	(6)	**↑4 ↓1**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓1**
F_{1b}	(6)	**↑3 ↓2**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓2**
F_2	(6)	↑*2* ↓*3*	**↑2 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓4**
F_3	(6)	↑*2* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*45*
F_4	(6)	↑3 ↓3	**↑2 ↓1**	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓4**
F_5	(6)	↑*0* ↓*2*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*4*	↑*1* ↓*4*	↑3 ↓3	↑*2* ↓*3*	↑*1* ↓*4*	↑*7* ↓*32*
F_6	(6)	↑*2* ↓*3*	↑*0* ↓*6*	↑*1* ↓*3*	**↑5 ↓1**	**↑4 ↓1**	↑*2* ↓*4*	↑*2* ↓*4*	↑*1* ↓*4*	↑*17* ↓*26*
T_1	(7)	↑*0* ↓*6*	**↑4 ↓3**	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑6 ↓1**	**↑6 ↓1**	**↑4 ↓2**	**↑35 ↓19**
T_2	(7)	↑*2* ↓*4*	**↑4 ↓3**	**↑4 ↓2**	**↑5 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑36 ↓17**
T_3	(7)	↑*1* ↓*5*	↑*0* ↓*3*	↑3 ↓3	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑4 ↓2**	**↑4 ↓2**	**↑27 ↓19**
T_4	(7)	**↑4 ↓2**	↑*2* ↓*5*	↑2 ↓2	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑33 ↓19**
T_5	(7)	**↑3 ↓0**	↑*2* ↓*3*	↑3 ↓3	**↑4 ↓3**	**↑4 ↓3**	**↑4 ↓3**	**↑4 ↓3**	**↑4 ↓3**	**↑28 ↓21**
T_6	(7)	**↑6 ↓0**	↑*0* ↓*3*	↑3 ↓3	**↑5 ↓2**	**↑4 ↓2**	**↑4 ↓3**	**↑4 ↓3**	**↑4 ↓3**	**↑30 ↓19**
All	(54)	↑*16* ↓*28*	↑*12* ↓*32*	↑*20* ↓*27*	**↑37 ↓14**	**↑37 ↓14**	**↑37 ↓16**	**↑35 ↓17**	**↑33 ↓18**	**↑227 ↓166**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑*0* ↓*9*	↑*1* ↓*8*	↑*3* ↓*7*	**↑7 ↓3**	**↑7 ↓3**	**↑7 ↓3**	**↑7 ↓2**	**↑8 ↓2**	**↑40 ↓37**
5	(10)	↑*0* ↓*8*	↑*0* ↓*10*	↑*4* ↓*5*	**↑7 ↓3**	**↑7 ↓3**	**↑7 ↓3**	**↑8 ↓2**	**↑8 ↓2**	**↑41 ↓36**
10	(10)	↑*0* ↓*10*	↑*0* ↓*10*	↑*0* ↓*6*	**↑7 ↓3**	**↑7 ↓3**	**↑7 ↓2**	**↑8 ↓2**	**↑8 ↓2**	↑*37* ↓*38*
20	(10)	↑*0* ↓*10*	↑*0* ↓*10*	↑*0* ↓*10*	**↑7 ↓1**	**↑7 ↓1**	**↑8 ↓1**	**↑8 ↓2**	**↑9 ↓0**	**↑39 ↓35**
40	(10)	↑*0* ↓*9*	↑*0* ↓*10*	↑*0* ↓*10*	**↑6 ↓2**	**↑9 ↓1**	**↑8 ↓0**	**↑9 ↓0**	**↑9 ↓1**	**↑41 ↓33**
80	(10)	↑*0* ↓*7*	↑*0* ↓*10*	↑*0* ↓*10*	↑*3* ↓*6*	**↑5 ↓3**	**↑8 ↓2**	**↑8 ↓2**	**↑9 ↓1**	↑*33* ↓*41*
C	(6)	↑*0* ↓*25*	↑*1* ↓*29*	↑*5* ↓*21*	**↑26 ↓2**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑150 ↓77**
S	(6)	↑*0* ↓*28*	↑*0* ↓*29*	↑*2* ↓*27*	↑*11* ↓*16*	↑14 ↓14	**↑15 ↓11**	**↑18 ↓10**	**↑21 ↓8**	↑*81* ↓*143*
		GBG								
F_{1a}	(30)	**↑15 ↓2**	**↑21 ↓2**	**↑27 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑212 ↓4**
F_{1b}	(30)	**↑16 ↓4**	**↑26 ↓1**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑222 ↓5**
F_2	(30)	**↑10 ↓7**	↑*7* ↓*8*	**↑17 ↓4**	**↑23 ↓5**	**↑25 ↓5**	**↑26 ↓1**	**↑28 ↓1**	**↑29 ↓0**	**↑165 ↓31**
F_3	(30)	**↑9 ↓8**	↑*0* ↓*26*	↑*4* ↓*21*	↑*4* ↓*22*	↑*0* ↓*27*	↑*0* ↓*30*	↑*0* ↓*30*	↑*0* ↓*30*	↑*17* ↓*194*
F_4	(30)	**↑11 ↓6**	↑*6* ↓*10*	**↑14 ↓4**	**↑27 ↓3**	**↑28 ↓2**	**↑28 ↓0**	**↑29 ↓0**	**↑29 ↓0**	**↑172 ↓25**
F_5	(30)	↑8 ↓8	↑*8* ↓*21*	**↑15 ↓14**	**↑20 ↓6**	**↑21 ↓5**	**↑23 ↓4**	**↑23 ↓5**	**↑20 ↓7**	**↑138 ↓70**
F_6	(30)	↑*9* ↓*10*	↑*5* ↓*20*	**↑16 ↓9**	**↑25 ↓3**	**↑23 ↓4**	**↑14 ↓9**	**↑14 ↓13**	↑*10* ↓*14*	**↑116 ↓82**
T_1	(35)	↑7 ↓*10*	**↑26 ↓7**	**↑29 ↓4**	**↑29 ↓6**	**↑29 ↓6**	**↑30 ↓5**	**↑30 ↓5**	**↑26 ↓6**	**↑206 ↓49**
T_2	(35)	↑*6* ↓*14*	**↑15 ↓12**	**↑22 ↓5**	**↑30 ↓3**	**↑30 ↓4**	**↑29 ↓6**	**↑29 ↓6**	**↑27 ↓6**	**↑188 ↓56**
T_3	(35)	↑*3* ↓*14*	↑*7* ↓*13*	**↑19 ↓8**	**↑29 ↓4**	**↑28 ↓5**	**↑24 ↓7**	**↑23 ↓9**	**↑23 ↓9**	**↑156 ↓69**
T_4	(35)	**↑25 ↓2**	↑*11* ↓*20*	**↑19 ↓7**	**↑28 ↓6**	**↑28 ↓6**	**↑28 ↓6**	**↑28 ↓6**	**↑28 ↓7**	**↑195 ↓60**
T_5	(35)	**↑14 ↓0**	↑*4* ↓*24*	↑*13* ↓*19*	↑16 ↓16	**↑17 ↓16**	**↑18 ↓11**	**↑22 ↓11**	**↑21 ↓10**	**↑125 ↓107**
T_6	(35)	**↑23 ↓5**	↑*10* ↓*12*	**↑21 ↓9**	**↑27 ↓4**	**↑25 ↓6**	**↑22 ↓9**	**↑22 ↓12**	**↑22 ↓13**	**↑172 ↓70**
All	(270)	↑*78* ↓*98*	↑*74* ↓*146*	**↑130 ↓100**	**↑196 ↓57**	**↑199 ↓57**	**↑196 ↓55**	**↑202 ↓59**	**↑198 ↓59**	**↑1273 ↓631**

Appendix C

Additional Results - Chapter 5

This appendix contains results that were omitted from Chapter 5 due to space constraints. Table C.1 and gives the performance analysis of CDE compared to DynDE on the n_p standard set of environments. CDE and DynDE used equal numbers of sub-populations as numbers of peaks. Table C.2 gives the performance analysis of DynDE10 compared to DynDE on the n_p standard set of environments. The comparative performance analysis on variations of the standard set of DynPopDE versus DynDE is given in Tables C.3 and C.4.

Table C.1: CDE vs DynDE performance analysis on the n_p standard set

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑4 ↓0**
10	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑9 ↓0**
25	(2)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
50	(2)	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓1**
100	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓0**
200	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓0**
C	(6)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑30 ↓0**
S	(6)	↑0 ↓0	↑0 ↓0	↑1 ↓1	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑25 ↓1**
All	(12)	↑0 ↓0	**↑2 ↓0**	**↑3 ↓1**	**↑9 ↓0**	**↑11 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑55 ↓1**
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	*↑1 ↓3*
10	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑2 ↓4*
25	(2)	**↑1 ↓0**	↑0 ↓0	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑5 ↓5
50	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑4 ↓5*
100	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑5 ↓5
200	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑4 ↓5*
C	(6)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑3 ↓0**	**↑2 ↓1**	**↑20 ↓2**
S	(6)	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓3*	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓4*	*↑1 ↓25*
All	(12)	**↑2 ↓0**	↑1 ↓1	*↑2 ↓3*	↑3 ↓3	↑4 ↓4	*↑4 ↓5*	*↑3 ↓6*	*↑2 ↓5*	*↑21 ↓27*
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑7 ↓0**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑7 ↓0**
25	(2)	↑0 ↓0	↑0 ↓0	↑1 ↓1	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑7 ↓1**
50	(2)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑6 ↓1**
100	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑5 ↓4**
200	(2)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	*↑4 ↓7*
C	(6)	↑0 ↓0	**↑2 ↓1**	**↑3 ↓0**	**↑5 ↓1**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑30 ↓2**
S	(6)	**↑1 ↓0**	↑0 ↓0	*↑1 ↓2*	*↑1 ↓2*	↑2 ↓2	*↑0 ↓3*	*↑0 ↓1*	↑1 ↓1	*↑6 ↓11*
All	(12)	**↑1 ↓0**	**↑2 ↓1**	**↑4 ↓2**	**↑6 ↓3**	**↑8 ↓2**	**↑5 ↓3**	**↑5 ↓1**	**↑5 ↓1**	**↑36 ↓13**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
10	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
25	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
50	(2)	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
100	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑9 ↓0**
200	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑6 ↓3**
C	(6)	↑0 ↓0	**↑3 ↓0**	**↑3 ↓0**	**↑5 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑33 ↓1**
S	(6)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑4 ↓1**	**↑5 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑29 ↓2**
All	(12)	↑0 ↓0	**↑4 ↓0**	**↑5 ↓0**	**↑9 ↓2**	**↑11 ↓1**	**↑11 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑62 ↓3**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	**↑6 ↓0**
10	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
25	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓0**
50	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓0**
100	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑10 ↓0**
200	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓2*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑7 ↓2**
C	(6)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑5 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑32 ↓1**
S	(6)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	**↑3 ↓1**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑25 ↓2**
All	(12)	↑1 ↓1	**↑3 ↓0**	**↑4 ↓0**	**↑8 ↓2**	**↑10 ↓0**	**↑11 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑57 ↓3**
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	**↑2 ↓0**	**↑4 ↓0**	**↑6 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓1**	↑1 ↓1	**↑2 ↓0**	**↑30 ↓3**
10	(10)	*↑0 ↓1*	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓0**	**↑8 ↓0**	**↑6 ↓0**	**↑7 ↓1**	**↑6 ↓1**	**↑44 ↓5**
25	(10)	**↑1 ↓0**	↑0 ↓0	**↑6 ↓2**	**↑9 ↓0**	**↑8 ↓1**	**↑8 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑46 ↓6**
50	(10)	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓2**	**↑8 ↓1**	**↑7 ↓1**	**↑41 ↓7**
100	(10)	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑7 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓1**	**↑39 ↓9**
200	(10)	↑0 ↓0	↑1 ↓1	*↑0 ↓1*	*↑1 ↓7*	**↑6 ↓3**	**↑7 ↓2**	**↑8 ↓1**	**↑8 ↓2**	**↑31 ↓17**
C	(30)	**↑2 ↓0**	**↑10 ↓1**	**↑12 ↓0**	**↑23 ↓3**	**↑28 ↓0**	**↑26 ↓1**	**↑23 ↓0**	**↑21 ↓1**	**↑145 ↓6**
S	(30)	**↑2 ↓1**	**↑2 ↓1**	↑6 ↓6	**↑12 ↓7**	**↑16 ↓7**	**↑15 ↓7**	**↑16 ↓7**	**↑17 ↓5**	**↑86 ↓41**
All	(60)	**↑4 ↓1**	**↑12 ↓2**	**↑18 ↓6**	**↑35 ↓10**	**↑44 ↓7**	**↑41 ↓8**	**↑39 ↓7**	**↑38 ↓6**	**↑231 ↓47**

Table C.2: DynDE10 vs DynDE performance analysis on the n_p standard set

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	*↑2 ↓14*
10	(2)	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑3 ↓1**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑5 ↓10*
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑6 ↓10*
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑6 ↓9*
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	**↑8 ↓7**
C	(6)	**↑4 ↓1**	**↑4 ↓2**	**↑4 ↓1**	*↑1 ↓3*	*↑1 ↓4*	*↑0 ↓5*	*↑1 ↓4*	*↑1 ↓4*	*↑16 ↓24*
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	*↑1 ↓4*	*↑1 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑14 ↓27*
All	(12)	**↑8 ↓2**	**↑8 ↓3**	**↑8 ↓2**	*↑2 ↓7*	*↑2 ↓9*	*↑0 ↓10*	*↑1 ↓9*	*↑1 ↓9*	*↑30 ↓51*
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	↑1 ↓1	*↑0 ↓2*	*↑1 ↓14*
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	**↑7 ↓7**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	**↑9 ↓6**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	*↑0 ↓2*	*↑0 ↓2*	**↑11 ↓5**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	**↑11 ↓4**
C	(6)	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	↑2 ↓2	*↑0 ↓5*	*↑1 ↓4*	*↑0 ↓5*	*↑18 ↓19*
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓1**	*↑2 ↓3*	*↑0 ↓4*	*↑0 ↓5*	**↑21 ↓17**
All	(12)	**↑7 ↓1**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑5 ↓3**	*↑2 ↓8*	*↑1 ↓8*	*↑0 ↓10*	**↑39 ↓36**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓14*
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑13 ↓2**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑15 ↓1**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑15 ↓1**
200	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
C	(6)	**↑3 ↓0**	**↑3 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓2**	*↑1 ↓3*	**↑26 ↓10**
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑32 ↓8**
All	(12)	**↑7 ↓1**	**↑7 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑7 ↓3**	**↑5 ↓4**	**↑58 ↓18**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓15*
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑3 ↓0**	**↑3 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓1**	**↑29 ↓7**
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑32 ↓8**
All	(12)	**↑7 ↓1**	**↑7 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑7 ↓2**	**↑61 ↓15**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	↑1 ↓1	*↑0 ↓2*	↑1 ↓1	*↑0 ↓2*	*↑2 ↓11*
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑4 ↓0**	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑31 ↓6**
S	(6)	**↑4 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑5 ↓0**	**↑4 ↓1**	**↑5 ↓0**	**↑4 ↓1**	**↑34 ↓5**
All	(12)	**↑8 ↓0**	**↑7 ↓1**	**↑8 ↓2**	**↑8 ↓2**	**↑9 ↓1**	**↑8 ↓2**	**↑9 ↓1**	**↑8 ↓2**	**↑65 ↓11**
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	*↑0 ↓5*	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓10*	*↑1 ↓9*	*↑0 ↓10*	*↑3 ↓7*	*↑1 ↓8*	*↑5 ↓68*
10	(10)	**↑1 ↓0**	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑3 ↓1**
25	(10)	**↑6 ↓0**	**↑8 ↓0**	**↑10 ↓0**	**↑8 ↓2**	**↑6 ↓3**	**↑6 ↓4**	↑5 ↓5	*↑4 ↓5*	**↑53 ↓19**
50	(10)	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓2**	**↑7 ↓2**	**↑6 ↓4**	**↑6 ↓4**	↑5 ↓5	**↑62 ↓17**
100	(10)	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓1**	**↑8 ↓2**	**↑7 ↓3**	**↑6 ↓4**	↑5 ↓5	**↑64 ↓15**
200	(10)	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓1**	**↑7 ↓3**	**↑6 ↓3**	**↑6 ↓4**	**↑66 ↓11**
C	(30)	**↑17 ↓1**	**↑17 ↓5**	**↑20 ↓5**	**↑17 ↓7**	**↑15 ↓9**	*↑12 ↓13*	**↑13 ↓12**	*↑9 ↓14*	**↑120 ↓66**
S	(30)	**↑20 ↓4**	**↑20 ↓5**	**↑20 ↓5**	**↑17 ↓8**	**↑17 ↓8**	**↑14 ↓11**	**↑13 ↓11**	*↑12 ↓13*	**↑133 ↓65**
All	(60)	**↑37 ↓5**	**↑37 ↓10**	**↑40 ↓10**	**↑34 ↓15**	**↑32 ↓17**	**↑26 ↓24**	**↑26 ↓23**	*↑21 ↓27*	**↑253 ↓131**

