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.

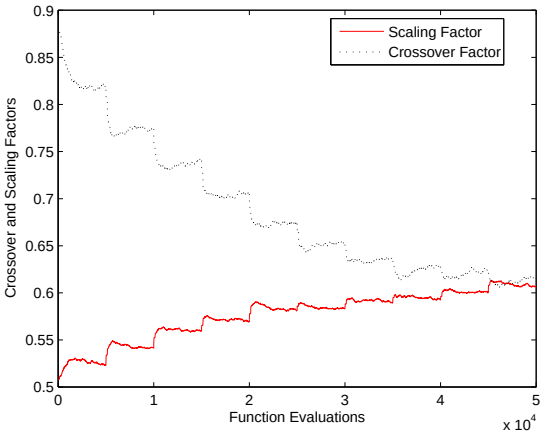

Figure 6.1: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2.

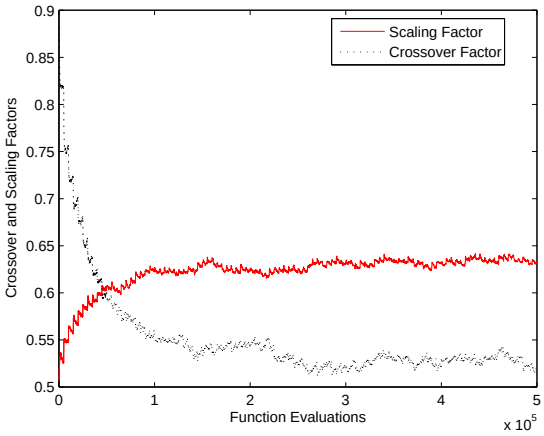

Figure 6.2: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2.

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.

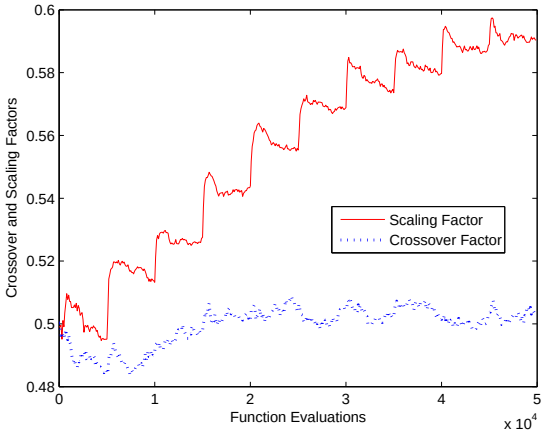

Figure 6.3: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2, when using random initial values.

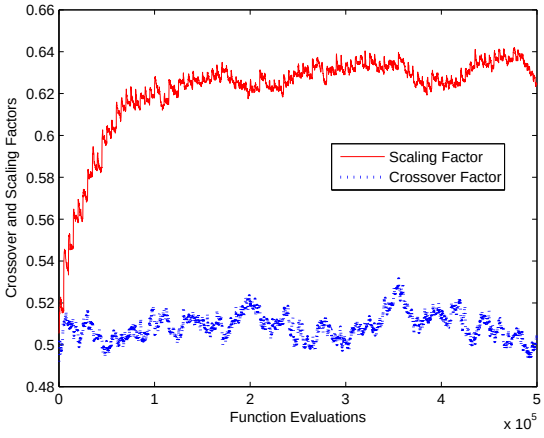

Figure 6.4: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2, when using random initial values.

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.

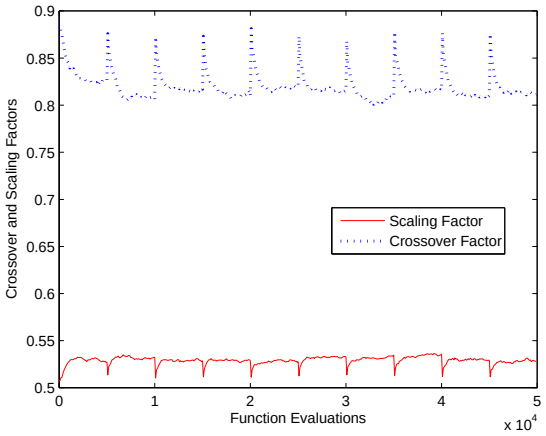

Figure 6.5: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2, when the factors are reset after changes in the environment.

