Chapter 5

UNKNOWN NUMBER OF OPTIMA

The previous chapter presented CDE, an extension of DynDE, consisting of two algorithms: CPE and RMC. This chapter proposes a novel extension to CDE, called DynPopDE. DynPopDE dynamically spawns and removes sub-populations to adapt the number of sub-populations to an appropriate value. DynPopDE is aimed at dynamic optimisation problems where the number of optima is unknown or fluctuating. DynPopDE is evaluated and compared to DynDE and CDE on problems where the number of optima is constant but unknown, and on problems where the number of optima fluctuates over time.

5.1 Introduction

Section 2.5.2 identified the number of optima as a factor that contributes to the hardness of an optimisation problem. The presence of local optima makes it possible that the optimisation algorithm may become trapped, which could lead to sub-optimal results. The locations of local optima are generally not of interest when optimising a static environment, and algorithms typically avoid wasting function evaluations on local optima.

The literature study in Chapter 3 found that several optimisation algorithms for dynamic environments aim to track local optima, as these may become global optima after a change in the environment [Blackwell and Branke, 2006], [Mendes and Mohais, 2005].

The use of multiple populations has become a common strategy for tracking multiple optima. However, the number of optima in a fitness landscape is typically not known, which means the parameter controlling the number of sub-populations must be manually tuned for specific problems. The problem of finding the appropriate number of sub-populations was ignored in the previous chapter, and is consequently the focus of this chapter.

This chapter investigates two types of environments. The first is when the number of optima in the environment is unknown, but constant. The second is when the number of optima is unknown and changes over time. The majority of current research on solving DOPs focuses on problems where the number of optima is unknown but does not change over time. This is the case with the popular moving peaks benchmark (MPB) and the generalised dynamic benchmark generator (GDBG). Environments in which the number of optima explicitly changes over time have only recently received attention by the optimisation community [du Plessis and Engelbrecht, 2012a].

The phrase "varying number of optima" is often used by researchers, but this generally refers only to the fact that some optima are obscured by others for a brief period during the algorithm's execution, and not to the number of optima fluctuating during the optimisation process. An environment in which the number of optima fluctuates over time has been included in the benchmark set of the IEEE WCCI-2012 competition on evolutionary computation for dynamic optimisation problems [Li *et al.*, 2011]. The extension applied to the MPB to allow fluctuating numbers of peaks, which was discussed in Section 2.5.3.1, is used in this chapter to simulate dynamic environments in which the number of optima fluctuates.

This chapter presents a new extension to CDE, called DynPopDE, which is discussed in Section 5.2. This algorithm maintains a dynamic set of sub-populations that can shrink or expand to suit the environment. DynPopDE spawns sub-populations when it finds that limited improvement in fitness is being made by the current sub-populations (Section 5.2.1). A sub-population is removed when it converges to an optimum that is already occupied by another sub-population (Section 5.2.2). DynPopDE thus avoids the problem of tuning the number of sub-populations, by adapting the parameter during the optimisation process. It is interesting to note that several earlier algorithms aimed at dynamic optimisation supported the adaptation of the number of sub-populations (e.g. SOS [Branke *et al.*,

2000], [Branke, 2002] and MGA [Ursem, 2000]). Later, more effective algorithms made use of a constant number of sub-populations (e.g. MPSO [Blackwell and Branke, 2006], DynDE [Mendes and Mohais, 2005] and *jDE* [Brest *et al.*, 2009]). DynPopDE revisits the domain of using a dynamic number of sub-populations by building on the competitive population evaluation process of CDE.

The remainder of the this chapter is structured as follows: Section 5.2 presents the DynPopDE algorithm. The research questions of this chapter are outlined and experimentally investigated in Section 5.3. These investigations include a scalability study of DynDE, CDE and DynPopDE on environments with various numbers of optima and environments where the number of optima fluctuates over time. DynPopDE is compared to other algorithms found in the literature in Section 5.4 and conclusions are drawn and presented in Section 5.5.

5.2 Dynamic Population Differential Evolution

This section introduces a novel approach specifically designed to address problems with unknown or fluctuating numbers of optima. Dynamic population differential evolution (DynPopDE) is an extension of CDE consisting of two components, namely spawning new populations (refer to Section 5.2.1) and removing populations (refer to Section 5.2.2). The complete DynPopDE algorithm is given in Section 5.2.3.

5.2.1 Spawning Populations

Section 5.3 will show that, even if the number of peaks is known, using a number of sub-populations equal to the number of peaks is not always an effective strategy. When the number of peaks is unknown, choosing the number of sub-populations to use would be, at best, an educated guess. It is therefore suggested that the number of sub-populations should not be a parameter of the algorithm, but that sub-populations should be spawned as needed. The question that must be answered is: When should new populations be spawned?

CDE is based on allocating processing time and function evaluations to populations based on a performance value, $\mathcal{P}_k(t)$ (refer to equation (4.2) in Section 4.3). Sub-populations

are evolved in sequence until all sub-populations converge to their respective optima in the fitness landscape. It is proposed here that an appropriate time to introduce an additional sub-population is when little further improvement in fitness is found for all the current sub-populations. Introducing new sub-populations earlier would be contrary to the competing population approach of CDE, where inferior sub-populations are deliberately excluded from the evolution process so that optima can be discovered earlier.

A detection scheme is suggested to indicate when evolution has reached a point of little or no improvement in fitness of current sub-populations. This point is referred to as stagnation in the context of this thesis. DynPopDE introduces a new population of randomly created individuals when stagnation is detected, so that previously undiscovered optima can be located. CDE calculates the performance value of a sub-population, P_k, as the product of its current relative fitness, $R_k(t)$, and the improvement that was made in fitness during the previous generation, $\Delta F(\vec{x}_{best,k}, t)$ (see equation (4.2)). The suggestion made in this section is that a meaningful indicator of stagnation is when all the current sub-populations receive a zero improvement of fitness after their last respective function evaluations. Let $n_k(t)$ be the number of current sub-populations. Define a function, $S(t)$, that is true if stagnation has occurred, as follows:

$$S(t) = \begin{cases} true \text{ if}(\Delta F(\vec{x}_{best,k}, t) = 0) \ \forall \ k \in \{1, \ldots, n_k(t)\} \\ false \text{ otherwise} \end{cases} \tag{5.1}$$

$\Delta F(\vec{x}_{best,k}, t)$ is as defined in equation (4.3). Note that equation (5.1) does not guarantee that stagnation of all populations has permanently occurred, since the fitness of some of the sub-populations may improve in subsequent generations. However, when $S(t)$ is true, it does mean that none of the sub-populations has improved its fitness in the previous generation. Since function evaluations are not effectively used by the current sub-populations, a new sub-population should be created which may lead to locating more optima in the problem space.