Table C.3: DynPopDE vs DynDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	**↑9 ↓0**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
20	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓1**
40	(2)	*↑0 ↓2*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑10 ↓3**
80	(2)	*↑0 ↓2*	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑7 ↓3**
C	(6)	*↑1 ↓2*	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑2 ↓0**	**↑2 ↓1**	**↑29 ↓3**
S	(6)	*↑1 ↓3*	**↑4 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑36 ↓4**
		GBG								
F_{1a}	(6)	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑2 ↓3*	*↑0 ↓6*	*↑0 ↓6*	**↑28 ↓15**
F_{1b}	(6)	**↑3 ↓1**	**↑6 ↓0**	**↑5 ↓1**	↑3 ↓3	*↑1 ↓3*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑18 ↓22*
F_2	(6)	**↑3 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓1**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	**↑23 ↓19**
F_3	(6)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	**↑2 ↓1**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑23 ↓3**
F_4	(6)	**↑3 ↓0**	**↑3 ↓0**	**↑5 ↓1**	**↑4 ↓2**	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑16 ↓23*
F_5	(6)	**↑3 ↓0**	**↑5 ↓0**	**↑4 ↓1**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑12 ↓31*
F_6	(6)	*↑0 ↓1*	**↑2 ↓0**	↑1 ↓1	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑3 ↓32*
T_1	(7)	**↑5 ↓0**	**↑4 ↓0**	*↑1 ↓5*	*↑1 ↓6*	*↑1 ↓5*	*↑0 ↓5*	*↑1 ↓5*	*↑1 ↓5*	*↑14 ↓31*
T_2	(7)	**↑2 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓2**	**↑3 ↓2**	*↑2 ↓5*	*↑1 ↓6*	*↑1 ↓6*	**↑22 ↓21**
T_3	(7)	*↑0 ↓2*	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓2**	**↑4 ↓2**	*↑2 ↓4*	*↑1 ↓6*	*↑1 ↓6*	**↑23 ↓22**
T_4	(7)	**↑5 ↓0**	**↑6 ↓1**	**↑4 ↓0**	*↑3 ↓4*	*↑2 ↓4*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑23 ↓27*
T_5	(7)	**↑1 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓2**	**↑3 ↓2**	*↑1 ↓3*	*↑1 ↓6*	*↑1 ↓6*	**↑20 ↓19**
T_6	(7)	**↑1 ↓0**	**↑5 ↓0**	**↑6 ↓0**	↑3 ↓3	*↑3 ↓4*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑21 ↓25*
All	(54)	**↑16 ↓7**	**↑36 ↓2**	**↑36 ↓5**	**↑30 ↓19**	**↑27 ↓19**	*↑17 ↓29*	*↑13 ↓35*	*↑13 ↓36*	**↑188 ↓152**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
5	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
40	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
80	(2)	*↑0 ↓1*	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓1**
C	(6)	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑34 ↓0**
S	(6)	*↑1 ↓2*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓2**
		GBG								
F_{1a}	(6)	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	*↑0 ↓5*	*↑0 ↓6*	**↑32 ↓12**
F_{1b}	(6)	**↑2 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓2**	**↑4 ↓2**	*↑2 ↓3*	*↑0 ↓6*	*↑0 ↓6*	**↑23 ↓19**
F_2	(6)	**↑3 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓1**	**↑3 ↓2**	*↑2 ↓4*	*↑0 ↓6*	*↑0 ↓6*	**↑22 ↓20**
F_3	(6)	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓4*	*↑0 ↓3*	**↑2 ↓1**	*↑2 ↓29*
F_4	(6)	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓1**	**↑4 ↓1**	**↑4 ↓2**	*↑2 ↓4*	*↑0 ↓5*	*↑0 ↓6*	**↑23 ↓19**
F_5	(6)	↑1 ↓1	*↑2 ↓3*	*↑1 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑4 ↓38*
F_6	(6)	**↑1 ↓0**	↑0 ↓0	*↑1 ↓3*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑2 ↓32*
T_1	(7)	**↑6 ↓1**	**↑4 ↓2**	*↑1 ↓5*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑0 ↓7*	*↑0 ↓6*	*↑14 ↓39*
T_2	(7)	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓3**	**↑4 ↓3**	↑3 ↓3	*↑1 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑19 ↓29*
T_3	(7)	*↑0 ↓2*	**↑5 ↓0**	**↑4 ↓2**	**↑4 ↓3**	**↑4 ↓3**	**↑4 ↓3**	*↑0 ↓7*	*↑0 ↓6*	*↑21 ↓26*
T_4	(7)	**↑3 ↓0**	**↑4 ↓2**	**↑4 ↓3**	*↑1 ↓4*	*↑1 ↓6*	*↑0 ↓7*	*↑0 ↓6*	*↑1 ↓6*	*↑14 ↓34*
T_5	(7)	**↑2 ↓0**	**↑3 ↓0**	**↑6 ↓1**	**↑4 ↓3**	**↑4 ↓3**	**↑4 ↓2**	*↑0 ↓4*	*↑0 ↓5*	**↑23 ↓18**
T_6	(7)	↑0 ↓0	**↑3 ↓0**	**↑4 ↓0**	**↑4 ↓3**	**↑4 ↓3**	*↑1 ↓5*	*↑0 ↓6*	*↑1 ↓6*	*↑17 ↓23*
All	(54)	**↑17 ↓5**	**↑32 ↓5**	**↑35 ↓14**	**↑30 ↓22**	**↑28 ↓24**	*↑21 ↓28*	*↑9 ↓37*	*↑11 ↓36*	**↑183 ↓171**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑13 ↓0**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓0**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
40	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	↑1 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
80	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑11 ↓1**
C	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑33 ↓0**
S	(6)	**↑2 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑41 ↓1**
		GBG								
F_{1a}	(6)	**↑2 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	↑2 ↓2	**↑36 ↓3**
F_{1b}	(6)	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑3 ↓2**	*↑1 ↓3*	**↑33 ↓8**
F_2	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓2**	*↑2 ↓3*	*↑0 ↓5*	*↑1 ↓4*	*↑0 ↓4*	**↑22 ↓18**
F_3	(6)	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓44*
F_4	(6)	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓0**	↑2 ↓2	*↑2 ↓4*	*↑1 ↓5*	*↑0 ↓4*	*↑0 ↓4*	↑19 ↓19
F_5	(6)	*↑1 ↓2*	*↑1 ↓3*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑2 ↓39*
F_6	(6)	↑0 ↓0	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓5*	*↑1 ↓36*
T_1	(7)	**↑4 ↓1**	**↑4 ↓3**	↑3 ↓3	*↑1 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑1 ↓5*	*↑19 ↓32*
T_2	(7)	**↑3 ↓1**	**↑4 ↓2**	**↑4 ↓3**	*↑2 ↓3*	*↑2 ↓5*	*↑2 ↓5*	*↑1 ↓5*	*↑0 ↓6*	*↑18 ↓30*
T_3	(7)	**↑2 ↓0**	**↑3 ↓2**	*↑2 ↓3*	**↑4 ↓3**	**↑4 ↓3**	*↑2 ↓5*	*↑2 ↓3*	*↑1 ↓2*	*↑20 ↓21*
T_4	(7)	**↑4 ↓1**	**↑4 ↓2**	**↑3 ↓2**	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑14 ↓37*
T_5	(7)	**↑3 ↓2**	**↑5 ↓0**	**↑3 ↓1**	**↑4 ↓3**	**↑4 ↓3**	↑3 ↓3	↑3 ↓3	*↑2 ↓3*	**↑27 ↓18**
T_6	(7)	**↑2 ↓0**	**↑3 ↓2**	**↑3 ↓2**	↑3 ↓3	*↑2 ↓4*	*↑2 ↓5*	*↑0 ↓6*	*↑0 ↓7*	*↑15 ↓29*
All	(54)	**↑20 ↓6**	**↑33 ↓11**	**↑30 ↓14**	**↑26 ↓23**	**↑27 ↓26**	*↑21 ↓29*	*↑18 ↓29*	*↑12 ↓30*	**↑187 ↓168**

Table C.4: DynPopDE vs DynDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑13 ↓1**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
20	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
80	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑9 ↓3**
C	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑2 ↓1**	**↑34 ↓1**
S	(6)	**↑2 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓0**	**↑38 ↓3**
		GBG								
F_{1a}	(6)	**↑2 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑40 ↓1**
F_{1b}	(6)	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑3 ↓2**	**↑37 ↓6**
F_2	(6)	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓1**	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑15 ↓26*
F_3	(6)	*↑0 ↓3*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓43*
F_4	(6)	**↑6 ↓0**	**↑5 ↓0**	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑13 ↓24*
F_5	(6)	**↑2 ↓0**	**↑4 ↓0**	**↑2 ↓1**	*↑1 ↓3*	*↑1 ↓3*	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓5*	*↑10 ↓20*
F_6	(6)	↑0 ↓0	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓41*
T_1	(7)	**↑3 ↓0**	**↑4 ↓2**	**↑4 ↓2**	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑21 ↓29*
T_2	(7)	**↑3 ↓1**	**↑4 ↓2**	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑19 ↓30*
T_3	(7)	**↑3 ↓0**	**↑3 ↓2**	**↑3 ↓2**	*↑2 ↓4*	*↑2 ↓5*	*↑2 ↓5*	*↑2 ↓5*	*↑1 ↓5*	*↑18 ↓28*
T_4	(7)	**↑4 ↓1**	**↑5 ↓2**	↑3 ↓3	*↑1 ↓5*	*↑1 ↓5*	*↑1 ↓6*	*↑1 ↓6*	*↑0 ↓7*	*↑16 ↓35*
T_5	(7)	**↑4 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑3 ↓2**	*↑3 ↓4*	↑2 ↓2	*↑2 ↓3*	*↑2 ↓3*	**↑26 ↓18**
T_6	(7)	**↑2 ↓0**	**↑2 ↓0**	**↑3 ↓2**	↑2 ↓2	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑0 ↓5*	*↑15 ↓21*
All	(54)	**↑21 ↓4**	**↑33 ↓9**	**↑32 ↓14**	↑22 ↓22	*↑24 ↓28*	*↑22 ↓28*	*↑19 ↓29*	*↑14 ↓31*	**↑187 ↓165**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓7
5	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	↑6 ↓6
10	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑6 ↓7*
20	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑6 ↓7*
40	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑5 ↓7*
80	(2)	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑6 ↓7*
C	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑2 ↓0**	**↑36 ↓0**
S	(6)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓41*
		GBG								
F_{1a}	(6)	**↑2 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓2**	*↑1 ↓4*	*↑0 ↓4*	*↑0 ↓6*	**↑20 ↓17**
F_{1b}	(6)	**↑3 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓2**	*↑1 ↓4*	*↑0 ↓6*	*↑0 ↓6*	**↑21 ↓19**
F_2	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑2 ↓1**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑14 ↓31*
F_3	(6)	*↑0 ↓2*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓42*
F_4	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑1 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑13 ↓30*
F_5	(6)	**↑2 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑2 ↓0**	↑1 ↓1	*↑1 ↓3*	*↑1 ↓3*	**↑18 ↓8**
F_6	(6)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓42*
T_1	(7)	**↑3 ↓0**	**↑4 ↓2**	**↑3 ↓2**	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑10 ↓35*
T_2	(7)	**↑4 ↓0**	**↑4 ↓2**	↑3 ↓3	*↑3 ↓4*	*↑3 ↓4*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑17 ↓33*
T_3	(7)	**↑4 ↓1**	**↑3 ↓2**	*↑1 ↓2*	*↑3 ↓4*	*↑3 ↓4*	*↑1 ↓4*	*↑1 ↓5*	*↑0 ↓6*	*↑16 ↓28*
T_4	(7)	**↑4 ↓1**	**↑5 ↓2**	**↑4 ↓2**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑13 ↓37*
T_5	(7)	**↑2 ↓1**	**↑4 ↓2**	**↑4 ↓2**	*↑3 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑0 ↓5*	*↑1 ↓6*	*↑18 ↓28*
T_6	(7)	**↑2 ↓0**	**↑4 ↓1**	**↑2 ↓1**	*↑2 ↓4*	*↑2 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑12 ↓28*
All	(54)	**↑19 ↓3**	**↑29 ↓17**	**↑23 ↓18**	*↑17 ↓32*	*↑16 ↓33*	*↑9 ↓39*	*↑6 ↓43*	*↑3 ↓45*	*↑122 ↓230*
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑6 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑8 ↓1**	**↑5 ↓1**	**↑4 ↓1**	**↑55 ↓7**
5	(10)	**↑2 ↓0**	**↑8 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓0**	**↑7 ↓1**	**↑6 ↓1**	**↑5 ↓2**	**↑55 ↓7**
10	(10)	**↑1 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑7 ↓1**	**↑5 ↓1**	**↑57 ↓7**
20	(10)	*↑0 ↓1*	**↑7 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑6 ↓1**	**↑56 ↓8**
40	(10)	*↑0 ↓3*	**↑7 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑7 ↓1**	**↑7 ↓2**	**↑54 ↓11**
80	(10)	*↑0 ↓5*	↑2 ↓2	**↑7 ↓1**	**↑8 ↓1**	**↑9 ↓1**	**↑6 ↓2**	**↑7 ↓2**	**↑6 ↓1**	**↑45 ↓15**
C	(30)	**↑3 ↓2**	**↑23 ↓0**	**↑29 ↓0**	**↑29 ↓0**	**↑29 ↓0**	**↑24 ↓0**	**↑17 ↓0**	**↑12 ↓2**	**↑166 ↓4**
S	(30)	*↑6 ↓7*	**↑19 ↓7**	**↑23 ↓6**	**↑20 ↓6**	**↑23 ↓5**	**↑22 ↓7**	**↑22 ↓7**	**↑21 ↓6**	**↑156 ↓51**
		GBG								
F_{1a}	(30)	**↑11 ↓0**	**↑25 ↓0**	**↑28 ↓0**	**↑28 ↓1**	**↑28 ↓2**	**↑20 ↓8**	*↑10 ↓16*	*↑6 ↓21*	**↑156 ↓48**
F_{1b}	(30)	**↑14 ↓1**	**↑27 ↓0**	**↑27 ↓1**	**↑20 ↓8**	**↑19 ↓9**	↑13 ↓13	*↑8 ↓20*	*↑4 ↓22*	**↑132 ↓74**
F_2	(30)	**↑24 ↓0**	**↑28 ↓0**	**↑20 ↓3**	*↑12 ↓14*	*↑9 ↓18*	*↑2 ↓25*	*↑1 ↓27*	*↑0 ↓27*	*↑96 ↓114*
F_3	(30)	*↑0 ↓10*	*↑0 ↓17*	*↑0 ↓23*	*↑2 ↓25*	*↑4 ↓24*	*↑5 ↓22*	*↑6 ↓21*	*↑8 ↓19*	*↑25 ↓161*
F_4	(30)	**↑25 ↓0**	**↑23 ↓0**	**↑16 ↓2**	*↑10 ↓13*	*↑7 ↓21*	*↑3 ↓25*	*↑0 ↓27*	*↑0 ↓27*	*↑84 ↓115*
F_5	(30)	**↑9 ↓4**	**↑16 ↓6**	**↑11 ↓10**	*↑4 ↓21*	*↑3 ↓21*	*↑1 ↓23*	*↑1 ↓25*	*↑1 ↓26*	*↑46 ↓136*
F_6	(30)	↑1 ↓1	*↑2 ↓14*	*↑2 ↓20*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑1 ↓28*	*↑6 ↓183*
T_1	(35)	**↑21 ↓2**	**↑20 ↓9**	*↑12 ↓17*	*↑5 ↓26*	*↑6 ↓27*	*↑5 ↓28*	*↑5 ↓29*	*↑4 ↓28*	*↑78 ↓166*
T_2	(35)	**↑15 ↓2**	**↑20 ↓7**	**↑18 ↓12**	*↑15 ↓16*	*↑13 ↓19*	*↑7 ↓26*	*↑4 ↓30*	*↑3 ↓31*	*↑95 ↓143*
T_3	(35)	**↑9 ↓5**	**↑19 ↓6**	**↑15 ↓9**	**↑18 ↓16**	↑17 ↓17	*↑11 ↓21*	*↑6 ↓26*	*↑3 ↓25*	*↑98 ↓125*
T_4	(35)	**↑20 ↓3**	**↑24 ↓9**	**↑18 ↓10**	*↑6 ↓25*	*↑5 ↓27*	*↑3 ↓31*	*↑2 ↓32*	*↑2 ↓33*	*↑80 ↓170*
T_5	(35)	**↑12 ↓4**	**↑21 ↓3**	**↑23 ↓6**	**↑18 ↓14**	↑16 ↓16	*↑12 ↓14*	*↑6 ↓21*	*↑6 ↓23*	**↑114 ↓101**
T_6	(35)	**↑7 ↓0**	**↑17 ↓3**	**↑18 ↓5**	*↑14 ↓15*	*↑13 ↓19*	*↑6 ↓26*	*↑3 ↓28*	*↑2 ↓30*	*↑80 ↓126*
All	(270)	**↑93 ↓25**	**↑163 ↓44**	**↑156 ↓65**	**↑125 ↓118**	*↑122 ↓130*	*↑90 ↓153*	*↑65 ↓173*	*↑53 ↓178*	*↑867 ↓886*

Appendix D

Additional Results - Chapter 6

This appendix contains results that were omitted from Chapter 6 due to space constraints. Tables D.1 and D.2 give the performance analysis of jSA2Ran compared to DynDE on variations of the standard set of environments. Tables D.3 and D.4 give the performance analysis of SABrNorRes compared to DynDE on variations of the standard set of environments. The comparative performance analysis on variations of the standard set of SACDE versus DynDE is given in Tables D.5 and D.6. Tables D.7 and D.8 give the performance analysis of SADynPopDE compared to CDE on variations of the standard set of environments. Tables D.9 and D.10 give the performance analysis of SADynPopDE compared to SACDE on variations of the standard set of environments. The comparative performance analysis on the n_p standard set of DynPopDE versus SACDE is given in Table D.11. Table D.12 gives the performance analysis of SACDE compared to CDE on the $n_p(t)$ standard set of environments. The comparative performance analysis on the $n_p(t)$ standard set of SADynPopDE versus SACDE is given in Table D.13.

Table D.1: jSA2Ran vs DynDE performance analysis - Part 1

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	**↑8 ↓0**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
C	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑2 ↓0**	**↑32 ↓0**
S	(6)	**↑1 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑39 ↓0**
		GBG								
F_{1a}	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	*↑1 ↓4*	*↑0 ↓6*	*↑0 ↓6*	**↑25 ↓17**
F_{1b}	(6)	↑1 ↓1	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	**↑24 ↓19**
F_2	(6)	↑0 ↓0	**↑3 ↓0**	**↑6 ↓0**	**↑4 ↓2**	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑13 ↓24*
F_3	(6)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑20 ↓0**
F_4	(6)	*↑0 ↓1*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓2**	*↑0 ↓6*	*↑0 ↓6*	**↑22 ↓15**
F_5	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓2**	*↑1 ↓5*	*↑0 ↓6*	**↑28 ↓13**
F_6	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑4 ↓1**	*↑1 ↓3*	**↑32 ↓5**
T_1	(7)	↑0 ↓0	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑3 ↓2**	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	**↑26 ↓20**
T_2	(7)	*↑0 ↓1*	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓1**	**↑4 ↓2**	*↑1 ↓5*	*↑0 ↓6*	**↑31 ↓15**
T_3	(7)	*↑0 ↓1*	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑5 ↓1**	**↑3 ↓1**	*↑2 ↓5*	*↑1 ↓5*	**↑29 ↓14**
T_4	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	*↑0 ↓5*	*↑2 ↓5*	*↑1 ↓6*	**↑27 ↓17**
T_5	(7)	**↑1 ↓0**	**↑3 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	*↑2 ↓3*	*↑2 ↓4*	*↑0 ↓5*	**↑23 ↓14**
T_6	(7)	↑0 ↓0	**↑4 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	↑3 ↓3	*↑1 ↓5*	*↑2 ↓5*	**↑28 ↓13**
All	(54)	*↑2 ↓3*	**↑42 ↓0**	**↑51 ↓0**	**↑48 ↓2**	**↑40 ↓5**	**↑22 ↓20**	*↑17 ↓30*	*↑13 ↓33*	**↑235 ↓93**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	**↑3 ↓2**
5	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	**↑8 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
20	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
80	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
C	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓1**	**↑35 ↓1**
S	(6)	**↑2 ↓0**	**↑4 ↓0**	**↑5 ↓1**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑33 ↓1**
		GBG								
F_{1a}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	*↑0 ↓4*	*↑0 ↓6*	**↑28 ↓10**
F_{1b}	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓1**	*↑0 ↓5*	*↑0 ↓6*	**↑27 ↓12**
F_2	(6)	*↑0 ↓1*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	↑3 ↓3	↑3 ↓3	*↑0 ↓4*	*↑0 ↓6*	**↑20 ↓17**
F_3	(6)	↑0 ↓0	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑25 ↓0**
F_4	(6)	↑0 ↓0	**↑2 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑3 ↓1**	↑3 ↓3	*↑1 ↓4*	*↑0 ↓6*	**↑19 ↓14**
F_5	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑40 ↓1**
F_6	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑41 ↓1**
T_1	(7)	*↑0 ↓1*	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	*↑3 ↓4*	*↑1 ↓6*	**↑38 ↓12**
T_2	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓1**	**↑4 ↓2**	*↑3 ↓4*	*↑2 ↓4*	**↑33 ↓11**
T_3	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑4 ↓1**	**↑3 ↓2**	↑2 ↓2	*↑3 ↓4*	**↑31 ↓9**
T_4	(7)	↑0 ↓0	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓2**	*↑1 ↓4*	**↑35 ↓6**
T_5	(7)	↑0 ↓0	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓2**	**↑4 ↓2**	*↑2 ↓3*	*↑3 ↓4*	**↑27 ↓11**
T_6	(7)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	*↑2 ↓4*	**↑36 ↓6**
All	(54)	**↑2 ↓1**	**↑43 ↓0**	**↑49 ↓1**	**↑51 ↓0**	**↑42 ↓4**	**↑38 ↓7**	**↑24 ↓17**	*↑19 ↓27*	**↑268 ↓57**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
5	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓1**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
80	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
C	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑40 ↓0**
S	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓1**
		GBG								
F_{1a}	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓2**	**↑38 ↓3**
F_{1b}	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑2 ↓1**	**↑37 ↓2**
F_2	(6)	*↑0 ↓1*	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑34 ↓1**
F_3	(6)	*↑0 ↓2*	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑16 ↓2**
F_4	(6)	*↑0 ↓1*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑32 ↓2**
F_5	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓1**
F_6	(6)	↑0 ↓0	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓0**
T_1	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑45 ↓1**
T_2	(7)	↑0 ↓0	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑2 ↓1**	**↑38 ↓1**
T_3	(7)	*↑0 ↓1*	**↑4 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑40 ↓1**
T_4	(7)	*↑0 ↓1*	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑40 ↓2**
T_5	(7)	*↑0 ↓3*	**↑5 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑36 ↓3**
T_6	(7)	*↑0 ↓1*	**↑3 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓2**	**↑39 ↓3**
All	(54)	*↑0 ↓8*	**↑39 ↓0**	**↑49 ↓0**	**↑54 ↓0**	**↑52 ↓0**	**↑47 ↓0**	**↑44 ↓0**	**↑35 ↓4**	**↑320 ↓12**