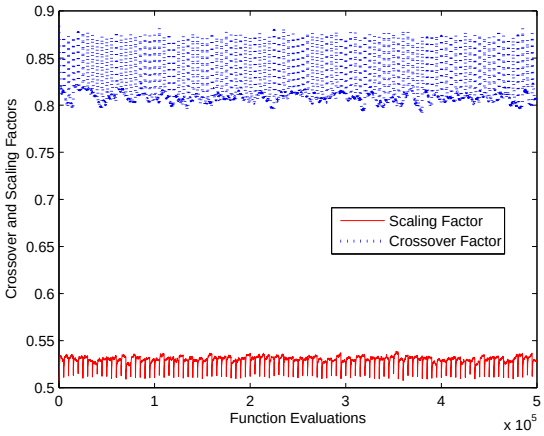

Figure 6.6: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2, when the factors are reset after changes in the environment.

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.

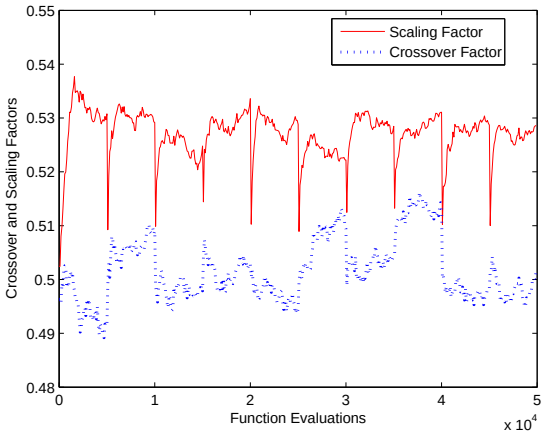

Figure 6.7: Scale and Crossover Factors for 50 000 function evaluations on the MPB, Scenario 2, when the factors are set to random values after changes in the environment.

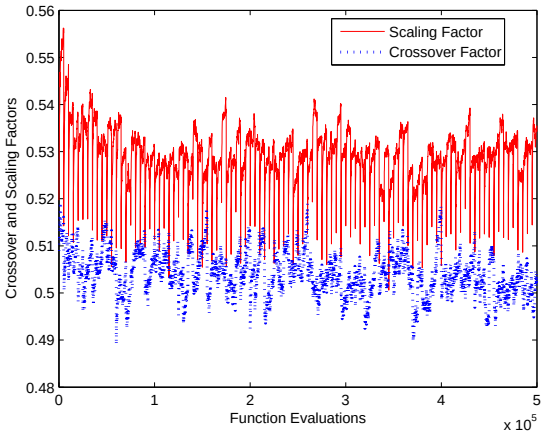

Figure 6.8: Scale and Crossover Factors for 500 000 function evaluations on the MPB, Scenario 2, when the factors are set to random values after changes in the environment.

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:

$$r_{brown}(t) \sim | N(0, r_{dev}(t)) | \tag{6.1}$$

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:

$$r_{pop,k} = \frac{\max\limits_{i_1,i_2 \in \{1,\ldots,n_{I,k}\}} \left\{ \|\vec{x}_{i_1} - \vec{x}_{i_2}\|_2 \right\}}{2} \tag{6.2}$$

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:

$$r_{brown}(t) \sim | C(0, r_{dev}(t)) | \tag{6.3}$$

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

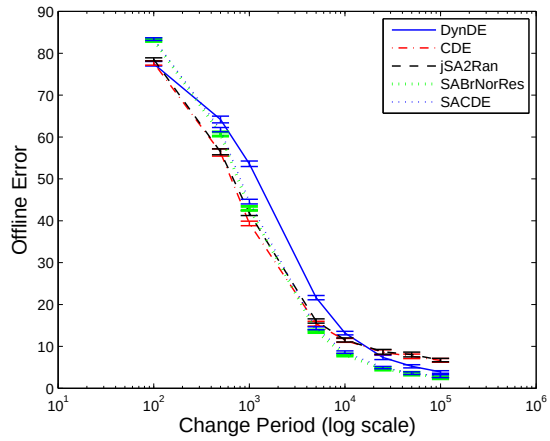

Figure 6.9: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_{1b} and change type T_4 for various settings of change period in 5 dimensions.