After each generation, DynPopDE determines the value of $S(t)$. If $S(t) = true$ then a new randomly generated sub-population is added to the set of sub-populations. The sub-population spawning approach allows DynPopDE to commence with only a single sub-population and adapt to an appropriate number of sub-populations. The number of

sub-populations is thus removed as a parameter from the algorithm.

5.2.2 Removing Populations

The previous section explained how new populations are spawned when necessary. However, it is possible that equation (5.1) detects stagnation incorrectly since it cannot guarantee that stagnation for all sub-populations has occurred indefinitely. More sub-populations than necessary may consequently be created. Furthermore, in problems where the number of optima fluctuate, it would be desirable to remove superfluous sub-populations when the number of optima decreases. The algorithms should thus be able to detect and remove redundant sub-populations.

DynDE and CDE reinitialises a sub-population through exclusion when the spatial difference between the sub-population and a more fit sub-population drops below the exclusion threshold. A reasonable assumption that can be made is that, when redundant populations are present (i.e. more sub-populations exist than optima), these sub-populations will perpetually be reinitialised and will not converge to specific optima, since exclusion prevents the convergence of more than one sub-population to the same optimum. Consequently, redundant sub-populations can be detected by finding sub-populations that are successively reinitialised through exclusion without reaching a point of apparent stagnation (i.e. there are no more optima available for the sub-population to occupy).

A sub-population, P_k, will be discarded when it is flagged for reinitialisation due to exclusion and if

$$\Delta F(\vec{x}_{best,k}, t) \neq 0 \tag{5.2}$$

The exclusion process, as given in Algorithm 13, is thus further adapted to remove sub-populations as detailed in Algorithm 14 (assuming a function minimisation problem).

This approach may, at times, incorrectly classify populations as redundant in that it does not guarantee the removal of populations only when the number of populations outnumber the number of optima. A sub-population may be discarded for converging to an optimum occupied by another sub-population even when undiscovered optima still exist in the problem space. However, no optimum is ever left unguarded through the discarding process since a sub-population is discarded only when it converges to an optimum already

Algorithm 14: DynPopDE Exclusion

```
for k_1 = 1, ..., n_k(t) do
    for k_2 = 1, ..., n_k(t) do
        if ||x⃗_{best,k_1} − x⃗_{best,k_2}||_2 < r_excl and k_1 ≠ k_2 then
            Let x⃗_mid = (x⃗_{best,k_1} + x⃗_{best,k_2})/2;
            if F(x⃗_mid) > F(x⃗_{best,k_1}) and F(x⃗_mid) > F(x⃗_{best,k_2}) then
                if F(x⃗_{best,k_1}) < F(x⃗_{best,k_2}) then
                    if ΔF(x⃗_{best,k_2}, t) = 0 then
                        Reinitialise population P_{k_2}
                    else
                        Discard population P_{k_2}
                    end
                else
                    if ΔF(x⃗_{best,k_1}, t) = 0 then
                        Reinitialise population P_{k_1}
                    else
                        Discard population P_{k_1}
                    end
                end
            end
        end
    end
end
```

occupied by another sub-population. No information about the fitness landscape is thus lost through discarding a sub-population. If all sub-populations have stagnated, a new sub-population will be created through the spawning process. The spawning and discarding components of DynPopDE thus reach a point of equilibrium, where sub-populations are created when function evaluations are not being used effectively by the current sub-populations, and where sub-populations are removed when converging to optima that are already guarded by other sub-populations.

5.2.3 DynPopDE Algorithm

The previous sections described the two components of DynPopDE. These components are incorporated into CDE to form the complete DynPopDE algorithm given in Algorithm 15.

Algorithm 15: DynPopDE Algorithm

while *termination criterion not met* **do**
  Allow the standard CDE algorithm to run for two generations;
  repeat
    for $k = 1, \ldots, n_k$ **do**
      Calculate the performance value, $\mathcal{P}_k(t)$
    end
    Select sub-population P_a such that $\mathcal{P}_a(t) = \min_{k=1,\ldots,n_k} \{\mathcal{P}_k(t)\}$;
    Evolve only sub-population P_a;
    $t = t + 1$;
    Perform Exclusion according to Algorithm 14;
    if $S(t) = true$ **then**
      Introduce new randomly created sub-population;
    end
  until *a change occurs in the environment*;
end

5.3 Experimental Results

This section details experiments conducted on DynDE, CDE and DynPopDE. Two main problem classes are investigated. The first is dynamic environments in which the number of optima is constant but unknown. The second class is dynamic optimisation problems in which the number of optima is fluctuating over time and unknown. Experiments were conducted to answer the following research questions:

1. *How many sub-populations should be used when the number of optima is known?* Before investigating problems with unknown numbers of optima, experiments are conducted on DynDE and CDE to determine which is the best strategy to follow when the number of optima is known. Two strategies are investigated, using the same number of sub-populations as the number of optima, and using a constant number of optima.

2. *How do DynDE, CDE and DynPopDE scale with respect to unknown numbers of optima, number of dimensions, and the change period?* A thorough scalability study was conducted on DynDE and CDE in the previous chapter with respect to the number of dimensions and change period. Experiments are conducted to determine if and how the number of optima affects the scalability behaviour of DynDE, CDE and DynPopDE.

3. *Does DynPopDE perform better than CDE and DynDE on problems where the number of optima is constant, but unknown?* The results of DynDE, CDE and DynPopDE are compared to determine whether DynPopDE is a more effective algorithm.

4. *How do DynDE, CDE and DynPopDE scale in terms of the maximum number of optima and percentage change in the environment on dynamic optimisation problems where the number of optima is unknown and fluctuates over time?* The scalability of the algorithms with regard to the maximum number of optima and percentage change in the number of optima in the problem space, is investigated. The effect of changing the number of dimensions and change period is also investigated.

5. *Is DynPopDE more effective than DynDE and CDE on dynamic problems where the*

number of optima is unknown and fluctuates over time? The results of DynDE, CDE and DynPopDE are compared to determine whether DynPopDE is a more effective algorithm.