Table D.2: jSA2Ran vs DynDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
80	(2)	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
C	(6)	↑0 ↓0	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑39 ↓0**
S	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓0**
		GBG								
F_{1a}	(6)	*↑0 ↓1*	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓1**
F_{1b}	(6)	*↑0 ↓2*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓2**
F_2	(6)	*↑0 ↓2*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓2**
F_3	(6)	*↑0 ↓3*	↑1 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	↑0 ↓0	↑0 ↓0	**↑17 ↓5**
F_4	(6)	*↑0 ↓2*	**↑4 ↓0**	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓2**
F_5	(6)	**↑2 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓0**
F_6	(6)	*↑0 ↓2*	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓2**
T_1	(7)	*↑0 ↓2*	**↑6 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑45 ↓3**
T_2	(7)	**↑1 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓0**
T_3	(7)	↑0 ↓0	**↑2 ↓1**	**↑3 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓1**
T_4	(7)	*↑1 ↓5*	**↑4 ↓1**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑44 ↓6**
T_5	(7)	*↑0 ↓3*	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑46 ↓3**
T_6	(7)	*↑0 ↓1*	**↑3 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓1**
All	(54)	*↑2 ↓11*	**↑36 ↓3**	**↑42 ↓0**	**↑54 ↓0**	**↑53 ↓0**	**↑53 ↓0**	**↑48 ↓0**	**↑46 ↓0**	**↑334 ↓14**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑10 ↓4**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑10 ↓4**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑11 ↓3**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑12 ↓2**
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑11 ↓3**
80	(2)	*↑0 ↓1*	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
C	(6)	*↑0 ↓1*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓1**
S	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	↑3 ↓3	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	**↑26 ↓16**
		GBG								
F_{1a}	(6)	↑0 ↓0	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑39 ↓0**
F_{1b}	(6)	↑0 ↓0	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓0**
F_2	(6)	*↑0 ↓1*	**↑3 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓1**
F_3	(6)	↑0 ↓0	*↑0 ↓2*	**↑1 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑28 ↓2**
F_4	(6)	↑0 ↓0	**↑3 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑36 ↓0**
F_5	(6)	**↑2 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓0**
F_6	(6)	↑1 ↓1	**↑3 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓0**	**↑36 ↓2**
T_1	(7)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑47 ↓1**
T_2	(7)	↑0 ↓0	**↑3 ↓1**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑45 ↓1**
T_3	(7)	**↑1 ↓0**	↑1 ↓1	**↑3 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑39 ↓1**
T_4	(7)	↑1 ↓1	**↑4 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑45 ↓1**
T_5	(7)	**↑1 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓1**	**↑7 ↓0**	**↑44 ↓1**
T_6	(7)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑35 ↓0**
All	(54)	↑3 ↓3	**↑31 ↓2**	**↑40 ↓0**	**↑51 ↓1**	**↑51 ↓3**	**↑49 ↓4**	**↑48 ↓5**	**↑49 ↓4**	**↑322 ↓22**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑0 ↓0	**↑8 ↓0**	**↑8 ↓1**	**↑9 ↓0**	**↑7 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑47 ↓6**
5	(10)	*↑0 ↓1*	**↑9 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑5 ↓1**	**↑58 ↓5**
10	(10)	**↑1 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑10 ↓0**	**↑8 ↓0**	**↑66 ↓3**
20	(10)	**↑1 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑8 ↓1**	**↑68 ↓2**
40	(10)	↑0 ↓0	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑65 ↓3**
80	(10)	↑1 ↓1	**↑7 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑65 ↓1**
C	(30)	*↑0 ↓1*	**↑27 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑27 ↓0**	**↑25 ↓0**	**↑19 ↓1**	**↑187 ↓2**
S	(30)	**↑3 ↓1**	**↑27 ↓0**	**↑28 ↓1**	**↑28 ↓1**	**↑25 ↓3**	**↑24 ↓4**	**↑23 ↓4**	**↑24 ↓4**	**↑182 ↓18**
		GBG								
F_{1a}	(30)	*↑0 ↓3*	**↑25 ↓0**	**↑28 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑23 ↓4**	**↑18 ↓10**	**↑15 ↓14**	**↑169 ↓31**
F_{1b}	(30)	*↑1 ↓4*	**↑26 ↓0**	**↑29 ↓0**	**↑30 ↓0**	**↑29 ↓1**	**↑21 ↓6**	**↑17 ↓11**	**↑14 ↓13**	**↑167 ↓35**
F_2	(30)	*↑0 ↓5*	**↑16 ↓0**	**↑24 ↓0**	**↑28 ↓2**	**↑21 ↓7**	**↑21 ↓9**	**↑16 ↓10**	**↑16 ↓12**	**↑142 ↓45**
F_3	(30)	*↑0 ↓4*	**↑7 ↓5**	**↑15 ↓0**	**↑24 ↓0**	**↑19 ↓0**	**↑15 ↓0**	**↑13 ↓0**	**↑13 ↓0**	**↑106 ↓9**
F_4	(30)	*↑0 ↓4*	**↑15 ↓0**	**↑22 ↓0**	**↑28 ↓0**	**↑26 ↓1**	**↑22 ↓5**	**↑17 ↓10**	**↑16 ↓13**	**↑146 ↓33**
F_5	(30)	**↑4 ↓1**	**↑26 ↓0**	**↑29 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑28 ↓2**	**↑25 ↓5**	**↑22 ↓7**	**↑193 ↓15**
F_6	(30)	*↑1 ↓3*	**↑22 ↓0**	**↑26 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑28 ↓1**	**↑27 ↓2**	**↑23 ↓4**	**↑187 ↓10**
T_1	(35)	*↑0 ↓5*	**↑32 ↓1**	**↑32 ↓0**	**↑34 ↓0**	**↑31 ↓2**	**↑27 ↓7**	**↑24 ↓10**	**↑21 ↓12**	**↑201 ↓37**
T_2	(35)	↑1 ↓1	**↑24 ↓1**	**↑31 ↓0**	**↑35 ↓0**	**↑31 ↓2**	**↑28 ↓4**	**↑21 ↓9**	**↑17 ↓11**	**↑188 ↓28**
T_3	(35)	*↑1 ↓2*	**↑17 ↓2**	**↑26 ↓0**	**↑34 ↓1**	**↑30 ↓2**	**↑26 ↓3**	**↑22 ↓7**	**↑21 ↓9**	**↑177 ↓26**
T_4	(35)	*↑2 ↓8*	**↑24 ↓1**	**↑32 ↓0**	**↑34 ↓0**	**↑31 ↓0**	**↑25 ↓5**	**↑24 ↓7**	**↑19 ↓11**	**↑191 ↓32**
T_5	(35)	*↑2 ↓6*	**↑22 ↓0**	**↑26 ↓0**	**↑31 ↓1**	**↑30 ↓3**	**↑25 ↓5**	**↑20 ↓8**	**↑20 ↓9**	**↑176 ↓32**
T_6	(35)	*↑0 ↓2*	**↑18 ↓0**	**↑26 ↓0**	**↑32 ↓0**	**↑31 ↓0**	**↑27 ↓3**	**↑22 ↓7**	**↑21 ↓11**	**↑177 ↓23**
All	(270)	*↑9 ↓26*	**↑191 ↓5**	**↑231 ↓1**	**↑258 ↓3**	**↑238 ↓12**	**↑209 ↓31**	**↑181 ↓52**	**↑162 ↓68**	**↑1479 ↓198**

Table D.3: SABrNorRes vs DynDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**12** ↓**0**
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**15** ↓**0**
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
C	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**44** ↓**0**
S	(6)	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**47** ↓**0**
		GBG								
F_{1a}	(6)	↑*0* ↓*2*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**40** ↓**2**
F_{1b}	(6)	↑**2** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**42** ↓**1**
F_2	(6)	↑*0* ↓*1*	↑1 ↓1	↑**2** ↓**1**	↑**3** ↓**2**	↑**3** ↓**1**	↑**3** ↓**1**	↑**3** ↓**1**	↑**3** ↓**2**	↑**18** ↓**10**
F_3	(6)	↑1 ↓1	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**1**
F_4	(6)	↑*0* ↓*1*	↑**3** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**4** ↓**1**	↑**32** ↓**2**
F_5	(6)	↑*0* ↓*5*	↑**5** ↓**1**	↑**4** ↓**0**	↑**3** ↓**0**	↑**3** ↓**0**	↑2 ↓2	↑*1* ↓*4*	↑*1* ↓*4*	↑**19** ↓**16**
F_6	(6)	↑**2** ↓**0**	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**39** ↓**0**
T_1	(7)	↑*0* ↓*4*	↑2 ↓2	↑**5** ↓**1**	↑**7** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**4** ↓**2**	↑**34** ↓**10**
T_2	(7)	↑**3** ↓**1**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**44** ↓**1**
T_3	(7)	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**5** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**4** ↓**2**	↑**35** ↓**5**
T_4	(7)	↑*0* ↓*4*	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**46** ↓**4**
T_5	(7)	↑*0* ↓*1*	↑**3** ↓**0**	↑**4** ↓**0**	↑**4** ↓**1**	↑**5** ↓**1**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**2**	↑**28** ↓**9**
T_6	(7)	↑**2** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**45** ↓**3**
All	(54)	↑**16** ↓**11**	↑**40** ↓**2**	↑**47** ↓**1**	↑**47** ↓**2**	↑**47** ↓**1**	↑**45** ↓**3**	↑**41** ↓**5**	↑**40** ↓**7**	↑**323** ↓**32**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**13** ↓**0**
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**14** ↓**0**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**15** ↓**0**
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
C	(6)	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**3** ↓**0**	↑**41** ↓**0**
S	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**48** ↓**0**
		GBG								
F_{1a}	(6)	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**41** ↓**1**
F_{1b}	(6)	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**41** ↓**1**
F_2	(6)	↑*0* ↓*3*	↑**3** ↓**1**	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**37** ↓**4**
F_3	(6)	↑*0* ↓*2*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**2**
F_4	(6)	↑*0* ↓*1*	↑1 ↓1	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**34** ↓**2**
F_5	(6)	↑*2* ↓*3*	↑*1* ↓*3*	↑*1* ↓*5*	↑*2* ↓*3*	↑*1* ↓*4*	↑*0* ↓*6*	↑*1* ↓*5*	↑*0* ↓*5*	↑*8* ↓*34*
F_6	(6)	↑*0* ↓*1*	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**40** ↓**1**
T_1	(7)	↑*0* ↓*2*	↑**4** ↓**3**	↑**3** ↓**1**	↑**7** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**0**	↑**38** ↓**9**
T_2	(7)	↑0 ↓0	↑**5** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**7** ↓**0**	↑**5** ↓**1**	↑**43** ↓**2**
T_3	(7)	↑1 ↓1	↑**5** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**4** ↓**1**	↑**40** ↓**7**
T_4	(7)	↑*0* ↓*6*	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**42** ↓**11**
T_5	(7)	↑*0* ↓*2*	↑**3** ↓**1**	↑**5** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**38** ↓**9**
T_6	(7)	↑1 ↓1	↑**5** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**42** ↓**7**
All	(54)	↑**13** ↓**12**	↑**40** ↓**5**	↑**44** ↓**5**	↑**50** ↓**3**	↑**49** ↓**4**	↑**47** ↓**6**	↑**47** ↓**5**	↑**42** ↓**5**	↑**332** ↓**45**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**14** ↓**0**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**13** ↓**0**
10	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
20	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
40	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**15** ↓**0**
80	(2)	↑*0* ↓*2*	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**2**
C	(6)	↑*0* ↓*1*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**39** ↓**1**
S	(6)	↑**5** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**47** ↓**1**
		GBG								
F_{1a}	(6)	↑*0* ↓*4*	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**41** ↓**4**
F_{1b}	(6)	↑*0* ↓*4*	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**41** ↓**4**
F_2	(6)	↑*0* ↓*6*	↑**2** ↓**0**	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**34** ↓**6**
F_3	(6)	↑*0* ↓*6*	↑**5** ↓**0**	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**27** ↓**6**
F_4	(6)	↑*0* ↓*6*	↑**1** ↓**0**	↑**2** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**32** ↓**6**
F_5	(6)	↑**6** ↓**0**	↑**4** ↓**2**	↑**4** ↓**2**	↑*2* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*16* ↓*31*
F_6	(6)	↑*0* ↓*4*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**42** ↓**4**
T_1	(7)	↑*1* ↓*3*	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**43** ↓**10**
T_2	(7)	↑*1* ↓*6*	↑**5** ↓**0**	↑**4** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**3** ↓**1**	↑**36** ↓**11**
T_3	(7)	↑*1* ↓*4*	↑**4** ↓**0**	↑**4** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**37** ↓**8**
T_4	(7)	↑*1* ↓*5*	↑**3** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**38** ↓**12**
T_5	(7)	↑*1* ↓*6*	↑**6** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**6** ↓**1**	↑**5** ↓**1**	↑**41** ↓**10**
T_6	(7)	↑*1* ↓*6*	↑**4** ↓**0**	↑**5** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**5** ↓**1**	↑**38** ↓**10**
All	(54)	↑*11* ↓*32*	↑**40** ↓**2**	↑**42** ↓**2**	↑**50** ↓**3**	↑**48** ↓**6**	↑**46** ↓**6**	↑**43** ↓**6**	↑**39** ↓**6**	↑**319** ↓**63**

Table D.4: SABrNorRes vs DynDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
5	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑13 ↓1**
10	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓1**
20	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑14 ↓0**
40	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
80	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
C	(6)	*↑0 ↓4*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑37 ↓4**
S	(6)	**↑3 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑45 ↓1**
		GBG								
F_{1a}	(6)	*↑0 ↓3*	↑2 ↓2	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓5**
F_{1b}	(6)	*↑0 ↓5*	*↑1 ↓3*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑36 ↓8**
F_2	(6)	*↑0 ↓6*	*↑1 ↓2*	**↑3 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑33 ↓8**
F_3	(6)	*↑0 ↓6*	**↑3 ↓0**	**↑2 ↓1**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	*↑1 ↓3*	*↑1 ↓3*	*↑12 ↓14*
F_4	(6)	*↑0 ↓6*	*↑0 ↓1*	**↑2 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑32 ↓8**
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓1**	**↑3 ↓2**	*↑2 ↓3*	*↑1 ↓4*	*↑1 ↓4*	**↑28 ↓14**
F_6	(6)	*↑0 ↓6*	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓6**
T_1	(7)	*↑1 ↓4*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑42 ↓9**
T_2	(7)	*↑1 ↓5*	**↑3 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑33 ↓11**
T_3	(7)	*↑1 ↓5*	↑1 ↓1	**↑2 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑30 ↓12**
T_4	(7)	*↑1 ↓6*	↑2 ↓2	**↑5 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑33 ↓14**
T_5	(7)	*↑1 ↓6*	**↑4 ↓2**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑42 ↓8**
T_6	(7)	*↑1 ↓6*	↑3 ↓3	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓9**
All	(54)	*↑9 ↓37*	**↑30 ↓8**	**↑40 ↓2**	**↑46 ↓1**	**↑47 ↓2**	**↑45 ↓4**	**↑41 ↓7**	**↑42 ↓7**	**↑300 ↓68**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑7 ↓9*
5	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑7 ↓9*
10	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑5 ↓9*
20	(2)	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑5 ↓8*
40	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑6 ↓9*
80	(2)	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓7
C	(6)	*↑0 ↓5*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑37 ↓5**
S	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓46*
		GBG								
F_{1a}	(6)	*↑0 ↓3*	*↑1 ↓3*	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑34 ↓6**
F_{1b}	(6)	*↑0 ↓6*	*↑1 ↓5*	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑33 ↓11**
F_2	(6)	*↑0 ↓6*	*↑0 ↓5*	*↑1 ↓2*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑29 ↓13**
F_3	(6)	*↑0 ↓6*	*↑2 ↓3*	*↑1 ↓2*	**↑2 ↓1**	**↑3 ↓0**	**↑3 ↓0**	**↑2 ↓1**	*↑0 ↓2*	*↑13 ↓15*
F_4	(6)	*↑0 ↓6*	*↑0 ↓5*	**↑2 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑31 ↓12**
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑46 ↓1**
F_6	(6)	*↑0 ↓6*	**↑3 ↓1**	**↑4 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑35 ↓8**
T_1	(7)	*↑1 ↓5*	**↑5 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑45 ↓5**
T_2	(7)	*↑1 ↓5*	*↑1 ↓4*	**↑2 ↓0**	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑33 ↓10**
T_3	(7)	*↑1 ↓5*	*↑1 ↓4*	*↑1 ↓4*	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑31 ↓15**
T_4	(7)	*↑1 ↓6*	*↑2 ↓5*	**↑3 ↓1**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑34 ↓14**
T_5	(7)	*↑1 ↓6*	*↑3 ↓4*	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑43 ↓10**
T_6	(7)	*↑1 ↓6*	*↑1 ↓5*	**↑3 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑35 ↓12**
All	(54)	*↑6 ↓44*	*↑19 ↓28*	**↑25 ↓11**	**↑41 ↓6**	**↑45 ↓6**	**↑45 ↓6**	**↑40 ↓7**	**↑37 ↓9**	**↑258 ↓117**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑5 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑7 ↓1**	**↑6 ↓1**	**↑61 ↓10**
5	(10)	**↑6 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑63 ↓10**
10	(10)	**↑5 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑5 ↓1**	**↑62 ↓10**
20	(10)	**↑6 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑8 ↓1**	**↑66 ↓8**
40	(10)	**↑5 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑67 ↓10**
80	(10)	*↑3 ↓5*	**↑9 ↓1**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑66 ↓10**
C	(30)	**↑11 ↓10**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑29 ↓0**	**↑21 ↓0**	**↑18 ↓0**	**↑198 ↓10**
S	(30)	**↑19 ↓8**	**↑24 ↓6**	**↑24 ↓5**	**↑24 ↓5**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑187 ↓48**
		GBG								
F_{1a}	(30)	*↑0 ↓13*	**↑18 ↓5**	**↑26 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑193 ↓18**
F_{1b}	(30)	*↑2 ↓17*	**↑19 ↓8**	**↑25 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑28 ↓0**	**↑193 ↓25**
F_2	(30)	*↑0 ↓22*	*↑7 ↓9*	**↑13 ↓3**	**↑24 ↓2**	**↑27 ↓1**	**↑27 ↓1**	**↑27 ↓1**	**↑26 ↓2**	**↑151 ↓41**
F_3	(30)	*↑1 ↓21*	**↑21 ↓3**	**↑18 ↓3**	**↑22 ↓1**	**↑23 ↓0**	**↑20 ↓1**	**↑17 ↓4**	**↑14 ↓5**	**↑136 ↓38**
F_4	(30)	*↑0 ↓20*	*↑5 ↓7*	**↑15 ↓2**	**↑28 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑28 ↓0**	**↑26 ↓1**	**↑161 ↓30**
F_5	(30)	**↑20 ↓8**	**↑22 ↓6**	**↑21 ↓7**	**↑16 ↓7**	**↑13 ↓12**	*↑10 ↓17*	*↑8 ↓19*	*↑7 ↓20*	**↑117 ↓96**
F_6	(30)	*↑2 ↓17*	**↑23 ↓1**	**↑26 ↓1**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑28 ↓0**	**↑28 ↓0**	**↑196 ↓19**
T_1	(35)	*↑3 ↓18*	**↑22 ↓6**	**↑25 ↓3**	**↑33 ↓2**	**↑32 ↓3**	**↑30 ↓3**	**↑29 ↓4**	**↑28 ↓4**	**↑202 ↓43**
T_2	(35)	*↑6 ↓17*	**↑19 ↓4**	**↑22 ↓0**	**↑29 ↓1**	**↑31 ↓2**	**↑30 ↓3**	**↑28 ↓3**	**↑24 ↓5**	**↑189 ↓35**
T_3	(35)	*↑4 ↓16*	**↑17 ↓5**	**↑19 ↓6**	**↑29 ↓3**	**↑29 ↓2**	**↑27 ↓3**	**↑25 ↓5**	**↑23 ↓7**	**↑173 ↓47**
T_4	(35)	*↑3 ↓27*	**↑19 ↓9**	**↑27 ↓4**	**↑28 ↓1**	**↑30 ↓1**	**↑30 ↓3**	**↑28 ↓5**	**↑28 ↓5**	**↑193 ↓55**
T_5	(35)	*↑3 ↓21*	**↑19 ↓7**	**↑26 ↓1**	**↑30 ↓2**	**↑31 ↓3**	**↑29 ↓4**	**↑28 ↓4**	**↑26 ↓4**	**↑192 ↓46**
T_6	(35)	*↑6 ↓19*	**↑19 ↓8**	**↑25 ↓2**	**↑31 ↓1**	**↑30 ↓2**	**↑29 ↓3**	**↑29 ↓3**	**↑29 ↓3**	**↑198 ↓41**
All	(270)	*↑55 ↓136*	**↑169 ↓45**	**↑198 ↓21**	**↑234 ↓15**	**↑236 ↓19**	**↑228 ↓25**	**↑212 ↓30**	**↑200 ↓34**	**↑1532 ↓325**