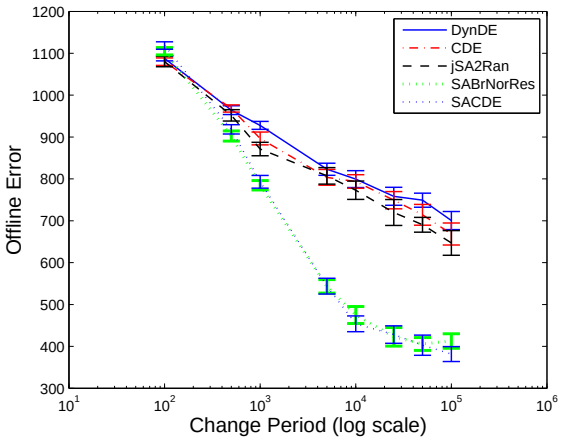

Figure 6.10: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_3 and change type T_4 for various settings of change period in 5 dimensions.

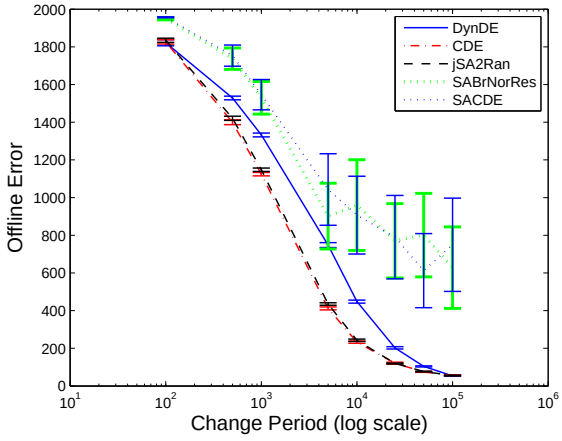

Figure 6.11: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_5 and change type T_4 for various settings of change period in 10 dimensions.

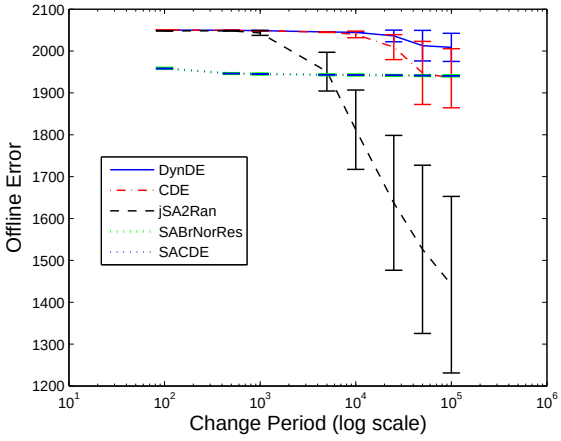

Figure 6.12: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_5 and change type T_4 for various settings of change period in 100 dimensions.

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

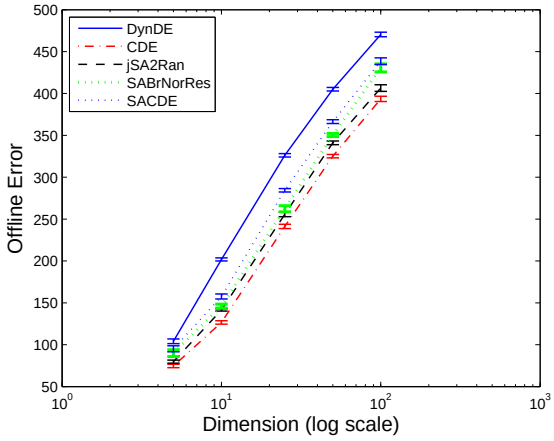

Figure 6.13: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_4 and change type T_1 for various settings of dimension with a change period of 5 000 function evaluations.