6. *What is the convergence profile of DynPopDE?* The DynPopDE algorithm commences with a single sub-population. Fewer individuals are expected to affect diversity negatively, as a smaller sample of the fitness landscape is used by the optimisation algorithm. The convergence behaviours of DynDE, CDE and DynPopDE are compared in terms of diversity, current error, and resulting offline error to assist in explaining the trends observed in the analyses of research questions 2, 3, 4 and 5. The numbers of sub-populations employed by DynPopDE for unknown and fluctuating numbers of optima are investigated.

7. *Is the process of spawning and removing sub-populations as used in DynPopDE more effective than the process used in MPSO2?* This research question investigates whether other spawning and removal processes, namely the approaches used in MPSO2 [Blackwell, 2007], are a better choice for incorporation into CDE, and subsequently yield better results.

8. *Is DynPopDE more effective than CDE and DynDE on the set of environments used in Chapter 4?* DynPopDE is tested on the set of dynamic environments that were used in the previous chapter to compare DynDE to CDE. This analysis is included because several of the environments in this set contain large numbers of optima, and consist of functions which were not used for evaluation in research questions 2, 3, 4 and 5.

Questions 1 to 3 are investigated on benchmark problems where the number of optima is kept constant during the optimisation process. Questions 4 and 5 are investigated on problems where the number of optima fluctuates over time. Questions 6 and 7 are investigated on both sets of problems. The next section outlines the experimental procedure that was followed to answer the research questions listed above. Sections 5.3.2 to 5.3.9 respectively cover research questions 1 to 8.

5.3.1 Experimental Procedure

Chapter 2 concluded that the extended MPB (refer to Section 2.5.3) is ideal for studying the effect of number of optima in the environment on optimisation algorithms. This chapter investigates the performance of DynDE, CDE and DynPopDE on two types of dynamic environments. Firstly, environments with various numbers of optima are investigated. The experimental procedure for these investigations is discussed in Section 5.3.1.1. Secondly, environments where the numbers of optima fluctuate over time are investigated. The experimental procedure for fluctuating number of optima experiments is given in Section 5.3.1.2.

A stopping criterion of 60 changes in the environment was used for all experiments. The offline error was used as the performance measure for all environments. Experiments were repeated 30 times to facilitate drawing statistically valid conclusions from the results. Mann-Whitney U tests were used to test statistical significance when comparing algorithms.

5.3.1.1 Constant Numbers of Optima Experimental Procedure

Variations of the Scenario 2 settings of the MPB were used to simulate environments with various numbers of peaks. The set of variations is defined here as the n_p *standard set*. The n_p *standard set* consists of all combinations of settings given in Table 5.1. The number of dimensions, number of peaks, change period, and underlying function are varied. Five settings for number of dimensions, six settings for number of peaks, and eight settings for change period were investigated on both of the peak functions. The n_p *standard set* thus consists of a total of 480 environments.

5.3.1.2 Fluctuating Numbers of Optima Experimental Procedure

Various settings of the extended MPB were used to simulate environments where the number of peaks fluctuates over time. The set of variations is defined here as the $n_p(t)$ *standard set*. The $n_p(t)$ *standard set* consists of all combinations of settings given in Table 5.2. The number of dimensions, maximum number of peaks, percentage change in the number of peaks, change period, and underlying function are varied. Five settings

Table 5.1: The n_p standard set

Setting	**Value**
Number of dimensions (n_d)	5, 10, 25, 50, 100
Number of Peaks (n_p)	5, 10, 25, 50, 100, 200
Max and Min Peak height	[30,70]
Max and Min Peak width	[1.0,12.0]
Change period (Cp)	100, 500, 1000, 5000, 10000, 25000, 50000, 100000
Change severity (C_s)	1.0
Height severity	7.0
Width severity	1.0
Function (F)	Cone, Sphere
Correlation	0.0

for number of dimensions, five settings for maximum number of peaks, five settings for percentage change in the number of peaks, and eight settings for change period were investigated on both of the peak functions. A total of 2 000 environments are thus included in the $n_p(t)$ *standard set*.

Table 5.2: The $n_p(t)$ standard set

Setting	**Value**
Number of dimensions (n_d)	5, 10, 25, 50, 100
Maximum Number of Peaks (M_{n_p})	10, 25, 50, 100, 200
Percentage Change in n_p (pc)	5, 10, 20, 40, 80
Max and Min Peak height	[30,70]
Max and Min Peak width	[1.0,12.0]
Change period (Cp)	100, 500, 1000, 5000, 10000, 25000, 50000, 100000
Change severity (C_s)	1.0
Height severity	7.0
Width severity	1.0
Function (F)	Cone, Sphere
Correlation	0.0

5.3.2 Research Question 1

How many sub-populations should be used when the number of optima is known?

The motivation for employing multiple populations is that multiple (preferably all) optima should be tracked. This enables the algorithm to locate the position of a new global optimum quickly when a change in the environment results in a different optimum becoming the global optimum. However, algorithms generally do not have information regarding the number of optima that are present in a fitness landscape. This, potentially, makes the number of sub-populations a very important parameter of an optimisation algorithm aimed at DOPs.

The number of optima are assumed to be known in advance for the purpose of this research question. The number of sub-populations used by DynDE and CDE can thus be set equal to the number of optima. These two algorithms were compared to DynDE and CDE where each used 10 sub-populations (the setting used in Chapter 4). DynDE and CDE are referred to as DynDE10 and CDE10 when using 10 sub-populations. This baseline comparison can show how beneficial it is to know the number of optima in advance.

DynDE, CDE, DynDE10 and CDE10 were compared using the n_p standard set which was defined in Section 5.3.1.1. CDE performed statistically significantly better than DynDE in 231 of the 480 experiments and worse in only 47 cases (a full analysis is available in Appendix C). This is consistent with the conclusion of Chapter 4 that CDE is a superior algorithm for solving DOPs than DynDE.

A more interesting result was found when comparing DynDE and CDE to their respective counterparts that always used 10 sub-populations, i.e. DynDE10 and CDE10. DynDE10 performed better than DynDE in 253 of the 480 experiments and worse in only 131 cases. CDE10 performed better than CDE in 236 cases and performed worse in only 126 cases. The algorithms that used 10 sub-populations thus performed better more often than the algorithms which used a number of sub-populations equal to the number of peaks that were present in the environment. This result is counter-intuitive since, for most of the environments that were investigated, DynDE10 and CDE10 had fewer sub-populations than the number of peaks, and could consequently not track all optima.