Table D.5: SACDE vs DynDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑9 ↓0**
5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑15 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑15 ↓0**
20	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
40	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
80	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑40 ↓0**
S	(6)	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑47 ↓0**
		GBG								
F_{1a}	(6)	*↑0 ↓3*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓3**
F_{1b}	(6)	*↑0 ↓2*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑40 ↓2**
F_2	(6)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑3 ↓3	↑3 ↓3	*↑2 ↓3*	*↑2 ↓3*	**↑3 ↓2**	*↑15 ↓18*
F_3	(6)	*↑0 ↓2*	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑41 ↓2**
F_4	(6)	*↑0 ↓2*	**↑3 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑30 ↓3**
F_5	(6)	*↑0 ↓5*	**↑3 ↓1**	**↑5 ↓1**	**↑3 ↓0**	**↑2 ↓1**	*↑2 ↓3*	*↑1 ↓4*	*↑0 ↓5*	*↑16 ↓20*
F_6	(6)	**↑2 ↓1**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑38 ↓1**
T_1	(7)	*↑0 ↓5*	*↑1 ↓2*	**↑3 ↓2**	**↑7 ↓0**	**↑6 ↓0**	**↑3 ↓1**	**↑4 ↓1**	**↑4 ↓2**	**↑28 ↓13**
T_2	(7)	*↑1 ↓3*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓0**	**↑42 ↓7**
T_3	(7)	*↑0 ↓1*	**↑7 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑4 ↓2**	**↑37 ↓9**
T_4	(7)	*↑0 ↓7*	**↑4 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑42 ↓9**
T_5	(7)	*↑0 ↓1*	**↑2 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑4 ↓2**	**↑4 ↓2**	**↑4 ↓1**	**↑4 ↓2**	**↑28 ↓9**
T_6	(7)	**↑1 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑44 ↓2**
All	(54)	*↑12 ↓17*	**↑38 ↓2**	**↑45 ↓2**	**↑47 ↓3**	**↑45 ↓4**	**↑42 ↓6**	**↑40 ↓7**	**↑39 ↓8**	**↑308 ↓49**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑14 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑15 ↓0**
20	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
40	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
80	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑41 ↓0**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
		GBG								
F_{1a}	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑41 ↓1**
F_{1b}	(6)	*↑0 ↓1*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑38 ↓1**
F_2	(6)	*↑0 ↓4*	↑1 ↓1	**↑3 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑29 ↓6**
F_3	(6)	*↑0 ↓5*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓5**
F_4	(6)	*↑0 ↓3*	*↑0 ↓1*	**↑4 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑29 ↓5**
F_5	(6)	↑3 ↓3	*↑1 ↓3*	*↑1 ↓5*	*↑2 ↓3*	*↑1 ↓5*	*↑2 ↓3*	*↑0 ↓4*	*↑1 ↓4*	*↑11 ↓30*
F_6	(6)	*↑1 ↓2*	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓2**
T_1	(7)	*↑1 ↓3*	*↑2 ↓3*	↑3 ↓3	**↑7 ↓0**	**↑6 ↓1**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓10**
T_2	(7)	*↑1 ↓2*	**↑4 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑6 ↓0**	**↑38 ↓3**
T_3	(7)	*↑1 ↓3*	**↑5 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑4 ↓1**	**↑39 ↓9**
T_4	(7)	*↑0 ↓7*	**↑3 ↓1**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑38 ↓11**
T_5	(7)	*↑0 ↓2*	**↑3 ↓1**	**↑4 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑35 ↓9**
T_6	(7)	*↑1 ↓2*	**↑5 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑42 ↓8**
All	(54)	*↑16 ↓19*	**↑34 ↓5**	**↑43 ↓7**	**↑48 ↓3**	**↑46 ↓5**	**↑46 ↓3**	**↑44 ↓4**	**↑42 ↓4**	**↑319 ↓50**
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
20	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
40	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
80	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
C	(6)	↑0 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑36 ↓0**
S	(6)	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑47 ↓0**
		GBG								
F_{1a}	(6)	*↑0 ↓3*	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑38 ↓3**
F_{1b}	(6)	*↑0 ↓6*	**↑4 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓7**
F_2	(6)	*↑0 ↓6*	↑0 ↓0	↑1 ↓1	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑28 ↓7**
F_3	(6)	*↑0 ↓6*	**↑4 ↓0**	**↑3 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑28 ↓6**
F_4	(6)	*↑0 ↓6*	↑0 ↓0	↑1 ↓0	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑30 ↓6**
F_5	(6)	**↑6 ↓0**	**↑4 ↓2**	**↑4 ↓2**	*↑2 ↓3*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑16 ↓31*
F_6	(6)	*↑0 ↓5*	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑40 ↓5**
T_1	(7)	*↑1 ↓4*	**↑3 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑39 ↓11**
T_2	(7)	*↑1 ↓5*	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑4 ↓1**	**↑3 ↓1**	**↑34 ↓10**
T_3	(7)	*↑1 ↓5*	**↑1 ↓0**	**↑4 ↓1**	**↑6 ↓0**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑33 ↓10**
T_4	(7)	*↑1 ↓6*	↑2 ↓2	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑38 ↓14**
T_5	(7)	*↑1 ↓6*	**↑4 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑37 ↓10**
T_6	(7)	*↑1 ↓6*	**↑4 ↓0**	**↑4 ↓0**	**↑7 ↓0**	**↑5 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑39 ↓10**
All	(54)	*↑11 ↓32*	**↑31 ↓3**	**↑38 ↓3**	**↑48 ↓3**	**↑47 ↓6**	**↑44 ↓6**	**↑42 ↓6**	**↑42 ↓6**	**↑303 ↓65**

Table D.6: SACDE vs DynDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
5	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
10	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓0**
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
40	(2)	*↑0 ↓1*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
80	(2)	*↑0 ↓2*	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓2**
C	(6)	*↑0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑35 ↓2**
S	(6)	*↑0 ↓1*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑42 ↓1**
		GBG								
F_{1a}	(6)	*↑0 ↓4*	*↑1 ↓2*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑35 ↓6**
F_{1b}	(6)	*↑0 ↓6*	*↑1 ↓4*	**↑3 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑34 ↓11**
F_2	(6)	*↑0 ↓6*	*↑0 ↓3*	↑2 ↓2	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑31 ↓11**
F_3	(6)	*↑0 ↓6*	**↑2 ↓0**	*↑1 ↓3*	**↑3 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑2 ↓0**	**↑3 ↓0**	**↑20 ↓9**
F_4	(6)	*↑0 ↓6*	*↑0 ↓3*	↑1 ↓1	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑29 ↓10**
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑3 ↓1**	**↑3 ↓2**	*↑2 ↓3*	*↑1 ↓4*	*↑1 ↓4*	**↑28 ↓14**
F_6	(6)	*↑0 ↓6*	**↑3 ↓0**	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓6**
T_1	(7)	*↑1 ↓5*	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑40 ↓10**
T_2	(7)	*↑1 ↓6*	↑2 ↓2	**↑4 ↓1**	**↑5 ↓0**	**↑6 ↓1**	**↑6 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑34 ↓13**
T_3	(7)	*↑1 ↓6*	*↑1 ↓3*	*↑1 ↓3*	**↑4 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑29 ↓14**
T_4	(7)	*↑1 ↓6*	↑2 ↓2	**↑3 ↓2**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑35 ↓13**
T_5	(7)	*↑1 ↓5*	**↑3 ↓2**	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑39 ↓7**
T_6	(7)	*↑1 ↓6*	*↑1 ↓3*	**↑4 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑37 ↓10**
All	(54)	*↑6 ↓37*	**↑25 ↓12**	**↑33 ↓7**	**↑46 ↓1**	**↑50 ↓2**	**↑47 ↓3**	**↑41 ↓4**	**↑43 ↓4**	**↑291 ↓70**
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑7 ↓9*
5	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑7 ↓9*
10	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑5 ↓9*
20	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓2*	*↑5 ↓10*
40	(2)	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓2*	*↑6 ↓10*
80	(2)	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑7 ↓7
C	(6)	*↑0 ↓6*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑3 ↓2**	**↑37 ↓8**
S	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓46*
		GBG								
F_{1a}	(6)	*↑0 ↓3*	*↑1 ↓4*	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑33 ↓7**
F_{1b}	(6)	*↑0 ↓6*	*↑1 ↓5*	*↑0 ↓3*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑31 ↓14**
F_2	(6)	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓4*	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑26 ↓16**
F_3	(6)	*↑0 ↓6*	*↑1 ↓2*	*↑1 ↓2*	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑28 ↓10**
F_4	(6)	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	**↑2 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑26 ↓16**
F_5	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑46 ↓1**
F_6	(6)	*↑0 ↓6*	**↑3 ↓1**	**↑4 ↓1**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑37 ↓8**
T_1	(7)	*↑1 ↓5*	**↑4 ↓1**	**↑3 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑43 ↓6**
T_2	(7)	*↑1 ↓5*	*↑1 ↓4*	↑2 ↓2	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓1**	**↑35 ↓12**
T_3	(7)	*↑1 ↓5*	*↑1 ↓4*	*↑1 ↓4*	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑36 ↓13**
T_4	(7)	*↑1 ↓6*	*↑2 ↓5*	*↑2 ↓4*	**↑5 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑38 ↓15**
T_5	(7)	*↑1 ↓6*	*↑3 ↓4*	**↑3 ↓2**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑42 ↓12**
T_6	(7)	*↑1 ↓6*	*↑1 ↓5*	*↑2 ↓3*	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑33 ↓14**
All	(54)	*↑6 ↓45*	*↑18 ↓29*	*↑19 ↓20*	**↑39 ↓5**	**↑47 ↓6**	**↑47 ↓6**	**↑44 ↓6**	**↑44 ↓9**	**↑264 ↓126**
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑3 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑57 ↓9**
5	(10)	**↑5 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑5 ↓1**	**↑6 ↓1**	**↑60 ↓9**
10	(10)	**↑5 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑6 ↓1**	**↑4 ↓1**	**↑58 ↓9**
20	(10)	**↑5 ↓2**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑7 ↓1**	**↑7 ↓2**	**↑64 ↓10**
40	(10)	**↑5 ↓3**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓2**	**↑67 ↓11**
80	(10)	↑4 ↓4	**↑9 ↓1**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑67 ↓9**
C	(30)	**↑11 ↓8**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑28 ↓0**	**↑25 ↓0**	**↑19 ↓0**	**↑17 ↓2**	**↑189 ↓10**
S	(30)	**↑16 ↓7**	**↑24 ↓6**	**↑24 ↓5**	**↑24 ↓5**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑24 ↓6**	**↑184 ↓47**
		GBG								
F_{1a}	(30)	*↑0 ↓14*	**↑15 ↓6**	**↑24 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑188 ↓20**
F_{1b}	(30)	*↑0 ↓21*	**↑15 ↓10**	**↑21 ↓4**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑30 ↓0**	**↑27 ↓0**	**↑183 ↓35**
F_2	(30)	*↑0 ↓24*	*↑2 ↓11*	*↑7 ↓9*	**↑20 ↓3**	**↑26 ↓3**	**↑24 ↓3**	**↑24 ↓3**	**↑26 ↓2**	**↑129 ↓58**
F_3	(30)	*↑0 ↓25*	**↑18 ↓2**	**↑17 ↓5**	**↑25 ↓0**	**↑27 ↓0**	**↑25 ↓0**	**↑22 ↓0**	**↑25 ↓0**	**↑159 ↓32**
F_4	(30)	*↑0 ↓23*	*↑3 ↓9*	**↑10 ↓7**	**↑24 ↓0**	**↑28 ↓0**	**↑27 ↓0**	**↑26 ↓0**	**↑26 ↓1**	**↑144 ↓40**
F_5	(30)	**↑21 ↓8**	**↑20 ↓6**	**↑22 ↓8**	**↑16 ↓7**	*↑12 ↓14*	*↑12 ↓15*	*↑7 ↓18*	*↑7 ↓20*	**↑117 ↓96**
F_6	(30)	*↑3 ↓20*	**↑19 ↓1**	**↑23 ↓1**	**↑30 ↓0**	**↑30 ↓0**	**↑29 ↓0**	**↑29 ↓0**	**↑29 ↓0**	**↑192 ↓22**
T_1	(35)	*↑4 ↓22*	**↑14 ↓7**	**↑19 ↓6**	**↑33 ↓2**	**↑31 ↓3**	**↑29 ↓3**	**↑29 ↓3**	**↑29 ↓4**	**↑188 ↓50**
T_2	(35)	*↑5 ↓21*	**↑18 ↓6**	**↑23 ↓3**	**↑26 ↓2**	**↑30 ↓3**	**↑30 ↓3**	**↑26 ↓4**	**↑25 ↓3**	**↑183 ↓45**
T_3	(35)	*↑4 ↓20*	**↑15 ↓7**	**↑17 ↓9**	**↑26 ↓2**	**↑31 ↓3**	**↑28 ↓4**	**↑28 ↓5**	**↑25 ↓5**	**↑174 ↓55**
T_4	(35)	*↑3 ↓32*	**↑13 ↓10**	**↑22 ↓8**	**↑30 ↓1**	**↑32 ↓2**	**↑32 ↓2**	**↑29 ↓3**	**↑30 ↓4**	**↑191 ↓62**
T_5	(35)	*↑3 ↓20*	**↑15 ↓7**	**↑20 ↓3**	**↑31 ↓2**	**↑30 ↓4**	**↑28 ↓4**	**↑26 ↓3**	**↑28 ↓4**	**↑181 ↓47**
T_6	(35)	*↑5 ↓20*	**↑17 ↓8**	**↑23 ↓5**	**↑29 ↓1**	**↑29 ↓2**	**↑30 ↓2**	**↑30 ↓3**	**↑32 ↓3**	**↑195 ↓44**
All	(270)	*↑51 ↓150*	**↑146 ↓51**	**↑178 ↓39**	**↑228 ↓15**	**↑235 ↓23**	**↑226 ↓24**	**↑211 ↓27**	**↑210 ↓31**	**↑1485 ↓360**