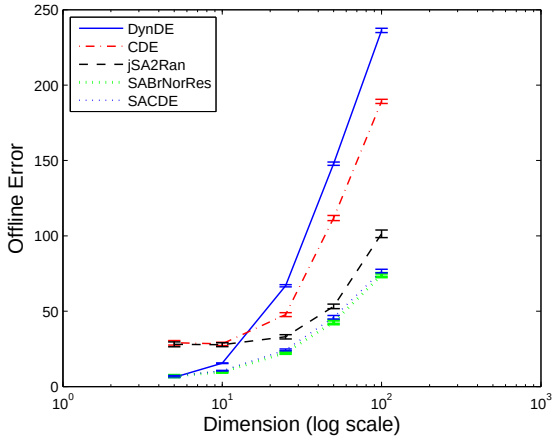

Figure 6.14: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_4 and change type T_1 for various settings of dimension with a change period of 100 000 function evaluations.

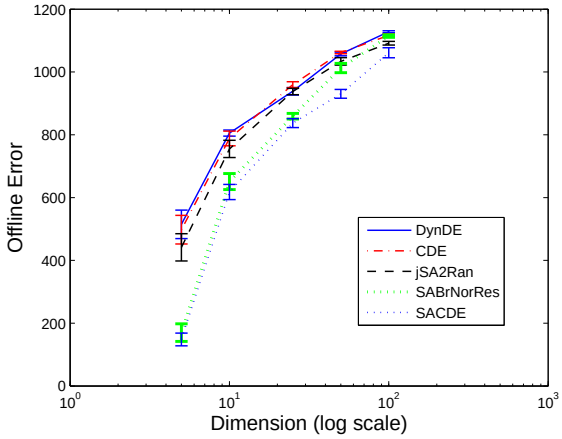

Figure 6.15: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_3 and change type T_1 for various settings of dimension with a change period of 25 000 function evaluations.

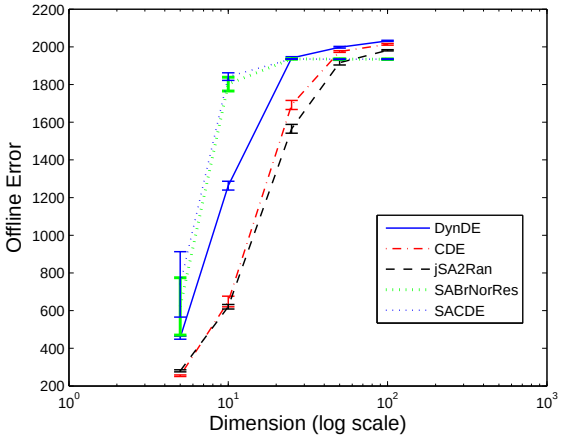

Figure 6.16: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using peak function F_5 and change type T_3 for various settings of dimension with a change period of 100 000 function evaluations.

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

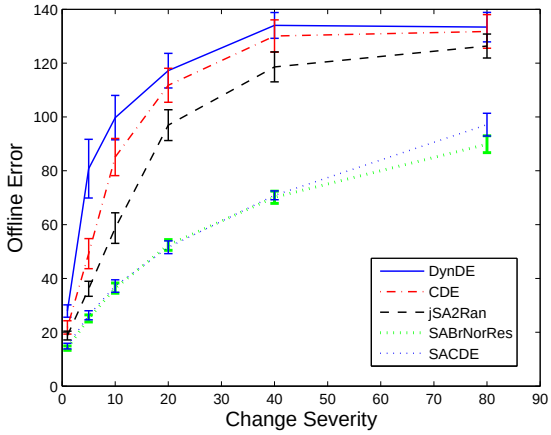

Figure 6.17: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the conical peak function in 10 dimensions for various settings of change severity with a change period of 500 function evaluations.