The explanation for the fact that DynDE10 and CDE10 performed better, overall,

than DynDE and CDE can be found by noting that an increase in the number of sub-populations results in a decreased number of generations that can be performed between the changes that occur in the environments after a set number of function evaluations. For example, DynDE with 200 sub-populations would result in about 1 500 function evaluations per generation. A dynamic environment that has a change period of 5 000 function evaluations would thus allow the algorithm to perform less than four DynDE generations between changes in the environment. Such a small number of generations would naturally result in poor performance by DynDE. The problem is less applicable to CDE after the second generation, when only the best-performing sub-population is evolved at any given time. However, a significant number of function evaluations would still be wasted during the initial performance calculation phase.

A comparative performance analysis between CDE and CDE10 was conducted to support the above explanation. Table 5.3 gives a count of the number of times CDE10 performed significantly better (indicated by $\uparrow$) than CDE for each of the settings of number of peaks. The number of times that CDE performed better than CDE10 is indicated by the $\downarrow$ symbol. Data was stratified into categories for different values of number of dimensions and change period. Cases where CDE10 performed better more often than CDE are shaded and printed in boldface, while cases where CDE performed better more often than CDE10 are printed in italics. A similar analysis was performed for DynDE10 versus DynDE, and is included in Appendix C, since results similar to the CDE10 versus CDE comparison were found.

Table 5.3 shows that the cases where CDE performed better than CDE10 were isolated to high change period experiments. This is consistent with the explanation that large numbers of sub-populations result in too few generations that are performed between changes in the environment. More generations can be performed as the period between changes is increased, which resulted in CDE eventually outperforming CDE10 as CDE can track more optima. Figures 5.1 and 5.2 give the offline errors of DynDE, CDE, DynDE10 and CDE10 on the conical peak function in five dimensions for various settings of number of peaks, with a change period of 5 000 and 100 000, respectively. Figure 5.1 shows that, at a low change period, CDE10 performed better than CDE when a large number of peaks was present in the environment. At high change periods, of which Figure 5.2 is an example,

enough function evaluations were available for CDE to locate and track optima effectively.

Table 5.3 shows that, as the number of dimensions was increased, CDE10 started to outperform CDE at increasingly higher change periods. As the number of dimensions was increased, the number of generations, required by an algorithm to locate optima, also increased (refer to the discussion on the influence of number of dimensions on the scalability of DynDE and CDE in Section 4.6.4.3). The algorithms using large sub-populations did not perform an adequate number of generations to locate optima in high dimensional environments. Figures 5.3 and 5.4 give the offline errors of DynDE, CDE, DynDE10 and CDE10 in 25 dimensions, with a change period of 5 000 and 100 000, while Figures 5.5 and 5.6 give the same information in 100 dimensions. A comparison of Figures 5.3 and 5.4 with Figures 5.1 and 5.2 shows a considerable comparative improvement of CDE10 with respect to CDE, with CDE10 performing significantly better than CDE in all cases with a high number of peaks in the low change period experiments. CDE still performed better than CDE10 in the high change period experiments. This trend changed in the 100 dimensional experiments where CDE10 always yielded results similar to or better than those of CDE. These results confirm that using a large number of sub-populations is not appropriate in high dimensions, even when a large number of function evaluations are available. It is better to use a smaller number of sub-populations and consequently perform more generations between changes in the environment.

This research question investigated whether using a number of sub-populations equal to the number of optima in the environment is effective. The experimental results showed that the cases where the algorithms used the same number of sub-populations as number of optima only yielded superior results in experiments where the number of optima was small, or when the change period was large in low dimensional environments. Ten sub-populations proved to be a more beneficial setting in the majority of cases that were investigated. This section thus showed that, even when the number of optima in the environment is known, this information is generally not beneficial.

The following sections assume no knowledge regarding the number of optima. DynDE and CDE will use 10 sub-populations at all times, as this value has been shown to produce reasonable results over a wide range of environments. Note that, for the rest of this thesis, the “10” suffix will be omitted from DynDE10 and CDE10.

Table 5.3: CDE10 vs CDE performance analysis

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓15
10	(2)	↑0 ↓0	↑0 ↓0	**↑1 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓1	↑1 ↓1
25	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑6 ↓10
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑5 ↓10
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑6 ↓9
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑8 ↓8
C	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑5 ↓1**	↑1 ↓3	↑0 ↓5	↑0 ↓5	↑0 ↓4	↑0 ↓5	↑14 ↓25
S	(6)	**↑3 ↓1**	**↑4 ↓1**	**↑4 ↓1**	↑1 ↓4	↑0 ↓5	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑12 ↓28
All	(12)	**↑7 ↓2**	**↑8 ↓2**	**↑9 ↓2**	↑2 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓9	↑0 ↓11	↑26 ↓53
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓11
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	**↑7 ↓6**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	↑0 ↓2	**↑8 ↓6**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑0 ↓1	↑0 ↓2	**↑11 ↓4**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑0 ↓2	**↑12 ↓4**
C	(6)	**↑2 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓0**	↑2 ↓2	↑0 ↓4	↑0 ↓4	↑0 ↓4	↑15 ↓17
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑2 ↓1**	↑1 ↓3	↑0 ↓5	**↑23 ↓14**
All	(12)	**↑6 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑7 ↓1**	**↑6 ↓3**	↑2 ↓5	↑1 ↓7	↑0 ↓9	**↑38 ↓31**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓11
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑12 ↓3**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	**↑13 ↓2**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑14 ↓2**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑15 ↓1**
C	(6)	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓1**	**↑2 ↓1**	↑1 ↓3	↑0 ↓4	**↑22 ↓11**
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑32 ↓8**
All	(12)	**↑7 ↓1**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓1**	**↑8 ↓2**	**↑6 ↓2**	**↑5 ↓4**	↑4 ↓5	**↑54 ↓19**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓2	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓12
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	**↑1 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	↑1 ↓1	**↑13 ↓2**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑14 ↓1**
100	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑3 ↓0**	**↑3 ↓1**	**↑5 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓0**	**↑2 ↓1**	↑1 ↓2	**↑26 ↓7**
S	(6)	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑32 ↓8**
All	(12)	**↑7 ↓1**	**↑7 ↓2**	**↑9 ↓2**	**↑8 ↓2**	**↑8 ↓2**	**↑8 ↓1**	**↑6 ↓2**	**↑5 ↓3**	**↑58 ↓15**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓1	↑0 ↓0	↑0 ↓1	↑0 ↓7
10	(2)	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑13 ↓1**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑15 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑4 ↓1**	**↑3 ↓0**	**↑2 ↓1**	**↑28 ↓6**
S	(6)	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓1**	**↑32 ↓2**
All	(12)	**↑7 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑7 ↓0**	**↑6 ↓2**	**↑60 ↓8**
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	↑0 ↓7	↑0 ↓9	↑0 ↓9	↑0 ↓7	↑0 ↓8	↑0 ↓6	↑0 ↓4	↑0 ↓6	↑0 ↓56
10	(10)	↑0 ↓0	↑0 ↓0	**↑2 ↓0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑0 ↓1	**↑2 ↓1**
25	(10)	**↑6 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑7 ↓2**	**↑7 ↓3**	**↑5 ↓4**	↑3 ↓6	↑3 ↓7	**↑51 ↓22**
50	(10)	**↑8 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓2**	**↑7 ↓3**	**↑5 ↓3**	↑4 ↓5	↑3 ↓6	**↑55 ↓19**
100	(10)	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑8 ↓1**	**↑8 ↓2**	**↑7 ↓3**	**↑5 ↓4**	↑4 ↓5	**↑61 ↓15**
200	(10)	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑8 ↓2**	**↑7 ↓3**	**↑7 ↓3**	↑5 ↓5	**↑67 ↓13**
C	(30)	**↑15 ↓2**	**↑19 ↓5**	**↑22 ↓5**	**↑16 ↓5**	**↑14 ↓10**	↑10 ↓11	↑6 ↓12	↑3 ↓16	**↑105 ↓66**
S	(30)	**↑19 ↓5**	**↑20 ↓4**	**↑20 ↓4**	**↑17 ↓7**	**↑16 ↓8**	**↑14 ↓8**	**↑13 ↓10**	↑12 ↓14	**↑131 ↓60**
All	(60)	**↑34 ↓7**	**↑39 ↓9**	**↑42 ↓9**	**↑33 ↓12**	**↑30 ↓18**	**↑24 ↓19**	↑19 ↓22	↑15 ↓30	**↑236 ↓126**