Table D.7: SADynPopDE vs CDE performance analysis - Part 1

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑**2** ↓**0**	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑*3* ↓*11*
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑*0* ↓*1*	↑*7* ↓*8*
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**13** ↓**0**
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
C	(6)	↑**6** ↓**0**	↑**5** ↓**1**	↑**5** ↓**1**	↑**4** ↓**2**	↑**3** ↓**2**	↑**3** ↓**2**	↑**5** ↓**1**	↑**4** ↓**0**	↑**35** ↓**9**
S	(6)	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**4** ↓**1**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**2**	↑**36** ↓**10**
		GBG								
F_{1a}	(6)	↑**4** ↓**1**	↑**4** ↓**2**	↑2 ↓2	↑*1* ↓*2*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**33** ↓**7**
F_{1b}	(6)	↑**2** ↓**1**	↑*2* ↓*3*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*5*	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑20 ↓20
F_2	(6)	↑**2** ↓**0**	↑**4** ↓**1**	↑**4** ↓**1**	↑**4** ↓**2**	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**36** ↓**5**
F_3	(6)	↑*0* ↓*1*	↑**3** ↓**1**	↑0 ↓0	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**33** ↓**2**
F_4	(6)	↑**4** ↓**0**	↑**3** ↓**1**	↑**3** ↓**2**	↑*1* ↓*5*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*3*	↑**2** ↓**0**	↑*13* ↓*21*
F_5	(6)	↑1 ↓1	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*43*
F_6	(6)	↑**4** ↓**0**	↑**3** ↓**1**	↑*1* ↓*3*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*4*	↑*8* ↓*29*
T_1	(7)	↑**6** ↓**0**	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*6*	↑*1* ↓*5*	↑**4** ↓**3**	↑**4** ↓**2**	↑**5** ↓**2**	↑*21* ↓*30*
T_2	(7)	↑**2** ↓**0**	↑**5** ↓**1**	↑*2* ↓*3*	↑*2* ↓*4*	↑*3* ↓*4*	↑**4** ↓**3**	↑**4** ↓**3**	↑**4** ↓**2**	↑**26** ↓**20**
T_3	(7)	↑**4** ↓**0**	↑**6** ↓**1**	↑**2** ↓**1**	↑*2* ↓*3*	↑*2* ↓*3*	↑**4** ↓**3**	↑**4** ↓**3**	↑**4** ↓**2**	↑**28** ↓**16**
T_4	(7)	↑*0* ↓*4*	↑*0* ↓*4*	↑*0* ↓*5*	↑*1* ↓*6*	↑*2* ↓*4*	↑**4** ↓**3**	↑**4** ↓**3**	↑**4** ↓**2**	↑*15* ↓*31*
T_5	(7)	↑**1** ↓**0**	↑**3** ↓**1**	↑**3** ↓**2**	↑**4** ↓**3**	↑**3** ↓**2**	↑**3** ↓**1**	↑**4** ↓**1**	↑**5** ↓**1**	↑**26** ↓**11**
T_6	(7)	↑**4** ↓**0**	↑**5** ↓**2**	↑**3** ↓**2**	↑*2* ↓*4*	↑*3* ↓*4*	↑*3* ↓*4*	↑**4** ↓**2**	↑**4** ↓**1**	↑**28** ↓**19**
All	(54)	↑**29** ↓**4**	↑**29** ↓**16**	↑*20* ↓*21*	↑*20* ↓*29*	↑*21* ↓*26*	↑**29** ↓**21**	↑**33** ↓**17**	↑**34** ↓**12**	↑**215** ↓**146**
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**9** ↓**2**
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑*0* ↓*1*	↑*0* ↓*1*	↑*0* ↓*1*	↑*0* ↓*1*	↑0 ↓0	↑**5** ↓**4**
10	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*7* ↓*9*
20	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**12** ↓**0**
40	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
80	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**16** ↓**0**
C	(6)	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑3 ↓3	↑*2* ↓*3*	↑2 ↓2	↑2 ↓2	↑**2** ↓**1**	↑**26** ↓**11**
S	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**4** ↓**1**	↑**4** ↓**1**	↑**4** ↓**1**	↑**39** ↓**4**
		GBG								
F_{1a}	(6)	↑**2** ↓**1**	↑*0* ↓*4*	↑*0* ↓*5*	↑*0* ↓*4*	↑*1* ↓*2*	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**21** ↓**16**
F_{1b}	(6)	↑**2** ↓**1**	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*4*	↑**3** ↓**2**	↑**6** ↓**0**	↑**5** ↓**0**	↑*16* ↓*23*
F_2	(6)	↑*0* ↓*2*	↑*0* ↓*2*	↑*1* ↓*4*	↑*2* ↓*4*	↑3 ↓3	↑3 ↓3	↑**4** ↓**0**	↑**6** ↓**0**	↑**19** ↓**18**
F_3	(6)	↑*0* ↓*2*	↑1 ↓1	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*3*	↑**4** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑*17* ↓*18*
F_4	(6)	↑1 ↓1	↑*0* ↓*2*	↑*0* ↓*3*	↑*2* ↓*4*	↑*2* ↓*3*	↑*2* ↓*3*	↑3 ↓3	↑**5** ↓**0**	↑*15* ↓*19*
F_5	(6)	↑2 ↓2	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*44*
F_6	(6)	↑**2** ↓**0**	↑**3** ↓**2**	↑*1* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*6* ↓*36*
T_1	(7)	↑**4** ↓**0**	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*1* ↓*4*	↑3 ↓3	↑**5** ↓**2**	↑*13* ↓*36*
T_2	(7)	↑**1** ↓**0**	↑*1* ↓*3*	↑*0* ↓*7*	↑*0* ↓*7*	↑*1* ↓*4*	↑**4** ↓**2**	↑**5** ↓**2**	↑**5** ↓**2**	↑*17* ↓*27*
T_3	(7)	↑1 ↓1	↑1 ↓1	↑*1* ↓*3*	↑*2* ↓*3*	↑*2* ↓*3*	↑**5** ↓**2**	↑**5** ↓**2**	↑**4** ↓**2**	↑**21** ↓**17**
T_4	(7)	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*6*	↑*2* ↓*5*	↑3 ↓3	↑**4** ↓**2**	↑*9* ↓*43*
T_5	(7)	↑*0* ↓*2*	↑*1* ↓*2*	↑*1* ↓*5*	↑*2* ↓*4*	↑**3** ↓**2**	↑**4** ↓**2**	↑**5** ↓**2**	↑**5** ↓**2**	↑21 ↓21
T_6	(7)	↑**3** ↓**0**	↑*1* ↓*3*	↑*0* ↓*5*	↑*0* ↓*7*	↑*0* ↓*5*	↑*2* ↓*5*	↑**4** ↓**3**	↑**5** ↓**2**	↑*15* ↓*30*
All	(54)	↑**21** ↓**9**	↑*15* ↓*22*	↑*12* ↓*34*	↑*12* ↓*38*	↑*12* ↓*31*	↑**24** ↓**23**	↑**31** ↓**18**	↑**34** ↓**14**	↑*161* ↓*189*
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**9** ↓**4**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**9** ↓**6**
10	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**9** ↓**6**
20	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑*0* ↓*1*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*5* ↓*8*
40	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑7 ↓7
80	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**14** ↓**0**
C	(6)	↑0 ↓0	↑**6** ↓**0**	↑**3** ↓**2**	↑*1* ↓*5*	↑*1* ↓*5*	↑*1* ↓*5*	↑*1* ↓*5*	↑*1* ↓*4*	↑*14* ↓*26*
S	(6)	↑**5** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**4** ↓**2**	↑**4** ↓**2**	↑**39** ↓**5**
		GBG								
F_{1a}	(6)	↑*1* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑**3** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑*16* ↓*27*
F_{1b}	(6)	↑*1* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*2*	↑**5** ↓**0**	↑**6** ↓**0**	↑*13* ↓*30*
F_2	(6)	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*5*	↑*1* ↓*4*	↑*2* ↓*3*	↑*4* ↓*39*
F_3	(6)	↑*0* ↓*5*	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*45*
F_4	(6)	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*5*	↑*1* ↓*5*	↑*2* ↓*4*	↑*2* ↓*3*	↑*5* ↓*38*
F_5	(6)	↑**6** ↓**0**	↑**4** ↓**2**	↑3 ↓3	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*13* ↓*35*
F_6	(6)	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*46*
T_1	(7)	↑**3** ↓**1**	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*6*	↑*2* ↓*5*	↑*2* ↓*5*	↑*7* ↓*44*
T_2	(7)	↑*1* ↓*4*	↑*1* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*5*	↑*2* ↓*5*	↑*2* ↓*3*	↑*6* ↓*44*
T_3	(7)	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑*0* ↓*7*	↑*0* ↓*6*	↑*0* ↓*5*	↑*2* ↓*3*	↑**4** ↓**3**	↑*9* ↓*41*
T_4	(7)	↑*1* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*1* ↓*6*	↑*2* ↓*5*	↑*2* ↓*5*	↑*6* ↓*50*
T_5	(7)	↑*1* ↓*5*	↑*1* ↓*3*	↑*1* ↓*4*	↑*0* ↓*7*	↑*0* ↓*6*	↑**4** ↓**3**	↑**4** ↓**3**	↑**4** ↓**3**	↑*15* ↓*34*
T_6	(7)	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*1* ↓*5*	↑*2* ↓*5*	↑*2* ↓*5*	↑*8* ↓*47*
All	(54)	↑*13* ↓*27*	↑*16* ↓*34*	↑*12* ↓*39*	↑*6* ↓*47*	↑*6* ↓*45*	↑*11* ↓*36*	↑*19* ↓*33*	↑*21* ↓*30*	↑*104* ↓*291*

Table D.8: SADynPopDE vs CDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**8** ↓**5**
5	(2)	↑**1** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**8** ↓**6**
10	(2)	↑**1** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**8** ↓**6**
20	(2)	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑*0* ↓*1*	↑*0* ↓*1*	↑*4* ↓*7*
40	(2)	↑0 ↓0	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*4* ↓*9*
80	(2)	↑*0* ↓*2*	↑**1** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*0* ↓*2*	↑*0* ↓*2*	↑*6* ↓*9*
C	(6)	↑*0* ↓*1*	↑**1** ↓**0**	↑*1* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑*2* ↓*33*
S	(6)	↑**4** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**1**	↑**4** ↓**2**	↑**5** ↓**1**	↑**3** ↓**2**	↑**3** ↓**2**	↑**36** ↓**9**
		GBG								
F_{1a}	(6)	↑*0* ↓*4*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑**5** ↓**0**	↑**6** ↓**0**	↑*11* ↓*32*
F_{1b}	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑**5** ↓**0**	↑**6** ↓**0**	↑*11* ↓*36*
F_2	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*1*	↑*0* ↓*43*
F_3	(6)	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*47*
F_4	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑**3** ↓**0**	↑*3* ↓*42*
F_5	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑3 ↓3	↑*1* ↓*3*	↑*1* ↓*4*	↑*1* ↓*5*	↑*1* ↓*5*	↑**25** ↓**20**
F_6	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*48*
T_1	(7)	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*2* ↓*5*	↑3 ↓3	↑*8* ↓*46*
T_2	(7)	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*2* ↓*5*	↑3 ↓3	↑*8* ↓*47*
T_3	(7)	↑*1* ↓*5*	↑*1* ↓*5*	↑*1* ↓*5*	↑*1* ↓*6*	↑*0* ↓*6*	↑*0* ↓*7*	↑*1* ↓*5*	↑*2* ↓*3*	↑*7* ↓*42*
T_4	(7)	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*1* ↓*5*	↑3 ↓3	↑*7* ↓*47*
T_5	(7)	↑*1* ↓*6*	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*5*	↑*2* ↓*3*	↑*8* ↓*43*
T_6	(7)	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*3* ↓*4*	↑3 ↓3	↑*12* ↓*43*
All	(54)	↑*10* ↓*36*	↑*13* ↓*34*	↑*13* ↓*38*	↑*8* ↓*46*	↑*5* ↓*47*	↑*6* ↓*47*	↑*14* ↓*37*	↑*19* ↓*25*	↑*88* ↓*310*
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑*1* ↓*14*
5	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑1 ↓1	↑*0* ↓*2*	↑*1* ↓*14*
10	(2)	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*16*
20	(2)	↑*0* ↓*1*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*14*
40	(2)	↑*0* ↓*2*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*15*
80	(2)	↑*0* ↓*1*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*14*
C	(6)	↑*0* ↓*4*	↑*0* ↓*1*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*41*
S	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*5*	↑*1* ↓*5*	↑*2* ↓*46*
		GBG								
F_{1a}	(6)	↑*0* ↓*3*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑**5** ↓**0**	↑*5* ↓*38*
F_{1b}	(6)	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑**6** ↓**0**	↑*6* ↓*40*
F_2	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*48*
F_3	(6)	↑*0* ↓*6*	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*46*
F_4	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*48*
F_5	(6)	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**4** ↓**2**	↑**43** ↓**3**
F_6	(6)	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*48*
T_1	(7)	↑*1* ↓*4*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*3* ↓*4*	↑*10* ↓*44*
T_2	(7)	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*7*	↑*2* ↓*5*	↑*6* ↓*47*
T_3	(7)	↑*1* ↓*4*	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*2* ↓*4*	↑*9* ↓*43*
T_4	(7)	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*2* ↓*5*	↑*9* ↓*47*
T_5	(7)	↑*1* ↓*6*	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*3* ↓*4*	↑*10* ↓*45*
T_6	(7)	↑*1* ↓*6*	↑*1* ↓*5*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*1* ↓*6*	↑*3* ↓*4*	↑*10* ↓*45*
All	(54)	↑*6* ↓*41*	↑*6* ↓*40*	↑*6* ↓*48*	↑*6* ↓*48*	↑*5* ↓*48*	↑*5* ↓*48*	↑*6* ↓*48*	↑*16* ↓*37*	↑*56* ↓*358*
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑**6** ↓**2**	↑**4** ↓**2**	↑*3* ↓*5*	↑*3* ↓*6*	↑*3* ↓*7*	↑*3* ↓*6*	↑*3* ↓*6*	↑**5** ↓**2**	↑*30* ↓*36*
5	(10)	↑**6** ↓**2**	↑**7** ↓**1**	↑**5** ↓**4**	↑*2* ↓*7*	↑*2* ↓*7*	↑*2* ↓*7*	↑*4* ↓*5*	↑*2* ↓*5*	↑*30* ↓*38*
10	(10)	↑**6** ↓**2**	↑**7** ↓**2**	↑**6** ↓**4**	↑5 ↓5	↑*3* ↓*6*	↑*3* ↓*6*	↑*4* ↓*6*	↑*3* ↓*6*	↑37 ↓37
20	(10)	↑**6** ↓**1**	↑**7** ↓**1**	↑**7** ↓**2**	↑*4* ↓*5*	↑*3* ↓*5*	↑*4* ↓*5*	↑*3* ↓*5*	↑*3* ↓*5*	↑**37** ↓**29**
40	(10)	↑**5** ↓**2**	↑**8** ↓**1**	↑**7** ↓**2**	↑**6** ↓**4**	↑5 ↓5	↑*4* ↓*5*	↑*4* ↓*6*	↑*4* ↓*6*	↑**43** ↓**31**
80	(10)	↑**4** ↓**3**	↑**7** ↓**1**	↑**8** ↓**2**	↑**7** ↓**3**	↑**7** ↓**3**	↑**7** ↓**3**	↑**6** ↓**4**	↑**6** ↓**4**	↑**52** ↓**23**
C	(30)	↑**12** ↓**5**	↑**17** ↓**2**	↑**13** ↓**12**	↑*8* ↓*22*	↑*6* ↓*22*	↑*6* ↓*21*	↑*8* ↓*20*	↑*7* ↓*16*	↑*77* ↓*120*
S	(30)	↑**21** ↓**7**	↑**23** ↓**6**	↑**23** ↓**7**	↑**19** ↓**8**	↑**17** ↓**11**	↑**17** ↓**11**	↑**16** ↓**12**	↑**16** ↓**12**	↑**152** ↓**74**
		GBG								
F_{1a}	(30)	↑*7* ↓*13*	↑*4* ↓*22*	↑*2* ↓*24*	↑*1* ↓*24*	↑*5* ↓*19*	↑**15** ↓**12**	↑**23** ↓**6**	↑**29** ↓**0**	↑*86* ↓*120*
F_{1b}	(30)	↑*5* ↓*16*	↑*2* ↓*26*	↑*0* ↓*29*	↑*0* ↓*28*	↑*0* ↓*27*	↑*8* ↓*17*	↑**22** ↓**6**	↑**29** ↓**0**	↑*66* ↓*149*
F_2	(30)	↑*2* ↓*19*	↑*4* ↓*20*	↑*5* ↓*22*	↑*6* ↓*24*	↑*7* ↓*22*	↑*10* ↓*20*	↑*11* ↓*16*	↑**14** ↓**10**	↑*59* ↓*153*
F_3	(30)	↑*0* ↓*20*	↑*4* ↓*15*	↑*0* ↓*24*	↑*6* ↓*24*	↑*6* ↓*21*	↑*10* ↓*18*	↑*12* ↓*18*	↑*12* ↓*18*	↑*50* ↓*158*
F_4	(30)	↑*5* ↓*18*	↑*3* ↓*20*	↑*3* ↓*22*	↑*3* ↓*27*	↑*2* ↓*25*	↑*3* ↓*25*	↑*5* ↓*22*	↑**12** ↓**9**	↑*36* ↓*168*
F_5	(30)	↑**21** ↓**3**	↑**16** ↓**14**	↑15 ↓15	↑*9* ↓*21*	↑*6* ↓*21*	↑*6* ↓*22*	↑*6* ↓*24*	↑*5* ↓*25*	↑*84* ↓*145*
F_6	(30)	↑*6* ↓*16*	↑*6* ↓*21*	↑*2* ↓*25*	↑*0* ↓*30*	↑*0* ↓*29*	↑*0* ↓*29*	↑*0* ↓*29*	↑*0* ↓*28*	↑*14* ↓*207*
T_1	(35)	↑**15** ↓**10**	↑*2* ↓*30*	↑*2* ↓*32*	↑*2* ↓*33*	↑*2* ↓*32*	↑*6* ↓*26*	↑*12* ↓*21*	↑**18** ↓**16**	↑*59* ↓*200*
T_2	(35)	↑*6* ↓*15*	↑*9* ↓*22*	↑*4* ↓*29*	↑*3* ↓*31*	↑*4* ↓*28*	↑*8* ↓*23*	↑*13* ↓*22*	↑**16** ↓**15**	↑*63* ↓*185*
T_3	(35)	↑*8* ↓*15*	↑*10* ↓*18*	↑*6* ↓*21*	↑*6* ↓*25*	↑*5* ↓*24*	↑*10* ↓*23*	↑*13* ↓*19*	↑**16** ↓**14**	↑*74* ↓*159*
T_4	(35)	↑*3* ↓*28*	↑*2* ↓*30*	↑*2* ↓*31*	↑*2* ↓*33*	↑*3* ↓*30*	↑*8* ↓*27*	↑*11* ↓*22*	↑*15* ↓*17*	↑*46* ↓*218*
T_5	(35)	↑*4* ↓*19*	↑*7* ↓*16*	↑*7* ↓*23*	↑*8* ↓*26*	↑*7* ↓*22*	↑*12* ↓*18*	↑*16* ↓*17*	↑**19** ↓**13**	↑*80* ↓*154*
T_6	(35)	↑*10* ↓*18*	↑*9* ↓*22*	↑*6* ↓*25*	↑*4* ↓*30*	↑*5* ↓*28*	↑*8* ↓*26*	↑*14* ↓*20*	↑**17** ↓**15**	↑*73* ↓*184*
All	(270)	↑*79* ↓*117*	↑*79* ↓*146*	↑*63* ↓*180*	↑*52* ↓*208*	↑*49* ↓*197*	↑*75* ↓*175*	↑*103* ↓*153*	↑**124** ↓**118**	↑*624* ↓*1294*