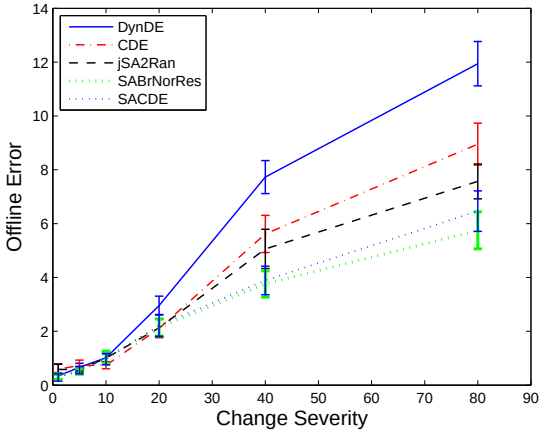

Figure 6.18: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the conical peak function in 10 dimensions for various settings of change severity with a change period of 100 000 function evaluations.

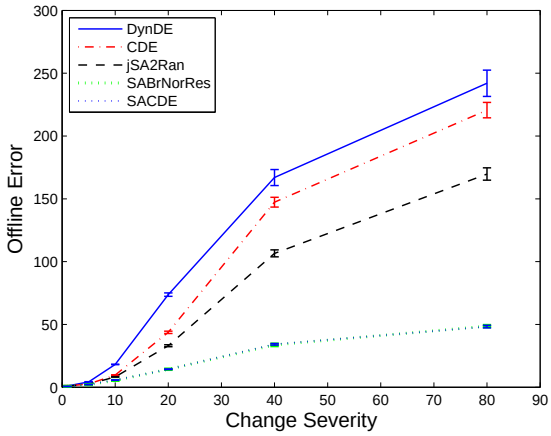

Figure 6.19: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the spherical peak function in 5 dimensions for various settings of change severity with a change period of 5 000 function evaluations.

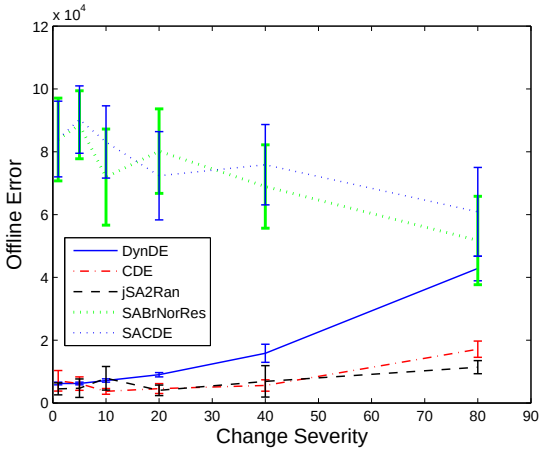

Figure 6.20: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the spherical peak function in 100 dimensions for various settings of change severity with a change period of 5 000 function evaluations.

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.

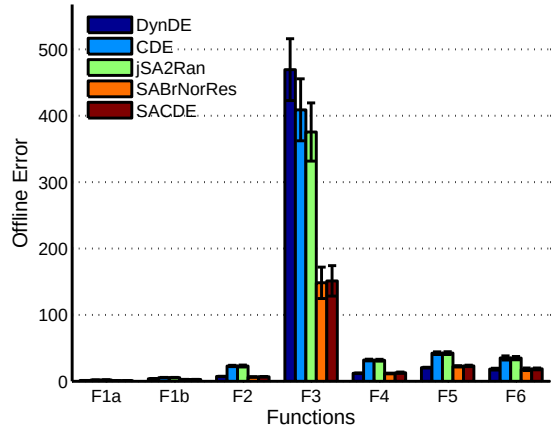

Figure 6.21: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 5 dimensions with a change period of 50 000 function evaluations.