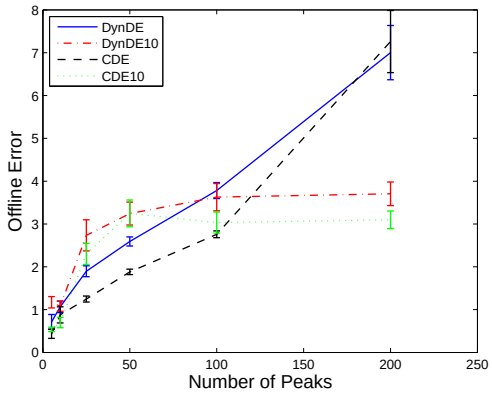

Figure 5.1: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 5 000 for various settings of number of peaks in 5 dimensions.

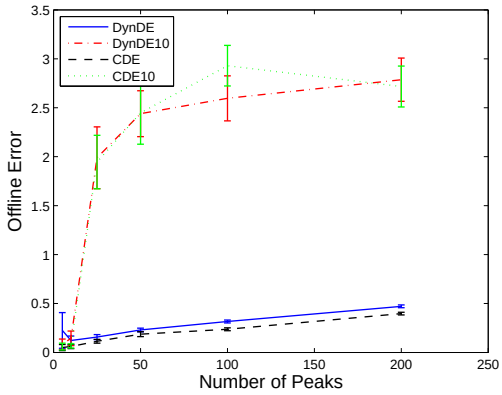

Figure 5.2: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 100 000 for various settings of number of peaks in 5 dimensions.

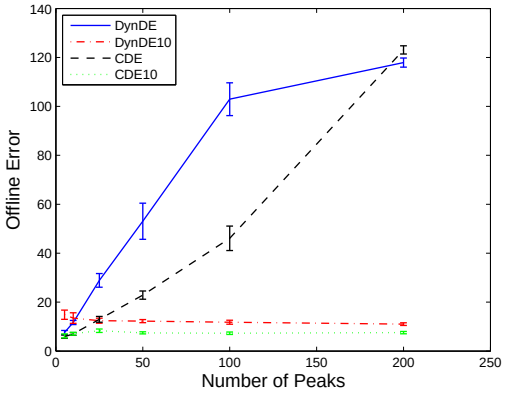

Figure 5.3: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 5 000 for various settings of number of peaks in 25 dimensions.

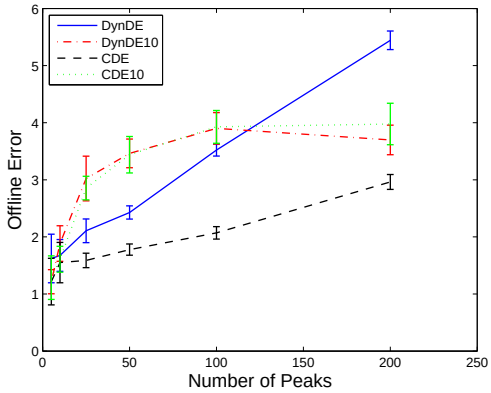

Figure 5.4: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 100 000 for various settings of number of peaks in 25 dimensions.

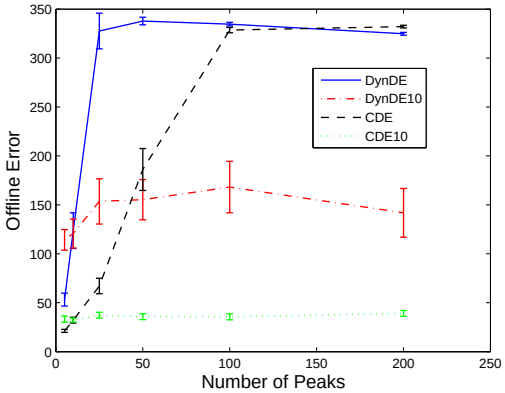

Figure 5.5: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 5 000 for various settings of number of peaks in 100 dimensions.

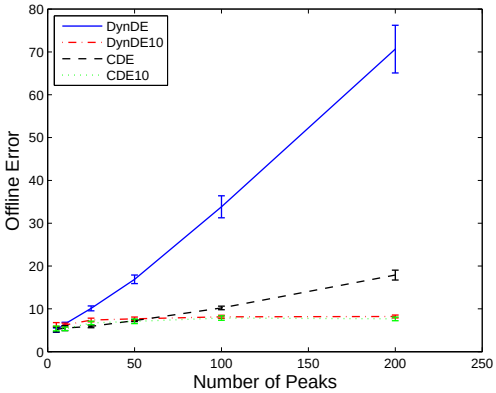

Figure 5.6: Offline errors of DynDE, DynDE10, CDE, and CDE10 on the conical peak function with a change period of 100 000 for various settings of number of peaks in 100 dimensions.