Table D.9: SADynPopDE vs SACDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑2 ↓14
5	(2)	**↑2 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓2	↑3 ↓10
10	(2)	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑1 ↓1	↑0 ↓1	↑5 ↓6
20	(2)	↑0 ↓0	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑4 ↓5
40	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓1	↑0 ↓1	**↑7 ↓3**
80	(2)	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑10 ↓1**
C	(6)	**↑3 ↓0**	**↑5 ↓1**	**↑4 ↓1**	**↑3 ↓2**	↑1 ↓2	↑1 ↓1	↑1 ↓1	↑0 ↓3	**↑18 ↓11**
S	(6)	**↑3 ↓0**	**↑3 ↓2**	↑2 ↓3	↑1 ↓5	↑1 ↓4	↑1 ↓4	↑1 ↓5	↑1 ↓5	↑13 ↓28
		GBG								
F_{1a}	(6)	**↑4 ↓0**	**↑6 ↓0**	**↑4 ↓0**	↑2 ↓2	↑2 ↓3	↑2 ↓3	↑1 ↓3	↑0 ↓3	**↑21 ↓14**
F_{1b}	(6)	**↑3 ↓0**	↑3 ↓3	↑3 ↓3	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑1 ↓3	**↑4 ↓2**	↑14 ↓26
F_2	(6)	**↑3 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓2**	**↑4 ↓2**	**↑3 ↓1**	**↑3 ↓2**	**↑31 ↓8**
F_3	(6)	**↑1 ↓0**	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑1 ↓40
F_4	(6)	**↑5 ↓0**	**↑3 ↓1**	**↑4 ↓1**	↑1 ↓5	↑1 ↓5	↑1 ↓5	↑0 ↓5	↑0 ↓6	↑15 ↓28
F_5	(6)	**↑4 ↓0**	↑0 ↓5	↑0 ↓6	↑0 ↓4	↑0 ↓3	↑0 ↓3	↑0 ↓3	↑2 ↓3	↑6 ↓27
F_6	(6)	**↑2 ↓0**	**↑1 ↓0**	↑0 ↓3	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑3 ↓33
T_1	(7)	**↑7 ↓0**	↑1 ↓4	↑0 ↓5	↑0 ↓7	↑0 ↓7	↑0 ↓7	↑1 ↓6	↑1 ↓6	↑10 ↓42
T_2	(7)	**↑4 ↓0**	**↑4 ↓2**	**↑4 ↓3**	↑2 ↓5	↑2 ↓5	↑1 ↓5	↑1 ↓4	↑2 ↓3	↑20 ↓27
T_3	(7)	**↑1 ↓0**	**↑5 ↓2**	**↑4 ↓2**	↑1 ↓3	↑1 ↓3	↑2 ↓3	↑1 ↓3	↑3 ↓3	↑18 ↓19
T_4	(7)	**↑6 ↓0**	↑2 ↓3	↑1 ↓4	↑0 ↓6	↑0 ↓7	↑0 ↓7	↑0 ↓6	↑1 ↓6	↑10 ↓39
T_5	(7)	**↑3 ↓0**	**↑3 ↓0**	**↑4 ↓2**	↑3 ↓4	↑3 ↓3	↑3 ↓3	↑2 ↓3	↑1 ↓4	**↑22 ↓19**
T_6	(7)	**↑1 ↓0**	↑3 ↓3	↑3 ↓3	↑1 ↓4	↑1 ↓5	↑1 ↓5	↑0 ↓5	↑1 ↓5	↑11 ↓30
All	(54)	**↑28 ↓0**	**↑26 ↓17**	↑22 ↓23	↑11 ↓36	↑9 ↓36	↑9 ↓35	↑7 ↓33	↑10 ↓35	↑122 ↓215
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑2 ↓7
5	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑1 ↓14
10	(2)	↑0 ↓0	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓14
20	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑1 ↓9
40	(2)	**↑1 ↓0**	↑0 ↓2	↑1 ↓1	↑1 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑3 ↓8
80	(2)	**↑1 ↓0**	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑0 ↓1	↑0 ↓1	↑5 ↓7
C	(6)	**↑1 ↓0**	↑0 ↓5	↑2 ↓3	↑2 ↓3	↑1 ↓3	↑1 ↓4	↑0 ↓3	↑0 ↓3	↑7 ↓24
S	(6)	**↑5 ↓0**	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑5 ↓35
		GBG								
F_{1a}	(6)	**↑2 ↓0**	↑0 ↓5	↑0 ↓3	↑0 ↓3	↑0 ↓3	↑0 ↓3	↑0 ↓3	↑0 ↓4	↑2 ↓24
F_{1b}	(6)	**↑3 ↓0**	↑0 ↓5	↑0 ↓4	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑2 ↓2	↑5 ↓31
F_2	(6)	**↑3 ↓0**	↑0 ↓2	↑1 ↓2	↑3 ↓3	↑3 ↓3	↑3 ↓3	↑2 ↓3	↑2 ↓3	↑17 ↓19
F_3	(6)	**↑3 ↓0**	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓4	↑3 ↓39
F_4	(6)	**↑3 ↓0**	↑0 ↓2	↑0 ↓2	↑3 ↓3	↑3 ↓3	↑2 ↓3	↑2 ↓3	↑1 ↓4	↑14 ↓20
F_5	(6)	**↑3 ↓0**	↑2 ↓2	↑2 ↓2	**↑3 ↓2**	↑3 ↓3	**↑3 ↓2**	**↑3 ↓2**	**↑3 ↓2**	**↑22 ↓15**
F_6	(6)	**↑2 ↓0**	↑0 ↓4	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑2 ↓40
T_1	(7)	**↑7 ↓0**	↑0 ↓7	↑0 ↓7	↑0 ↓7	↑0 ↓7	↑0 ↓7	↑0 ↓7	↑1 ↓6	↑8 ↓48
T_2	(7)	**↑2 ↓0**	↑0 ↓4	↑0 ↓3	↑2 ↓4	↑2 ↓4	↑1 ↓4	↑0 ↓4	↑0 ↓3	↑7 ↓26
T_3	(7)	**↑2 ↓0**	↑1 ↓2	↑1 ↓2	**↑3 ↓2**	**↑3 ↓2**	**↑3 ↓2**	**↑3 ↓2**	↑2 ↓2	**↑18 ↓14**
T_4	(7)	**↑3 ↓0**	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓7	↑0 ↓6	↑0 ↓6	↑1 ↓5	↑4 ↓41
T_5	(7)	**↑3 ↓0**	↑1 ↓2	↑2 ↓3	↑3 ↓3	↑3 ↓3	↑3 ↓3	↑3 ↓3	↑3 ↓3	**↑21 ↓20**
T_6	(7)	**↑2 ↓0**	↑0 ↓5	↑0 ↓4	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑7 ↓39
All	(54)	**↑25 ↓0**	↑2 ↓35	↑5 ↓33	↑11 ↓36	↑10 ↓37	↑9 ↓37	↑7 ↓36	↑8 ↓33	↑77 ↓247
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑0 ↓1	↑4 ↓8
5	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑1 ↓14
10	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑1 ↓14
20	(2)	↑0 ↓0	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓14
40	(2)	↑0 ↓0	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑0 ↓12
80	(2)	**↑1 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑1 ↓10
C	(6)	↑0 ↓0	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓4	↑0 ↓5	↑0 ↓4	↑0 ↓36
S	(6)	**↑4 ↓0**	↑0 ↓6	↑0 ↓5	↑0 ↓5	↑1 ↓5	↑1 ↓5	↑1 ↓5	↑0 ↓5	↑7 ↓36
		GBG								
F_{1a}	(6)	**↑3 ↓0**	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓5	↑0 ↓4	↑0 ↓4	↑3 ↓35
F_{1b}	(6)	**↑5 ↓0**	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓5	↑0 ↓4	↑5 ↓38
F_2	(6)	**↑5 ↓0**	↑0 ↓5	↑0 ↓4	↑0 ↓6	↑0 ↓6	↑0 ↓5	↑0 ↓5	↑0 ↓3	↑5 ↓34
F_3	(6)	**↑3 ↓0**	↑0 ↓6	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑3 ↓41
F_4	(6)	**↑5 ↓0**	↑0 ↓4	↑0 ↓5	↑0 ↓6	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓4	↑5 ↓34
F_5	(6)	↑0 ↓1	↑0 ↓3	↑1 ↓2	**↑2 ↓1**	**↑3 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑24 ↓7**
F_6	(6)	**↑3 ↓0**	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑0 ↓6	↑3 ↓42
T_1	(7)	**↑6 ↓0**	↑0 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑12 ↓42
T_2	(7)	**↑4 ↓0**	↑0 ↓7	↑0 ↓7	↑0 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓4	↑1 ↓3	↑8 ↓39
T_3	(7)	**↑4 ↓0**	↑0 ↓6	↑0 ↓5	↑0 ↓7	↑0 ↓5	↑1 ↓4	↑1 ↓5	↑1 ↓2	↑7 ↓34
T_4	(7)	**↑3 ↓1**	↑0 ↓6	↑0 ↓7	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑8 ↓44
T_5	(7)	**↑3 ↓0**	↑0 ↓4	↑0 ↓4	↑0 ↓6	↑0 ↓5	↑1 ↓4	↑1 ↓4	↑1 ↓4	↑6 ↓31
T_6	(7)	**↑4 ↓0**	↑0 ↓6	↑0 ↓5	↑0 ↓6	↑0 ↓6	↑1 ↓6	↑1 ↓6	↑1 ↓6	↑7 ↓41
All	(54)	**↑28 ↓1**	↑0 ↓47	↑1 ↓45	↑2 ↓48	↑4 ↓44	↑7 ↓41	↑7 ↓41	↑6 ↓36	↑55 ↓303

Table D.10: SADynPopDE vs SACDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑1 ↓11*
5	(2)	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑1 ↓14*
10	(2)	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑1 ↓14*
20	(2)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓14*
40	(2)	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑1 ↓14*
80	(2)	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓14*
C	(6)	**↑1 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓42*
S	(6)	**↑5 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑5 ↓39*
		GBG								
F_{1a}	(6)	**↑2 ↓0**	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓4*	*↑2 ↓36*
F_{1b}	(6)	**↑2 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑2 ↓41*
F_2	(6)	**↑3 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑3 ↓42*
F_3	(6)	**↑4 ↓0**	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑4 ↓41*
F_4	(6)	**↑4 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑4 ↓41*
F_5	(6)	↑1 ↓1	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓3*	*↑1 ↓2*	*↑1 ↓2*	**↑2 ↓1**	**↑2 ↓0**	*↑7 ↓12*
F_6	(6)	**↑2 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑2 ↓42*
T_1	(7)	**↑2 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑1 ↓6*	*↑6 ↓42*
T_2	(7)	**↑2 ↓0**	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑2 ↓47*
T_3	(7)	**↑2 ↓0**	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓4*	*↑0 ↓5*	*↑2 ↓38*
T_4	(7)	**↑4 ↓1**	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑1 ↓6*	*↑1 ↓6*	*↑6 ↓47*
T_5	(7)	**↑1 ↓0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓4*	*↑1 ↓39*
T_6	(7)	**↑7 ↓0**	*↑0 ↓7*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑7 ↓42*
All	(54)	**↑24 ↓1**	*↑0 ↓48*	*↑0 ↓47*	*↑0 ↓51*	*↑1 ↓50*	*↑1 ↓49*	*↑2 ↓46*	*↑2 ↓44*	*↑30 ↓336*
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	**↑2 ↓0**	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑8 ↓8
5	(2)	↑0 ↓0	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑6 ↓8*
10	(2)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑5 ↓9*
20	(2)	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑6 ↓8*
40	(2)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑4 ↓7*
80	(2)	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑1 ↓7*
C	(6)	**↑2 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑2 ↓42*
S	(6)	**↑1 ↓0**	*↑0 ↓3*	↑2 ↓2	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑28 ↓5**
		GBG								
F_{1a}	(6)	**↑1 ↓0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓41*
F_{1b}	(6)	↑1 ↓1	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓41*
F_2	(6)	*↑0 ↓1*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓43*
F_3	(6)	**↑3 ↓0**	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑3 ↓40*
F_4	(6)	**↑1 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓42*
F_5	(6)	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓4*	*↑1 ↓17*
F_6	(6)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓42*
T_1	(7)	**↑1 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓42*
T_2	(7)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓48*
T_3	(7)	↑1 ↓1	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑1 ↓43*
T_4	(7)	*↑1 ↓2*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑1 ↓50*
T_5	(7)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓43*
T_6	(7)	**↑4 ↓0**	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓7*	*↑4 ↓40*
All	(54)	**↑10 ↓3**	*↑0 ↓42*	*↑2 ↓44*	*↑4 ↓44*	*↑5 ↓44*	*↑6 ↓44*	*↑5 ↓46*	*↑5 ↓46*	*↑37 ↓313*
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	**↑8 ↓0**	*↑0 ↓9*	*↑1 ↓7*	*↑1 ↓7*	*↑2 ↓7*	*↑2 ↓6*	*↑2 ↓6*	*↑1 ↓6*	*↑17 ↓48*
5	(10)	**↑5 ↓0**	*↑1 ↓9*	*↑1 ↓8*	*↑1 ↓9*	*↑1 ↓9*	*↑1 ↓8*	*↑1 ↓8*	*↑1 ↓9*	*↑12 ↓60*
10	(10)	**↑4 ↓0**	*↑1 ↓8*	*↑1 ↓9*	*↑1 ↓8*	*↑1 ↓8*	*↑1 ↓8*	*↑2 ↓8*	*↑1 ↓8*	*↑12 ↓57*
20	(10)	**↑2 ↓0**	*↑2 ↓7*	*↑1 ↓7*	*↑2 ↓7*	*↑1 ↓7*	*↑1 ↓8*	*↑1 ↓7*	*↑1 ↓7*	*↑11 ↓50*
40	(10)	**↑2 ↓0**	*↑2 ↓7*	*↑3 ↓6*	*↑2 ↓7*	*↑2 ↓6*	*↑2 ↓5*	*↑1 ↓7*	*↑1 ↓6*	*↑15 ↓44*
80	(10)	**↑4 ↓0**	*↑2 ↓6*	*↑3 ↓6*	*↑3 ↓6*	*↑2 ↓5*	*↑3 ↓5*	*↑1 ↓5*	*↑1 ↓6*	*↑19 ↓39*
C	(30)	**↑7 ↓0**	*↑5 ↓24*	*↑6 ↓22*	*↑5 ↓23*	*↑2 ↓22*	*↑2 ↓21*	*↑1 ↓21*	*↑0 ↓22*	*↑28 ↓155*
S	(30)	**↑18 ↓0**	*↑3 ↓22*	*↑4 ↓21*	*↑5 ↓21*	*↑7 ↓20*	*↑8 ↓19*	*↑7 ↓20*	*↑6 ↓20*	*↑58 ↓143*
		GBG								
F_{1a}	(30)	**↑12 ↓0**	*↑6 ↓19*	*↑4 ↓20*	*↑2 ↓23*	*↑2 ↓23*	*↑2 ↓23*	*↑1 ↓21*	*↑0 ↓21*	*↑29 ↓150*
F_{1b}	(30)	**↑14 ↓1**	*↑3 ↓24*	*↑3 ↓25*	*↑0 ↓28*	*↑0 ↓28*	*↑0 ↓27*	*↑1 ↓24*	*↑6 ↓20*	*↑27 ↓177*
F_2	(30)	**↑14 ↓1**	*↑5 ↓19*	*↑6 ↓18*	*↑7 ↓22*	*↑7 ↓23*	*↑7 ↓22*	*↑5 ↓21*	*↑5 ↓20*	*↑56 ↓146*
F_3	(30)	**↑14 ↓0**	*↑0 ↓27*	*↑0 ↓27*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓27*	*↑14 ↓201*
F_4	(30)	**↑18 ↓0**	*↑3 ↓19*	*↑4 ↓20*	*↑4 ↓26*	*↑4 ↓25*	*↑3 ↓25*	*↑2 ↓25*	*↑1 ↓25*	*↑39 ↓165*
F_5	(30)	**↑9 ↓3**	*↑2 ↓13*	*↑3 ↓12*	*↑5 ↓12*	*↑7 ↓10*	**↑10 ↓9**	**↑11 ↓10**	**↑13 ↓9**	*↑60 ↓78*
F_6	(30)	**↑9 ↓0**	*↑1 ↓22*	*↑0 ↓27*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑0 ↓30*	*↑10 ↓199*
T_1	(35)	**↑23 ↓0**	*↑1 ↓29*	*↑1 ↓30*	*↑1 ↓32*	*↑2 ↓32*	*↑2 ↓32*	*↑3 ↓31*	*↑4 ↓30*	*↑37 ↓216*
T_2	(35)	**↑12 ↓0**	*↑4 ↓26*	*↑4 ↓26*	*↑4 ↓29*	*↑5 ↓29*	*↑3 ↓29*	*↑2 ↓26*	*↑3 ↓22*	*↑37 ↓187*
T_3	(35)	**↑10 ↓1**	*↑6 ↓19*	*↑5 ↓20*	*↑4 ↓25*	*↑4 ↓22*	*↑6 ↓21*	*↑5 ↓21*	*↑6 ↓19*	*↑46 ↓148*
T_4	(35)	**↑17 ↓4**	*↑2 ↓27*	*↑1 ↓30*	*↑1 ↓32*	*↑1 ↓34*	*↑1 ↓33*	*↑2 ↓31*	*↑4 ↓30*	*↑29 ↓221*
T_5	(35)	**↑10 ↓0**	*↑4 ↓17*	*↑6 ↓21*	*↑6 ↓25*	*↑6 ↓23*	*↑7 ↓22*	*↑6 ↓23*	*↑5 ↓21*	*↑50 ↓152*
T_6	(35)	**↑18 ↓0**	*↑3 ↓25*	*↑3 ↓22*	*↑2 ↓28*	*↑2 ↓29*	*↑3 ↓29*	*↑2 ↓29*	*↑3 ↓30*	*↑36 ↓192*
All	(270)	**↑115 ↓5**	*↑28 ↓189*	*↑30 ↓192*	*↑28 ↓215*	*↑29 ↓211*	*↑32 ↓206*	*↑28 ↓202*	*↑31 ↓194*	*↑321 ↓1414*

Table D.11: DynPopDE vs SACDE performance analysis on the n_p standard set

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
10	(2)	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	**↑7 ↓6**
25	(2)	**↑1 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
50	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
100	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
200	(2)	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
C	(6)	**↑2 ↓0**	**↑2 ↓1**	**↑4 ↓1**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑36 ↓2**
S	(6)	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑41 ↓6**
All	(12)	**↑8 ↓0**	**↑7 ↓1**	**↑9 ↓2**	**↑10 ↓1**	**↑11 ↓1**	**↑11 ↓1**	**↑11 ↓1**	**↑10 ↓1**	**↑77 ↓8**
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓0	**↑13 ↓0**
10	(2)	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑0 ↓0	*↑0 ↓2*	*↑1 ↓4*
25	(2)	**↑1 ↓0**	**↑1 ↓0**	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑12 ↓0**
50	(2)	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓2**
100	(2)	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
200	(2)	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
C	(6)	**↑1 ↓0**	*↑1 ↓3*	↑1 ↓1	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑27 ↓5**
S	(6)	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	↑5 ↓0	↑5 ↓1	↑5 ↓1	↑5 ↓0	**↑4 ↓1**	**↑39 ↓3**
All	(12)	**↑7 ↓0**	**↑6 ↓3**	**↑5 ↓1**	**↑10 ↓0**	**↑10 ↓1**	**↑10 ↓1**	**↑10 ↓0**	**↑8 ↓2**	**↑66 ↓8**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑4 ↓6*
10	(2)	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	*↑2 ↓8*
25	(2)	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	↑6 ↓6
50	(2)	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑6 ↓4**
100	(2)	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓2*	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑8 ↓4**
200	(2)	**↑1 ↓0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑7 ↓3**
C	(6)	**↑3 ↓0**	↑1 ↓1	*↑0 ↓4*	↑1 ↓1	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑23 ↓7**
S	(6)	**↑5 ↓0**	*↑0 ↓1*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓4*	*↑0 ↓3*	*↑1 ↓2*	**↑4 ↓2**	*↑10 ↓24*
All	(12)	**↑8 ↓0**	*↑1 ↓2*	*↑0 ↓10*	*↑1 ↓7*	**↑5 ↓4**	**↑5 ↓3**	**↑5 ↓3**	**↑8 ↓2**	**↑33 ↓31**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑2 ↓7*
10	(2)	**↑2 ↓0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑2 ↓10*
25	(2)	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓9*
50	(2)	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑5 ↓10*
100	(2)	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓2*	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑6 ↓9*
200	(2)	**↑2 ↓0**	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑5 ↓9*
C	(6)	**↑5 ↓0**	*↑1 ↓4*	*↑0 ↓1*	*↑0 ↓4*	**↑1 ↓0**	**↑4 ↓1**	**↑3 ↓1**	**↑3 ↓1**	**↑17 ↓12**
S	(6)	**↑6 ↓0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑6 ↓42*
All	(12)	**↑11 ↓0**	*↑1 ↓10*	*↑0 ↓7*	*↑0 ↓10*	*↑1 ↓6*	*↑4 ↓7*	*↑3 ↓7*	*↑3 ↓7*	*↑23 ↓54*
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑8 ↓3**
10	(2)	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑9 ↓4**
25	(2)	**↑2 ↓0**	↑1 ↓1	*↑0 ↓1*	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	**↑7 ↓4**
50	(2)	**↑2 ↓0**	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	**↑3 ↓1**
100	(2)	**↑2 ↓0**	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑0 ↓0	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑9 ↓2**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓2**
C	(6)	**↑5 ↓0**	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓2*	↑0 ↓0	**↑2 ↓0**	*↑2 ↓3*	↑3 ↓3	*↑13 ↓16*
S	(6)	**↑6 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑3 ↓0**	**↑36 ↓0**
All	(12)	**↑11 ↓0**	**↑5 ↓3**	*↑4 ↓5*	**↑5 ↓2**	**↑4 ↓0**	**↑7 ↓0**	**↑7 ↓3**	**↑6 ↓3**	**↑49 ↓16**
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	**↑6 ↓0**	**↑7 ↓2**	**↑5 ↓2**	**↑5 ↓2**	**↑6 ↓2**	**↑6 ↓2**	**↑5 ↓3**	↑3 ↓3	**↑43 ↓16**
10	(10)	**↑9 ↓0**	*↑2 ↓3*	*↑2 ↓4*	*↑1 ↓3*	*↑2 ↓4*	*↑2 ↓5*	*↑2 ↓6*	*↑1 ↓7*	*↑21 ↓32*
25	(10)	**↑7 ↓0**	*↑3 ↓4*	*↑2 ↓4*	**↑5 ↓4**	**↑6 ↓2**	**↑7 ↓2**	**↑6 ↓2**	**↑6 ↓2**	**↑42 ↓20**
50	(10)	**↑7 ↓0**	*↑2 ↓3*	*↑3 ↓6*	**↑4 ↓3**	**↑5 ↓2**	**↑6 ↓1**	**↑6 ↓1**	**↑8 ↓1**	**↑41 ↓17**
100	(10)	**↑9 ↓0**	*↑2 ↓4*	*↑3 ↓4*	**↑6 ↓4**	**↑6 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑50 ↓16**
200	(10)	**↑7 ↓0**	**↑4 ↓3**	*↑3 ↓5*	**↑5 ↓4**	**↑6 ↓1**	**↑8 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑51 ↓16**
C	(30)	**↑16 ↓0**	*↑6 ↓12*	*↑5 ↓12*	**↑11 ↓7**	**↑17 ↓0**	**↑22 ↓1**	**↑20 ↓5**	**↑19 ↓5**	**↑116 ↓42**
S	(30)	**↑29 ↓0**	**↑14 ↓7**	↑13 ↓13	**↑15 ↓13**	**↑14 ↓12**	**↑15 ↓11**	**↑16 ↓9**	**↑16 ↓10**	**↑132 ↓75**
All	(60)	**↑45 ↓0**	**↑20 ↓19**	*↑18 ↓25*	**↑26 ↓20**	**↑31 ↓12**	**↑37 ↓12**	**↑36 ↓14**	**↑35 ↓15**	**↑248 ↓117**