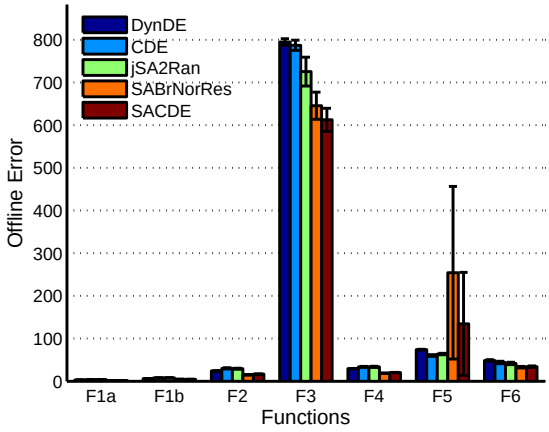

Figure 6.22: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 10 dimensions with a change period of 50 000 function evaluations.

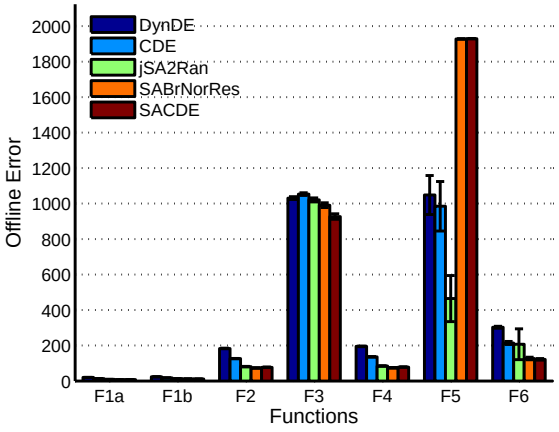

Figure 6.23: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 50 dimensions with a change period of 50 000 function evaluations.

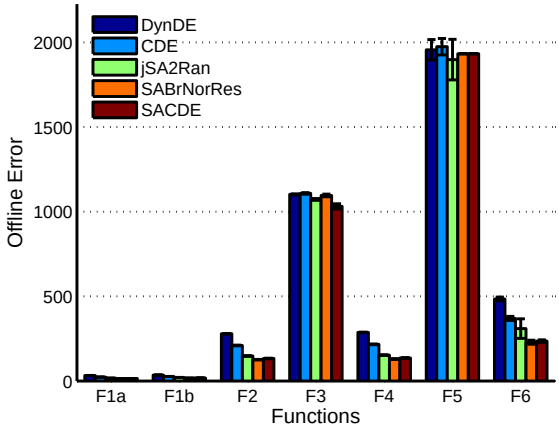

Figure 6.24: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using change type T_1 for various functions in 100 dimensions with a change period of 50 000 function evaluations.

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

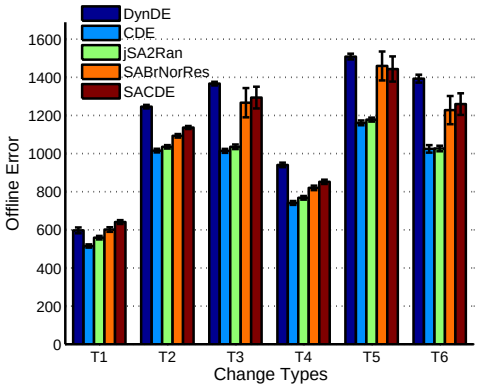

Figure 6.25: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_5 for various change types in 5 dimensions with a change period of 1 000 function evaluations.

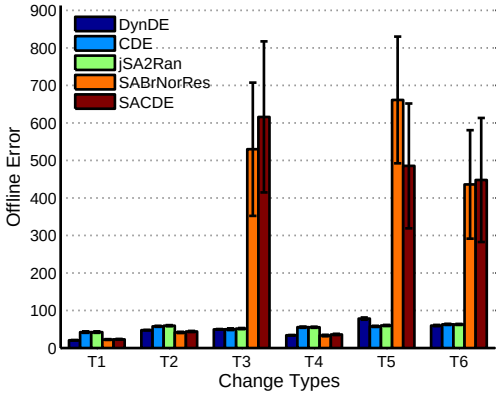

Figure 6.26: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_5 for various change types in 5 dimensions with a change period of 50 000 function evaluations.