5.3.3 Research Question 2

How do DynDE, CDE and DynPopDE scale with respect to unknown numbers of optima, number of dimensions, and change period?

Chapter 4 investigated the scalability of DynDE and CDE with respect to change severity, underlying function, number of dimensions, change type, and change period. This research question investigates scalability with respect to the number of optima in the environment. DynPopDE, the extension to CDE proposed in this chapter, is included in the scalability study. DynPopDE was also executed on the n_p standard set of experiments, following the experimental procedure described in Section 5.3.1.1.

The first trend that can be observed is that increasing the number of optima tended to result in an increased offline error for the algorithms. Figure 5.7 shows that the offline error of DynDE, CDE and DynPopDE increased as the number of optima was increased, when using the conical peak function in five dimensions with a change period of 5 000. Several exceptions to this general trend were, however, observed. For example, consider Figure 5.12 (the 25 dimensional case of the experiment shown in Figure 5.7), where the offline error of DynDE decreased as the number of peaks was increased; the offline error of CDE and DynPopDE initially increased, but then decreased when the number of peaks exceeded 25.

An increased number of optima in an MPB environment can thus be either detrimental or beneficial to an optimisation algorithm. Large numbers of peaks make it impossible for an algorithm to track all peaks. This may have the result that a global optimum is not discovered, which explains the increasing offline errors visible in Figure 5.7. Conversely, an increased number of peaks raises the average function value of an MPB environment [Moser and Chiong, 2010], and makes it easier for an algorithm to locate local optima. A consequence of this effect is the decreasing offline errors visible in Figure 5.12.

CDE generally performed better than DynDE, while DynPopDE generally yielded lower offline errors than CDE. The magnitude of the difference in offline error was found to be related to the change period and the number of dimensions. The effect of the number of dimensions will be discussed first.

Figures 5.7 to 5.10 show the offline errors of DynDE, CDE and DynPopDE on the

conical peak function with a change period of 5 000 for different settings of the number of peaks in 5, 10, 50 and 100 dimensions, respectively. Similar trends were found when using the spherical peak function. CDE and DynPopDE generally performed considerably better than DynDE. A clear trend with regard to the comparative performance of CDE and DynPopDE over the increasing number of dimensions is visible. DynPopDE outperformed CDE by a wide margin in the five dimensional case (for all cases except when a small number of peaks was used). This margin is reduced in Figure 5.8 where the number of dimensions was increased to 10, but DynPopDE still performed better than CDE in the majority of cases. The difference in performance between CDE and DynPopDE narrowed and eventually disappeared as the number of dimensions was increased to 50 and 100 dimensions. Figures 5.9 and 5.10 show overlapping confidence intervals for the offline errors of CDE and DynPopDE.

DynPopDE thus scaled better than CDE on low dimensional problems, but not on high dimensional problems. Research question six will endeavour to shed some light on this phenomenon.

Recall from Chapter 4 that a very low change period was found to be detrimental to the performance of DynDE and CDE. This is due, in part, to the small number of generations that can be performed by DynDE and CDE at low change periods. DynPopDE has an advantage over DynDE and CDE in that it commences with a single sub-population, and can consequently perform more generations than DynDE and CDE.

Figure 5.11 shows the offline errors of DynDE, CDE and DynPopDE on the conical peak function in 25 dimensions with a change period of 500 function evaluations. The advantage of commencing with a small number of sub-populations is clearly illustrated by the magnitude by which the offline error of DynPopDE was lower than that of CDE, and especially than that of DynDE. As the change period was increased to 5 000 function evaluations, as shown in Figure 5.12, the difference in performance between CDE and DynPopDE narrowed to the point of overlapping confidence intervals. The difference in the number of generations that CDE and DynPopDE performed decreased as enough function evaluations were available for CDE to enter the phase where only one sub-population was evolved per generation.

DynPopDE scaled better than CDE when the change period was increased to values

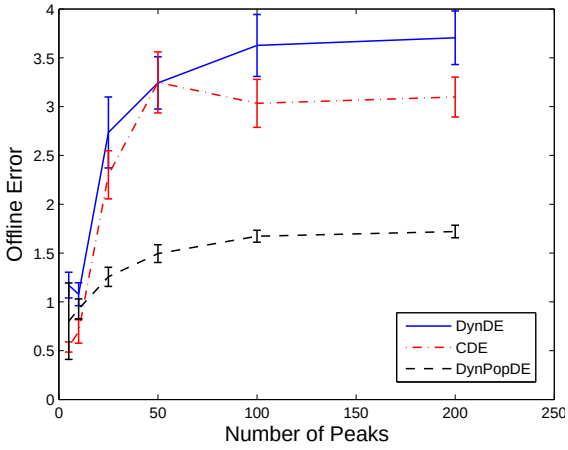

Figure 5.7: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 5 dimensions.

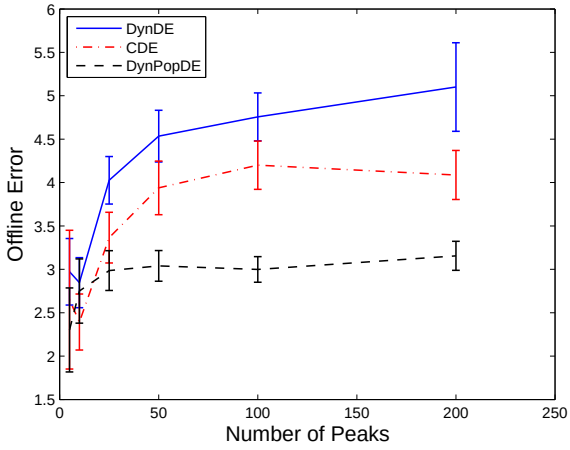

Figure 5.8: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 10 dimensions.

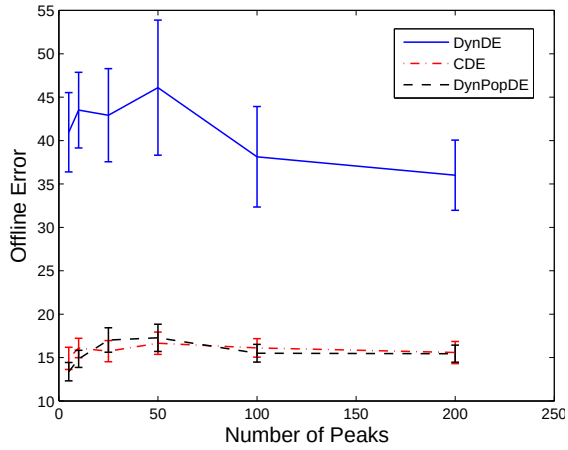

Figure 5.9: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 50 dimensions.