Table D.12: SACDE vs CDE performance analysis on the $n_p(t)$ standard set

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
n_p 10	(10)	**↑10 ↓0**	**↑9 ↓0**	**↑7 ↓0**	*↑2 ↓4*	*↑1 ↓3*	*↑0 ↓3*	*↑1 ↓2*	*↑1 ↓3*	**↑31 ↓15**
25	(10)	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑3 ↓0**	**↑45 ↓0**
50	(10)	**↑9 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑7 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑2 ↓0**	**↑3 ↓0**	**↑49 ↓0**
100	(10)	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑64 ↓0**
200	(10)	**↑9 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑10 ↓0**	**↑7 ↓0**	**↑8 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑63 ↓0**
pc 5	(10)	**↑9 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓2**	**↑2 ↓1**	↑1 ↓1	**↑2 ↓1**	**↑2 ↓1**	**↑33 ↓6**
10	(10)	**↑10 ↓0**	**↑8 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑3 ↓1**	**↑2 ↓1**	**↑1 ↓0**	**↑2 ↓0**	**↑36 ↓3**
20	(10)	**↑8 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑7 ↓1**	**↑4 ↓1**	**↑5 ↓1**	**↑3 ↓1**	**↑2 ↓1**	**↑49 ↓5**
40	(10)	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑6 ↓1**	**↑61 ↓1**
80	(10)	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑73 ↓0**
C	(25)	**↑23 ↓0**	**↑25 ↓0**	**↑24 ↓0**	**↑18 ↓3**	**↑17 ↓3**	**↑13 ↓3**	**↑12 ↓2**	**↑12 ↓3**	**↑144 ↓14**
S	(25)	**↑24 ↓0**	**↑20 ↓0**	**↑18 ↓0**	**↑15 ↓1**	**↑8 ↓0**	**↑9 ↓0**	**↑6 ↓0**	**↑8 ↓0**	**↑108 ↓1**
All	(50)	**↑47 ↓0**	**↑45 ↓0**	**↑42 ↓0**	**↑33 ↓4**	**↑25 ↓3**	**↑22 ↓3**	**↑18 ↓2**	**↑20 ↓3**	**↑252 ↓15**
Set.	**Max**	**10 Dimensions**								
n_p 10	(10)	**↑9 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑73 ↓0**
25	(10)	**↑6 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑7 ↓0**	**↑70 ↓0**
50	(10)	**↑7 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑5 ↓1**	**↑66 ↓1**
100	(10)	**↑8 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑71 ↓0**
200	(10)	**↑8 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑7 ↓0**	**↑6 ↓0**	**↑66 ↓0**
pc 5	(10)	**↑9 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓1**	**↑58 ↓1**
10	(10)	**↑8 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑6 ↓0**	**↑7 ↓0**	**↑3 ↓0**	**↑61 ↓0**
20	(10)	**↑8 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑7 ↓0**	**↑74 ↓0**
40	(10)	**↑6 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓0**
80	(10)	**↑7 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑77 ↓0**
C	(25)	**↑14 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑21 ↓0**	**↑19 ↓0**	**↑18 ↓0**	**↑19 ↓0**	**↑14 ↓1**	**↑155 ↓1**
S	(25)	**↑24 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑24 ↓0**	**↑23 ↓0**	**↑20 ↓0**	**↑191 ↓0**
All	(50)	**↑38 ↓0**	**↑50 ↓0**	**↑50 ↓0**	**↑46 ↓0**	**↑44 ↓0**	**↑42 ↓0**	**↑42 ↓0**	**↑34 ↓1**	**↑346 ↓1**
Set.	**Max**	**25 Dimensions**								
n_p 10	(10)	**↑5 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑70 ↓1**
25	(10)	*↑1 ↓2*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑69 ↓2**
50	(10)	**↑3 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑10 ↓0**	**↑70 ↓1**
100	(10)	**↑2 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑68 ↓0**
200	(10)	*↑1 ↓2*	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑66 ↓2**
pc 5	(10)	**↑4 ↓2**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑8 ↓0**	**↑7 ↓0**	**↑62 ↓2**
10	(10)	**↑3 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑6 ↓0**	**↑10 ↓0**	**↑67 ↓1**
20	(10)	**↑2 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑71 ↓1**
40	(10)	*↑1 ↓2*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑71 ↓2**
80	(10)	**↑2 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑72 ↓0**
C	(25)	*↑0 ↓6*	**↑25 ↓0**	**↑25 ↓0**	**↑24 ↓0**	**↑22 ↓0**	**↑20 ↓0**	**↑19 ↓0**	**↑21 ↓0**	**↑156 ↓6**
S	(25)	**↑12 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑187 ↓0**
All	(50)	**↑12 ↓6**	**↑50 ↓0**	**↑50 ↓0**	**↑49 ↓0**	**↑47 ↓0**	**↑45 ↓0**	**↑44 ↓0**	**↑46 ↓0**	**↑343 ↓6**
Set.	**Max**	**50 Dimensions**								
n_p 10	(10)	*↑0 ↓5*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑67 ↓5**
25	(10)	*↑0 ↓6*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑10 ↓0**	**↑68 ↓6**
50	(10)	*↑0 ↓3*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑66 ↓3**
100	(10)	*↑0 ↓3*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑67 ↓3**
200	(10)	*↑1 ↓6*	**↑9 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑10 ↓0**	**↑65 ↓6**
pc 5	(10)	*↑1 ↓4*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑8 ↓0**	**↑6 ↓0**	**↑10 ↓0**	**↑63 ↓4**
10	(10)	*↑0 ↓4*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑7 ↓0**	**↑9 ↓0**	**↑8 ↓0**	**↑64 ↓4**
20	(10)	*↑0 ↓4*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓0**	**↑10 ↓0**	**↑68 ↓4**
40	(10)	*↑0 ↓3*	**↑9 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑68 ↓3**
80	(10)	*↑0 ↓8*	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑70 ↓8**
C	(25)	*↑0 ↓15*	**↑24 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑23 ↓0**	**↑20 ↓0**	**↑17 ↓0**	**↑23 ↓0**	**↑157 ↓15**
S	(25)	*↑1 ↓8*	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑25 ↓0**	**↑176 ↓8**
All	(50)	*↑1 ↓23*	**↑49 ↓0**	**↑50 ↓0**	**↑50 ↓0**	**↑48 ↓0**	**↑45 ↓0**	**↑42 ↓0**	**↑48 ↓0**	**↑333 ↓23**
Set.	**Max**	**100 Dimensions**								
n_p 10	(10)	*↑0 ↓6*	*↑2 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑5 ↓5	*↑3 ↓5*	↑5 ↓5	*↑30 ↓41*
25	(10)	*↑0 ↓9*	*↑2 ↓5*	↑5 ↓5	*↑4 ↓5*	*↑4 ↓5*	↑5 ↓5	*↑4 ↓5*	**↑5 ↓3**	*↑29 ↓42*
50	(10)	*↑0 ↓10*	*↑0 ↓5*	*↑3 ↓5*	↑5 ↓5	↑5 ↓5	*↑3 ↓5*	*↑4 ↓5*	↑3 ↓3	*↑23 ↓43*
100	(10)	*↑0 ↓8*	*↑0 ↓5*	*↑4 ↓5*	↑5 ↓5	*↑4 ↓5*	*↑2 ↓5*	↑4 ↓4	**↑4 ↓3**	*↑23 ↓40*
200	(10)	*↑0 ↓8*	*↑0 ↓4*	**↑4 ↓2**	↑5 ↓5	*↑3 ↓4*	*↑2 ↓5*	**↑5 ↓2**	**↑5 ↓3**	*↑24 ↓33*
pc 5	(10)	*↑0 ↓9*	*↑2 ↓4*	**↑5 ↓4**	↑5 ↓5	*↑2 ↓4*	*↑2 ↓5*	*↑3 ↓4*	↑5 ↓5	*↑24 ↓40*
10	(10)	*↑0 ↓9*	*↑1 ↓5*	↑5 ↓5	*↑4 ↓5*	*↑4 ↓5*	*↑2 ↓5*	*↑4 ↓5*	**↑4 ↓3**	*↑24 ↓42*
20	(10)	*↑0 ↓8*	*↑1 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	*↑3 ↓5*	*↑3 ↓4*	*↑3 ↓4*	*↑25 ↓41*
40	(10)	*↑0 ↓8*	*↑0 ↓5*	*↑3 ↓4*	↑5 ↓5	↑5 ↓5	↑5 ↓5	**↑5 ↓3**	**↑5 ↓2**	*↑28 ↓37*
80	(10)	*↑0 ↓7*	*↑0 ↓5*	*↑3 ↓4*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑5 ↓5	**↑5 ↓3**	*↑28 ↓39*
C	(25)	*↑0 ↓17*	**↑4 ↓0**	**↑21 ↓0**	**↑24 ↓0**	**↑21 ↓0**	**↑17 ↓0**	**↑20 ↓0**	**↑22 ↓0**	**↑129 ↓17**
S	(25)	*↑0 ↓24*	*↑0 ↓24*	*↑0 ↓22*	*↑0 ↓25*	*↑0 ↓24*	*↑0 ↓25*	*↑0 ↓21*	*↑0 ↓17*	*↑0 ↓182*
All	(50)	*↑0 ↓41*	*↑4 ↓24*	*↑21 ↓22*	*↑24 ↓25*	*↑21 ↓24*	*↑17 ↓25*	*↑20 ↓21*	**↑22 ↓17**	*↑129 ↓199*
Set.	**Max**	**All Dimensions**								
n_p 10	(50)	**↑24 ↓12**	**↑41 ↓5**	**↑42 ↓5**	**↑37 ↓9**	**↑35 ↓8**	**↑31 ↓8**	**↑30 ↓7**	**↑31 ↓8**	**↑271 ↓62**
25	(50)	↑17 ↓17	**↑41 ↓5**	**↑43 ↓5**	**↑38 ↓5**	**↑37 ↓5**	**↑36 ↓5**	**↑35 ↓5**	**↑34 ↓3**	**↑281 ↓50**
50	(50)	**↑19 ↓14**	**↑39 ↓5**	**↑43 ↓5**	**↑41 ↓5**	**↑36 ↓5**	**↑35 ↓5**	**↑31 ↓5**	**↑30 ↓4**	**↑274 ↓48**
100	(50)	**↑19 ↓11**	**↑40 ↓5**	**↑43 ↓5**	**↑44 ↓5**	**↑40 ↓5**	**↑33 ↓5**	**↑36 ↓4**	**↑38 ↓3**	**↑293 ↓43**
200	(50)	**↑19 ↓16**	**↑37 ↓4**	**↑42 ↓2**	**↑42 ↓5**	**↑37 ↓4**	**↑36 ↓5**	**↑34 ↓2**	**↑37 ↓3**	**↑284 ↓41**
pc 5	(50)	**↑23 ↓15**	**↑39 ↓4**	**↑41 ↓4**	**↑36 ↓7**	**↑24 ↓5**	**↑24 ↓6**	**↑25 ↓5**	**↑28 ↓7**	**↑240 ↓53**
10	(50)	**↑21 ↓14**	**↑39 ↓5**	**↑41 ↓5**	**↑36 ↓6**	**↑36 ↓6**	**↑25 ↓6**	**↑27 ↓5**	**↑27 ↓3**	**↑252 ↓50**
20	(50)	**↑18 ↓13**	**↑41 ↓5**	**↑45 ↓5**	**↑42 ↓6**	**↑39 ↓6**	**↑38 ↓6**	**↑33 ↓5**	**↑31 ↓5**	**↑287 ↓51**
40	(50)	**↑17 ↓13**	**↑39 ↓5**	**↑43 ↓4**	**↑43 ↓5**	**↑42 ↓5**	**↑41 ↓5**	**↑38 ↓3**	**↑41 ↓3**	**↑304 ↓43**
80	(50)	**↑19 ↓15**	**↑40 ↓5**	**↑43 ↓4**	**↑45 ↓5**	**↑44 ↓5**	**↑43 ↓5**	**↑43 ↓5**	**↑43 ↓3**	**↑320 ↓47**
C	(125)	*↑37 ↓38*	**↑103 ↓0**	**↑120 ↓0**	**↑112 ↓3**	**↑102 ↓3**	**↑88 ↓3**	**↑87 ↓2**	**↑92 ↓4**	**↑741 ↓53**
S	(125)	**↑61 ↓32**	**↑95 ↓24**	**↑93 ↓22**	**↑90 ↓26**	**↑83 ↓24**	**↑83 ↓25**	**↑79 ↓21**	**↑78 ↓17**	**↑662 ↓191**
All	(250)	**↑98 ↓70**	**↑198 ↓24**	**↑213 ↓22**	**↑202 ↓29**	**↑185 ↓27**	**↑171 ↓28**	**↑166 ↓23**	**↑170 ↓21**	**↑1403 ↓244**

Table D.13: SADynPopDE vs SACDE performance analysis on the $n_p(t)$ standard set