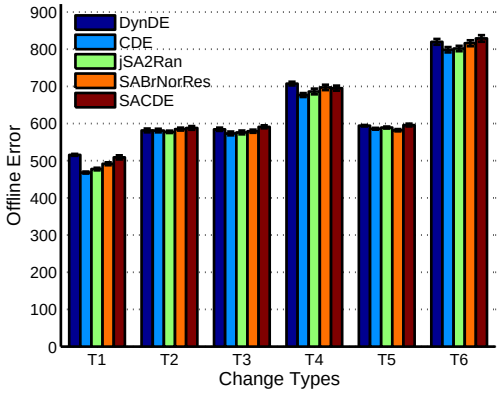

Figure 6.27: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_2 for various change types in 50 dimensions with a change period of 1 000 function evaluations.

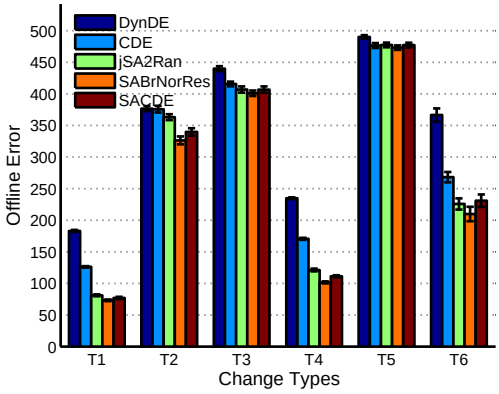

Figure 6.28: Offline errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the GDBG using function F_2 for various change types in 50 dimensions with a change period of 50 000 function evaluations.

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

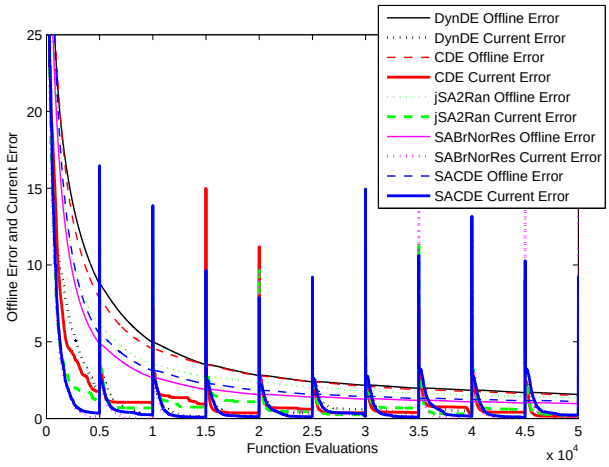

Figure 6.29: Offline and current errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB with the Scenario 2 settings.

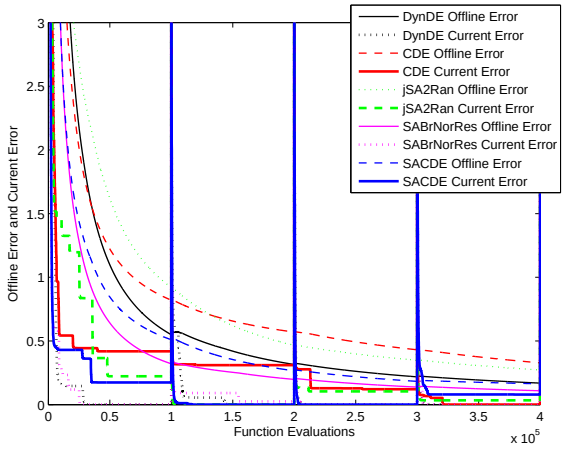

Figure 6.30: Offline and current errors of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB using the Scenario 2 settings with a change period of 100 000.