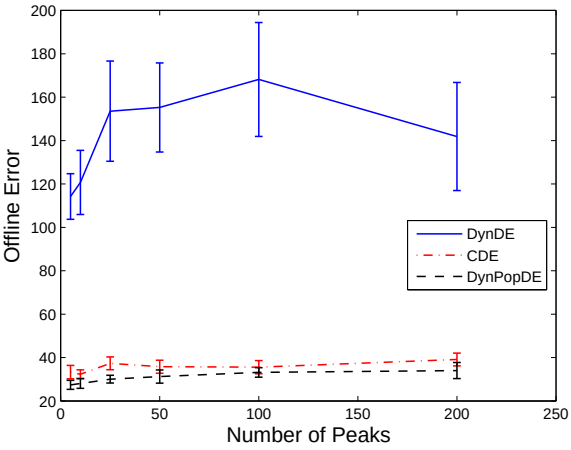

Figure 5.10: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 100 dimensions.

greater than 5 000. Figure 5.13 shows the performance of DynDE, CDE and DynPopDE when a change period of 25 000 was used. DynPopDE yielded considerably lower offline errors for larger numbers of peaks. The difference in performance increased as the change period was increased (refer to Figure 5.14). DynDE and CDE gave similar offline errors at a high change period. This is consistent with results found in Chapter 4 which showed that the difference in performance between DynDE and CDE diminishes as the change period is increased.

This research question compared the scalability of DynDE, CDE and DynPopDE relative to the number of optima in the environment. The experimental results indicate that CDE, generally, scaled better than DynDE, and that DynPopDE yielded further improved results. DynPopDE scaled better than CDE, especially in environments with a very low or high change period, and in low dimensions. The next research question investigates whether the differences between DynPopDE and the other two algorithms are statistically significant.

5.3.4 Research Question 3

Does DynPopDE perform better than CDE and DynDE on problems where the number of optima is constant, but unknown?

The previous section identified trends that suggest that DynPopDE is more scalable than DynDE and CDE in terms of number of optima in the environment. This research question aims to determine whether DynPopDE is indeed more effective than the other two algorithms.

A performance analysis was conducted on the results of DynDE, CDE and DynPopDE on the n_p standard set of experiments (defined in Section 5.3.1.1). The offline error of DynPopDE was compared to that of DynDE and CDE. The number of times that each algorithm performed statistically significantly better than the other (at a 95% confidence level according to a Mann-Whitney U test), was counted. Section 5.3.4.1 discusses the comparison of DynPopDE to DynDE, while Section 5.3.4.2 discusses the comparison of DynPopDE to CDE.

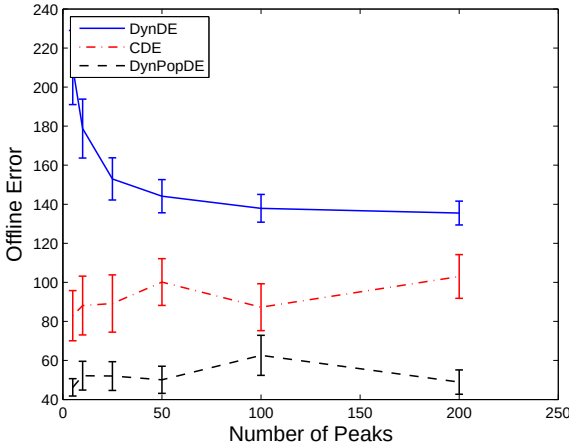

Figure 5.11: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of number of peaks in 25 dimensions.

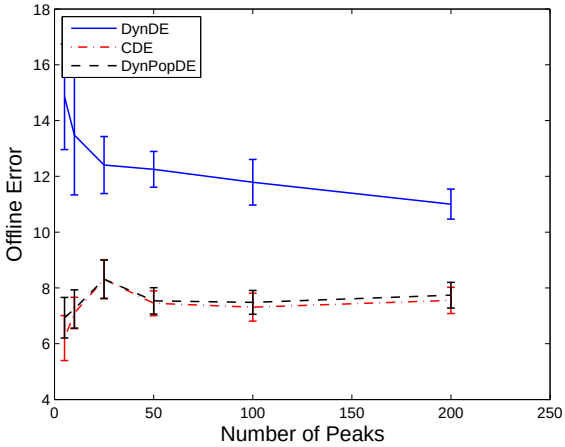

Figure 5.12: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of number of peaks in 25 dimensions.

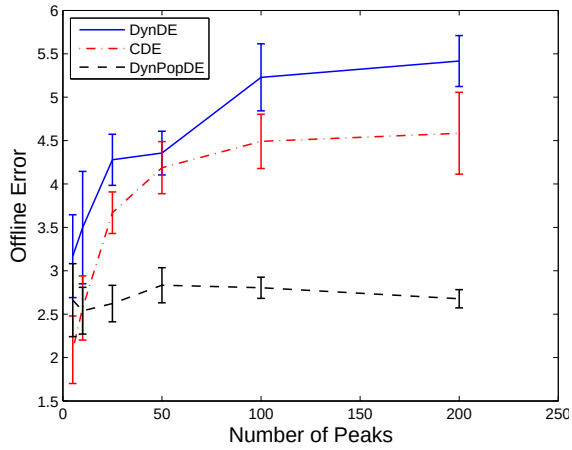

Figure 5.13: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 25 000 for various settings of number of peaks in 25 dimensions.

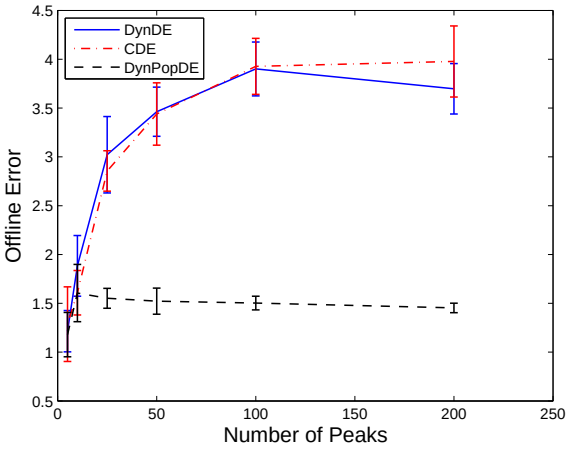

Figure 5.14: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 100 000 for various settings of number of peaks in 25 dimensions.