Cp		**100**	**500**	**1000**	**5000**	**10000**	**25000**	**50000**	**100000**	**Total**
Set.	**Max**	**5 Dimensions**								
n_p 10	(10)	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**76** ↓**0**
25	(10)	↑**7** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**71** ↓**0**
50	(10)	↑**3** ↓**0**	↑**1** ↓**0**	↑**3** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**57** ↓**0**
100	(10)	↑**3** ↓**0**	↑2 ↓2	↑1 ↓1	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**56** ↓**3**
200	(10)	↑**2** ↓**0**	*↑0 ↓2*	*↑0 ↓2*	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**52** ↓**4**
pc 5	(10)	↑**8** ↓**0**	↑**4** ↓**3**	↑**2** ↓**1**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**63** ↓**4**
10	(10)	↑**6** ↓**0**	↑**3** ↓**1**	↑**3** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**61** ↓**1**
20	(10)	↑**3** ↓**0**	↑**3** ↓**0**	↑**5** ↓**2**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**61** ↓**2**
40	(10)	↑**3** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**63** ↓**0**
80	(10)	↑**3** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**64** ↓**0**
C	(25)	↑**12** ↓**0**	↑**11** ↓**2**	↑**9** ↓**3**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**157** ↓**5**
S	(25)	↑**11** ↓**0**	↑**9** ↓**2**	↑**12** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**24** ↓**0**	↑**24** ↓**0**	↑**25** ↓**0**	↑**155** ↓**2**
All	(50)	↑**23** ↓**0**	↑**20** ↓**4**	↑**21** ↓**3**	↑**50** ↓**0**	↑**50** ↓**0**	↑**49** ↓**0**	↑**49** ↓**0**	↑**50** ↓**0**	↑**312** ↓**7**
Set.	**Max**	**10 Dimensions**								
n_p 10	(10)	↑**6** ↓**0**	*↑0 ↓2*	↑**5** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**49** ↓**2**
25	(10)	↑**6** ↓**0**	*↑0 ↓5*	*↑0 ↓3*	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**52** ↓**8**
50	(10)	↑**3** ↓**0**	*↑0 ↓6*	*↑0 ↓5*	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**50** ↓**11**
100	(10)	↑**1** ↓**0**	*↑0 ↓6*	*↑0 ↓7*	↑**6** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**47** ↓**13**
200	(10)	↑**3** ↓**1**	*↑0 ↓6*	*↑0 ↓7*	↑**2** ↓**1**	↑**8** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**41** ↓**15**
pc 5	(10)	↑**6** ↓**1**	*↑0 ↓4*	*↑1 ↓5*	↑**8** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**52** ↓**10**
10	(10)	↑**4** ↓**0**	*↑0 ↓5*	*↑1 ↓4*	↑**6** ↓**1**	↑**8** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**47** ↓**10**
20	(10)	↑**2** ↓**0**	*↑0 ↓4*	*↑0 ↓2*	↑**4** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**45** ↓**6**
40	(10)	↑**5** ↓**0**	*↑0 ↓5*	*↑1 ↓4*	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**52** ↓**9**
80	(10)	↑**2** ↓**0**	*↑0 ↓7*	*↑2 ↓7*	↑**5** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**43** ↓**14**
C	(25)	↑**5** ↓**0**	*↑0 ↓19*	*↑1 ↓16*	↑**12** ↓**0**	↑**22** ↓**0**	↑**23** ↓**0**	↑**22** ↓**0**	↑**21** ↓**0**	↑**106** ↓**35**
S	(25)	↑**14** ↓**1**	*↑0 ↓6*	*↑4 ↓6*	↑**18** ↓**1**	↑**24** ↓**0**	↑**24** ↓**0**	↑**24** ↓**0**	↑**25** ↓**0**	↑**133** ↓**14**
All	(50)	↑**19** ↓**1**	*↑0 ↓25*	*↑5 ↓22*	↑**30** ↓**1**	↑**46** ↓**0**	↑**47** ↓**0**	↑**46** ↓**0**	↑**46** ↓**0**	↑**239** ↓**49**
Set.	**Max**	**25 Dimensions**								
n_p 10	(10)	↑**5** ↓**0**	*↑0 ↓9*	*↑0 ↓9*	*↑2 ↓5*	↑**5** ↓**2**	↑**5** ↓**1**	↑**5** ↓**0**	↑**5** ↓**0**	↑**27** ↓**26**
25	(10)	↑**6** ↓**0**	*↑0 ↓7*	*↑0 ↓9*	*↑0 ↓6*	↑**3** ↓**2**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**25** ↓**24**
50	(10)	↑**3** ↓**0**	*↑0 ↓9*	*↑0 ↓9*	*↑0 ↓8*	↑2 ↓2	↑**4** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	*↑27 ↓28*
100	(10)	↑**5** ↓**0**	*↑1 ↓8*	*↑0 ↓7*	*↑0 ↓8*	↑2 ↓2	↑**5** ↓**1**	↑**8** ↓**0**	↑**9** ↓**0**	↑**30** ↓**26**
200	(10)	↑**3** ↓**0**	*↑0 ↓8*	*↑0 ↓9*	*↑0 ↓8*	*↑0 ↓6*	↑**4** ↓**2**	↑**7** ↓**1**	↑**9** ↓**0**	*↑23 ↓34*
pc 5	(10)	↑**5** ↓**0**	*↑1 ↓8*	*↑0 ↓9*	*↑1 ↓5*	↑3 ↓3	↑**7** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**33** ↓**25**
10	(10)	↑**4** ↓**0**	*↑0 ↓7*	*↑0 ↓8*	*↑0 ↓6*	↑**4** ↓**2**	↑**6** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**30** ↓**23**
20	(10)	↑**3** ↓**0**	*↑0 ↓7*	*↑0 ↓6*	*↑1 ↓7*	*↑2 ↓3*	↑**5** ↓**1**	↑**7** ↓**0**	↑**8** ↓**0**	↑**26** ↓**24**
40	(10)	↑**6** ↓**0**	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓8*	*↑2 ↓3*	↑**3** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	*↑24 ↓30*
80	(10)	↑**4** ↓**0**	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓9*	*↑1 ↓3*	*↑2 ↓3*	↑**5** ↓**1**	↑**7** ↓**0**	*↑19 ↓36*
C	(25)	↑**4** ↓**0**	*↑0 ↓25*	*↑0 ↓24*	*↑0 ↓25*	*↑0 ↓13*	↑3 ↓3	↑**10** ↓**0**	↑**16** ↓**0**	*↑33 ↓90*
S	(25)	↑**18** ↓**0**	*↑1 ↓16*	*↑0 ↓19*	*↑2 ↓10*	↑**12** ↓**1**	↑**20** ↓**1**	↑**23** ↓**1**	↑**23** ↓**0**	↑**99** ↓**48**
All	(50)	↑**22** ↓**0**	*↑1 ↓41*	*↑0 ↓43*	*↑2 ↓35*	*↑12 ↓14*	↑**23** ↓**4**	↑**33** ↓**1**	↑**39** ↓**0**	*↑132 ↓138*
Set.	**Max**	**50 Dimensions**								
n_p 10	(10)	↑**4** ↓**0**	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓9*	*↑0 ↓9*	*↑4 ↓5*	*↑3 ↓4*	↑**4** ↓**1**	*↑15 ↓48*
25	(10)	↑**4** ↓**0**	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓7*	*↑0 ↓5*	*↑1 ↓3*	↑**3** ↓**2**	*↑8 ↓47*
50	(10)	↑**3** ↓**0**	*↑0 ↓9*	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓7*	*↑0 ↓5*	↑1 ↓1	↑**5** ↓**0**	*↑9 ↓41*
100	(10)	↑**3** ↓**0**	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓9*	*↑0 ↓9*	*↑0 ↓7*	↑**2** ↓**1**	↑**5** ↓**0**	*↑10 ↓45*
200	(10)	↑**2** ↓**0**	*↑0 ↓8*	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓8*	*↑0 ↓6*	*↑1 ↓3*	↑**6** ↓**1**	*↑9 ↓45*
pc 5	(10)	↑**4** ↓**0**	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓9*	*↑0 ↓6*	*↑1 ↓5*	*↑2 ↓4*	↑**7** ↓**2**	*↑14 ↓46*
10	(10)	↑**3** ↓**0**	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓7*	*↑1 ↓4*	*↑1 ↓2*	↑**7** ↓**0**	*↑12 ↓43*
20	(10)	↑**2** ↓**0**	*↑0 ↓10*	*↑0 ↓8*	*↑0 ↓9*	*↑0 ↓7*	*↑1 ↓5*	↑1 ↓1	↑**5** ↓**0**	*↑9 ↓40*
40	(10)	↑**2** ↓**0**	*↑0 ↓9*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓10*	*↑1 ↓6*	↑**3** ↓**0**	↑**2** ↓**0**	*↑8 ↓45*
80	(10)	↑**5** ↓**0**	*↑0 ↓7*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓10*	*↑0 ↓8*	*↑1 ↓5*	↑2 ↓2	*↑8 ↓52*
C	(25)	↑**4** ↓**0**	*↑0 ↓21*	*↑0 ↓25*	*↑0 ↓25*	*↑0 ↓25*	*↑0 ↓23*	*↑1 ↓10*	↑**5** ↓**3**	*↑10 ↓132*
S	(25)	↑**12** ↓**0**	*↑0 ↓25*	*↑0 ↓23*	*↑0 ↓23*	*↑0 ↓15*	*↑4 ↓5*	↑**7** ↓**2**	↑**18** ↓**1**	*↑41 ↓94*
All	(50)	↑**16** ↓**0**	*↑0 ↓46*	*↑0 ↓48*	*↑0 ↓48*	*↑0 ↓40*	*↑4 ↓28*	*↑8 ↓12*	↑**23** ↓**4**	*↑51 ↓226*
Set.	**Max**	**100 Dimensions**								
n_p 10	(10)	↑**2** ↓**1**	*↑1 ↓8*	*↑1 ↓9*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑5 ↓5	*↑29 ↓43*
25	(10)	↑**4** ↓**0**	*↑0 ↓3*	*↑1 ↓7*	*↑3 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑5 ↓5	*↑28 ↓35*
50	(10)	↑**1** ↓**0**	*↑1 ↓2*	*↑0 ↓6*	*↑3 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑**5** ↓**4**	*↑25 ↓32*
100	(10)	↑**1** ↓**0**	*↑0 ↓4*	*↑0 ↓7*	*↑2 ↓5*	*↑4 ↓5*	↑5 ↓5	↑5 ↓5	↑**5** ↓**4**	*↑22 ↓35*
200	(10)	↑1 ↓1	*↑0 ↓6*	*↑0 ↓8*	*↑0 ↓5*	*↑2 ↓5*	↑5 ↓5	↑5 ↓5	↑**5** ↓**3**	*↑18 ↓38*
pc 5	(10)	↑**3** ↓**0**	*↑0 ↓7*	*↑2 ↓6*	*↑4 ↓5*	*↑4 ↓5*	↑5 ↓5	↑5 ↓5	↑**5** ↓**4**	*↑28 ↓37*
10	(10)	↑**1** ↓**0**	*↑2 ↓3*	*↑0 ↓6*	*↑4 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑**5** ↓**4**	*↑27 ↓33*
20	(10)	↑**2** ↓**0**	*↑0 ↓4*	*↑0 ↓8*	*↑3 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑**5** ↓**3**	*↑25 ↓35*
40	(10)	↑1 ↓1	*↑0 ↓4*	*↑0 ↓9*	*↑1 ↓5*	*↑4 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	*↑21 ↓39*
80	(10)	↑**2** ↓**1**	*↑0 ↓5*	*↑0 ↓8*	*↑1 ↓5*	*↑3 ↓5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	*↑21 ↓39*
C	(25)	↑**3** ↓**0**	*↑0 ↓10*	*↑0 ↓24*	*↑0 ↓25*	*↑0 ↓25*	*↑0 ↓25*	*↑0 ↓25*	*↑0 ↓21*	*↑3 ↓155*
S	(25)	↑**6** ↓**2**	*↑2 ↓13*	*↑2 ↓13*	↑**13** ↓**0**	↑**21** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**119** ↓**28**
All	(50)	↑**9** ↓**2**	*↑2 ↓23*	*↑2 ↓37*	*↑13 ↓25*	*↑21 ↓25*	↑25 ↓25	↑25 ↓25	↑**25** ↓**21**	*↑122 ↓183*
Set.	**Max**	**All Dimensions**								
n_p 10	(50)	↑**25** ↓**1**	*↑11 ↓29*	*↑16 ↓28*	↑**25** ↓**19**	↑**29** ↓**16**	↑**31** ↓**11**	↑**29** ↓**9**	↑**30** ↓**6**	↑**196** ↓**119**
25	(50)	↑**27** ↓**0**	*↑7 ↓25*	*↑8 ↓29*	*↑20 ↓21*	↑**27** ↓**14**	↑**30** ↓**10**	↑**31** ↓**8**	↑**34** ↓**7**	↑**184** ↓**114**
50	(50)	↑**13** ↓**0**	*↑2 ↓26*	*↑3 ↓29*	*↑20 ↓23*	↑**27** ↓**14**	↑**29** ↓**10**	↑**34** ↓**6**	↑**40** ↓**4**	↑**168** ↓**112**
100	(50)	↑**13** ↓**0**	*↑3 ↓29*	*↑1 ↓32*	*↑18 ↓22*	↑**26** ↓**16**	↑**30** ↓**13**	↑**35** ↓**6**	↑**39** ↓**4**	↑**165** ↓**122**
200	(50)	↑**11** ↓**2**	*↑0 ↓30*	*↑0 ↓35*	*↑12 ↓24*	↑**20** ↓**19**	↑**28** ↓**13**	↑**32** ↓**9**	↑**40** ↓**4**	↑**143** ↓**136**
pc 5	(50)	↑**26** ↓**1**	*↑5 ↓32*	*↑5 ↓31*	↑**23** ↓**19**	↑**27** ↓**14**	↑**31** ↓**10**	↑**34** ↓**9**	↑**39** ↓**6**	↑**190** ↓**122**
10	(50)	↑**18** ↓**0**	*↑5 ↓26*	*↑4 ↓28*	*↑20 ↓22*	↑**27** ↓**14**	↑**31** ↓**9**	↑**33** ↓**7**	↑**39** ↓**4**	↑**177** ↓**110**
20	(50)	↑**12** ↓**0**	*↑3 ↓25*	*↑5 ↓26*	*↑18 ↓21*	↑**27** ↓**15**	↑**31** ↓**11**	↑**32** ↓**6**	↑**38** ↓**3**	↑**166** ↓**107**
40	(50)	↑**17** ↓**1**	*↑5 ↓27*	*↑6 ↓33*	*↑18 ↓23*	↑**26** ↓**18**	↑**29** ↓**11**	↑**33** ↓**5**	↑**34** ↓**5**	↑**168** ↓**123**
80	(50)	↑**16** ↓**1**	*↑5 ↓29*	*↑8 ↓35*	*↑16 ↓24*	↑**22** ↓**18**	↑**26** ↓**16**	↑**29** ↓**11**	↑**33** ↓**7**	↑**155** ↓**141**
C	(125)	↑**28** ↓**0**	*↑11 ↓77*	*↑10 ↓92*	*↑37 ↓75*	*↑47 ↓63*	↑51 ↓51	↑**58** ↓**35**	↑**67** ↓**24**	*↑309 ↓417*
S	(125)	↑**61** ↓**3**	*↑12 ↓62*	*↑18 ↓61*	↑**58** ↓**34**	↑**82** ↓**16**	↑**97** ↓**6**	↑**103** ↓**3**	↑**116** ↓**1**	↑**547** ↓**186**
All	(250)	↑**89** ↓**3**	*↑23 ↓139*	*↑28 ↓153*	*↑95 ↓109*	↑**129** ↓**79**	↑**148** ↓**57**	↑**161** ↓**38**	↑**183** ↓**25**	↑**856** ↓**603**

Appendix E

List of Symbols

a	general counter
b	general counter
c	change counter
$\vec{c}_p$	change in p-th peak location
$\vec{d}$	average location of all individuals
d_j	j-th component of vector $\vec{d}$
$f_{description}$	underlying benchmark function
f_p	p-th peak / underlying benchmark function
$f_{p,opti}$	optimum value of the p-th underlying benchmark function
g	generation counter
h_p	height of p-th peak
i	individual index
j	dimension / component index
j_{rand}	randomly selected index
k	sub-population index
$\vec{l}_F$	location of the global optimum of dynamic function F
$\vec{l}_p$	location of optimum of function f_p
$l_{p,j}$	j-th component of location of optimum of function f_p
n_c	total number of changes in a dynamic environment
n_d	number of dimensions

n_{excess}	threshold of number of free swarms
n_{exp}	number of experimental environments
n_{free}	number of free swarms
n_g	number of generations
n_I	total number of individuals in population including sub-populations
$n_{I,k}$	number of individuals in k-th sub-population
n_k	number of sub-populations
n_p	number of peaks
n_s	number of samples
n_t	total number of function evaluations
$\vec{o}$	vector used in GDBG
p	peak / underlying function index
q	GDBG control parameter index
r	random number
$\vec{r}$	random vector
r_j	random number associated with dimension j
r_{brown}	Brownian radius
r_{dev}	deviation from which r_{brown} is selected
r_{excl}	exclusion radius
r_{conv}	convergence radius
$r_{pop,k}$	population radius of of k-th sub-population
s	GDBG selected dimensions
t	time, generation counter
$\vec{u}_i$	i-th DE trial vector
$u_{i,j}$	j-th component of i-th trial vector
$\vec{v}_i$	i-th DE mutant vector
$v_{i,j}$	j-th component of i-th mutant vector
w_p	width of p-th peak
$\vec{x}^*$	global optimum
$\vec{x}_i$	i-th individual in the population

$\vec{x}_{i,k}$	i-th individual in the k-th population
$x_{i,j}$	j-th component of i-th individual
$\vec{x}_{best}$	best individual in the population P_x since the last change in the environment
$\vec{x}_{best}(g)$	best individual in the population P_x in generation g
$\vec{x}_{best,k}$	best individual in sub-population P_k since the last change in the environment
$\vec{x}_{brown}$	Brownian individual
$\vec{y}_i$	personal best of i-th individual
$\hat{\vec{y}}_i$	global best
$\vec{z}$	general individual
$A_{i,k}$	age of i-th individual in k-th sub-population
B	basis function in MPB
$C(a,b)$	random number from Cauchy distribution with median a and scale value b
Cr	crossover factor
Cr_i	crossover factor associated with i-th individual
Cr_{new_i}	new crossover factor associated with i-th individual
C_s	change severity
Ct	change type
Cp	change period
D	diversity
Det	change detection strategy
D_{AP}	average diversity per sub-population
$E_{BC}(\vec{x}_i(t), c)$	error of $\vec{x}_i$ immediately before the c-th change in the environment
F	function to be optimised
F_{opti}	global optimum of function F
$\mathcal{F}$	scale factor
$\mathcal{F}_i$	scale factor associated with i-th individual
$\mathcal{F}_{new_i}$	new scale factor associated with i-th individual
$\mathcal{F}_l$	lower bound of scale factor in *jDE*
$\mathcal{F}_u$	range of scale factor changes in *jDE*
$\mathcal{G}$	DE variant greediness factor

H	performance measure
$H_{OE,a}$	average offline errors of an algorithm on environment a
L	Longest diagonal in the search space
M_{n_p}	maximum number of peaks
M_c	maximum fraction change in the number of peaks
$N(a,b)$	random number from normal distribution with mean a and deviation b
P_x	population
P_k	k-th sub-population
P_{ar}	archive sub-population
$\mathbf{R}$	rotation matrix
R_k	relative fitness of k-th sub-population
RE	relative error
S	stagnation function
$\mathcal{S}^{n_d}$	n_d dimensional search space
$\mathbf{T}$	transformation matrix
$\mathcal{T}$	set of all time steps
$U(a,b)$	random number between a and b sampled from an uniform distribution
$V_{max,F}$	upper search range bound of function F
$V_{min,F}$	lower search range bound of function F
V_{max,f_p}	upper search range bound of underlying function f_p
V_{min,f_p}	lower search range bound of underlying function f_p
$W_{best}(g)$	function value of the best individual within a window of ω generations
$W_{worst}(g)$	function value of the worst individual within a window of ω generations
α	method of selecting the DE base vector
β	number of DE difference vectors
γ	method of producing DE trial vectors
η	constant controlling gradient step sizes
λ	correlation in peak shifts
$\rho_1 \dots \rho_5$	parameters
ϱ	parameter

σ_{f_p}	stretch factor associated with f_p
ς	value used in GDBG
$\tau_1 \dots \tau_{14}$	parameters
$\vec{\phi}_\Omega$	vector of GDBG control parameter Ω
$\phi_{\Omega,q}$	q-th component GDBG control parameter Ω
$\phi_{\Omega,max}$	maximum value of the GDBG control parameter Ω
$\phi_{\Omega,min}$	minimum value of the GDBG control parameter Ω
$\phi_{\Omega,sev}$	severity by which the value of the q-th component GDBG control parameter Ω is changed
ϖ	parameter
ψ_p	weighting factor of p-th underlying function used in GDBG
ω	window size
Ω	GDBG control parameter
$\mathcal{P}$	performance value
$\mathcal{P}_k$	performance value of k-th sub-population

Appendix F

List of Abbreviations

API	Average Percentage Improvement
CDE	Competing Differential Evolution
CESO	Collaborative Evolutionary-Swarm Optimisation
CPE	Competitive Population Evaluation
DE	Differential Evolution
DOP	Dynamic Optimisation Problem
DynPopDE	Dynamic Population Differential Evolution
EA	Evolutionary Algorithm
EO	Extremal Optimisation
EP	Evolutionary Programming
ES	Evolution Strategies
ESCA	Evolutionary Swarm Cooperative Algorithm
GA	Genetic Algorithm
GDBG	Generalised Dynamic Benchmark Generator
MGA	Multinational Genetic Algorithm
MMEO	Multi-Phase Multi-Individual Extremal Optimisation
MPB	Moving Peaks Benchmark
PSO	Particle Swarm Optimisation
RMC	Reinitialisation Midpoint Check
SACDE	Self-Adaptive Competing Differential Evolution

SADynPopDE	Self-Adaptive Dynamic Population Differential Evolution
SOS	Self-Organizing Scouts
SBGA	Shifting Balance Genetic Algorithm
SDE	Self-adaptive Differential Evolution
SPSO	Speciation-based Particle Swarm Optimisation

Appendix G

Derived Publications

This appendix lists the publications that have resulted from this study.

M.C. du Plessis and A.P. Engelbrecht. Improved differential evolution for dynamic optimization problems. IEEE Congress on Evolutionary Computation, pages 229-234. IEEE, 2008.

M.C. du Plessis and A.P. Engelbrecht. Self-adaptive competitive differential evolution for dynamic environments. IEEE Symposium on Differential Evolution. IEEE, 2011.

M.C. du Plessis and A.P. Engelbrecht. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research*, 218(1):7-20. Elsevier, 2012

M.C. du Plessis and A.P. Engelbrecht. Differential evolution for dynamic environments with unknown numbers of optima. *Journal of Global Optimization*. Springer, Available online February 7, 2012

M.C. du Plessis and A.P. Engelbrecht. Self-adaptive differential evolution for dynamic environments with fluctuating numbers of optima. *Metaheuristics for Dynamic Optimization*, E. Alba, A. Nakib, P. Siarry (eds.), Springer, to appear 2012