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

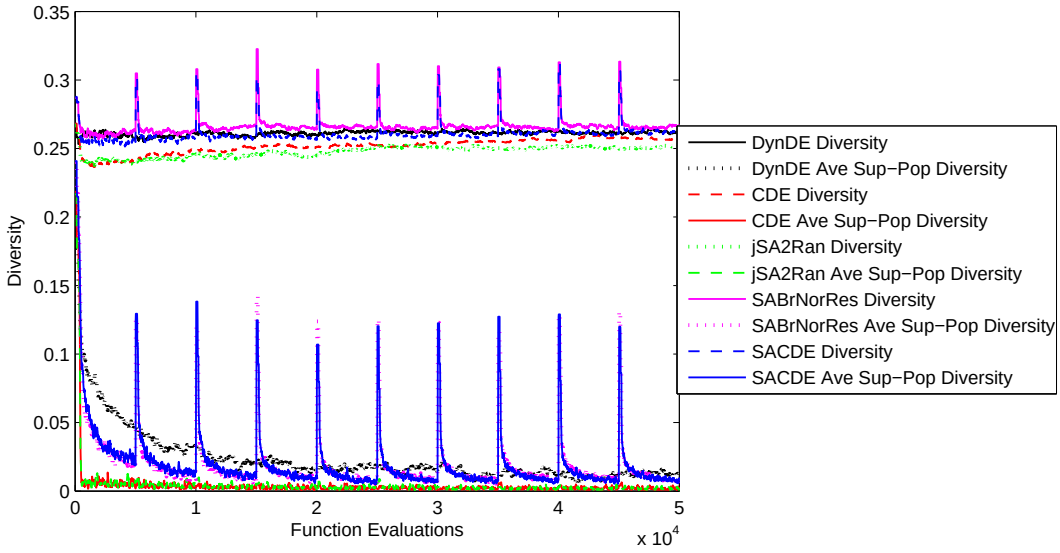

Figure 6.31: Diversity profiles of DynDE, CDE, jSA2Ran, SABrNorRes and SACDE on the MPB with the Scenario 2 settings.

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.

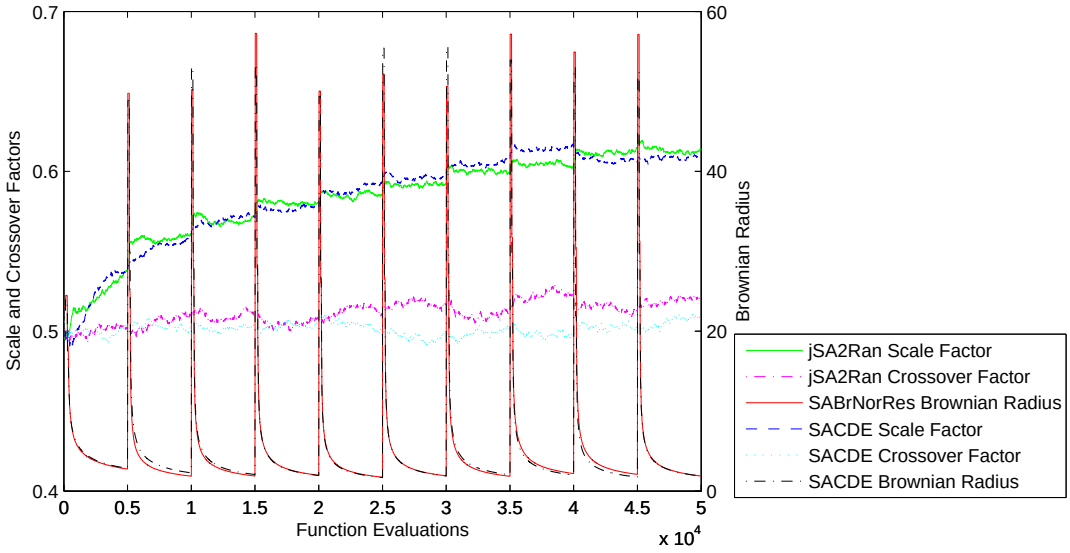

Figure 6.32: Self-adaptive value profiles of jSA2Ran, SABrNorRes and SACDE on the MPB with the Scenario 2 settings.

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	↑2 ↓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.