5.3.4.1 DynPopDE compared to DynDE on the n_p standard set

The result of the performance analysis of DynPopDE, compared to DynDE, is given in Table 5.4. The results are stratified into different settings for number of peaks, underlying function, change period and number of dimensions. Cells in which DynPopDE performed better than DynDE more often, are shaded and printed in boldface. Cells in which DynDE performed better than DynPopDE more often, are printed in italics. DynPopDE performed statistically significantly better than DynDE in 410 of the 480 experiments, and worse in only 21 experiments. The majority of the cases where DynDE performed better than DynPopDE were isolated in 100 dimensional experiments and this occurrence was more prevalent at lower change periods. Despite this, DynPopDE performed significantly better than DynDE in 60 of the 100 dimensional experiments, while DynDE outperformed DynPopDE in only 15 of the 100 dimensional experiments.

The average percentage improvement (API), calculated as in equation (4.6) on page 148, of DynPopDE over DynDE was calculated to determine how much better, on average, DynPopDE is than DynDE. The average percentage improvement of DynPopDE over DynDE over all experiments was found to be 42.04%. The APIs per dimension were found to be 52.46%, 45.07%, 49.12%, 49.66% and 13.90% for 5, 10, 25, 50 and 100 dimensions respectively. Greater improvements in offline error were thus found in lower dimensions. The APIs per change period were found to be 18.84%, 39.49%, 43.06%, 45.39%, 46.01%, 47.28%, 47.60% and 48.68% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations, respectively. The magnitude of the improvement over DynDE thus increased as the change period was increased. The APIs, in terms of the number of peaks, were found to be 34.36%, 26.09%, 43.97%, 47.54%, 49.96% and 50.35% for 5, 10, 25, 50, 100 and 200 peaks respectively. The percentage improvement increased as the number of peaks increased, with the exception of the 10 peak case. DynDE used 10 sub-populations and thus had an advantage when 10 peaks were present in the environment. This analysis indicates that DynPopDE is a more effective algorithm than DynDE over several settings of the number of peaks.

Table 5.4: DynPopDE vs DynDE performance analysis on the n_p standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	*↑0 ↓1*	↑1 ↓1	↑0 ↓0	**↑1 ↓0**	**↑2 ↓0**	**↑9 ↓2**
25	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
100	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
200	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
C	(6)	**↑4 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑43 ↓0**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓0**	**↑5 ↓0**	**↑6 ↓0**	**↑43 ↓2**
All	(12)	**↑10 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑10 ↓1**	**↑11 ↓1**	**↑10 ↓0**	**↑11 ↓0**	**↑12 ↓0**	**↑86 ↓2**
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑13 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑12 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑16 ↓0**
200	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
C	(6)	**↑4 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑39 ↓0**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑10 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑11 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑87 ↓0**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑14 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
100	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
C	(6)	*↑1 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑40 ↓2**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑7 ↓2**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑11 ↓0**	**↑10 ↓0**	**↑88 ↓2**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑14 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
50	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
100	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
C	(6)	*↑0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓0**	**↑41 ↓2**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑6 ↓2**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑12 ↓0**	**↑11 ↓0**	**↑89 ↓2**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	↑0 ↓0	**↑6 ↓5**
10	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓1*	**↑6 ↓5**
25	(2)	↑0 ↓0	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑7 ↓1**
50	(2)	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑2 ↓0**	↑1 ↓0	**↑2 ↓0**	↑1 ↓1	**↑2 ↓0**	**↑11 ↓2**
100	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓1**
C	(6)	*↑0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑4 ↓0**	**↑40 ↓2**
S	(6)	**↑3 ↓0**	*↑2 ↓3*	*↑2 ↓3*	**↑3 ↓2**	↑2 ↓2	**↑3 ↓0**	↑2 ↓2	**↑3 ↓1**	**↑20 ↓13**
All	(12)	**↑3 ↓2**	**↑8 ↓3**	**↑8 ↓3**	**↑9 ↓2**	**↑8 ↓2**	**↑9 ↓0**	**↑8 ↓2**	**↑7 ↓1**	**↑60 ↓15**
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	**↑6 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑8 ↓1**	**↑9 ↓0**	**↑8 ↓1**	**↑6 ↓0**	**↑64 ↓5**
10	(10)	**↑7 ↓0**	**↑9 ↓1**	**↑8 ↓1**	**↑6 ↓2**	**↑8 ↓2**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑55 ↓7**
25	(10)	**↑4 ↓0**	**↑8 ↓0**	**↑9 ↓1**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑9 ↓0**	**↑66 ↓1**
50	(10)	**↑7 ↓1**	**↑9 ↓1**	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓0**	**↑10 ↓0**	**↑9 ↓1**	**↑10 ↓0**	**↑73 ↓3**
100	(10)	**↑6 ↓2**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓2**
200	(10)	**↑6 ↓3**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓3**
C	(30)	**↑9 ↓6**	**↑29 ↓0**	**↑30 ↓0**	**↑28 ↓0**	**↑29 ↓0**	**↑28 ↓0**	**↑27 ↓0**	**↑23 ↓0**	**↑203 ↓6**
S	(30)	**↑27 ↓0**	**↑26 ↓3**	**↑25 ↓3**	**↑26 ↓3**	**↑25 ↓3**	**↑26 ↓0**	**↑25 ↓2**	**↑27 ↓1**	**↑207 ↓15**
All	(60)	**↑36 ↓6**	**↑55 ↓3**	**↑55 ↓3**	**↑54 ↓3**	**↑54 ↓3**	**↑54 ↓0**	**↑52 ↓2**	**↑50 ↓1**	**↑410 ↓21**

5.3.4.2 DynPopDE compared to CDE on the n_p standard set

The result of the performance analysis of DynPopDE compared to CDE on the n_p standard set of experiments is given in Table 5.5. DynPopDE performed statistically significantly better than CDE in 323 of the 480 experiments and worse in only 49 cases. The scalability study which was conducted under the previous research question found that DynPopDE does not scale well to high dimensional problems. This observation is confirmed in the performance analysis in 100 dimensions, where CDE performed better, more often, than DynPopDE. CDE performed better more often than DynPopDE in 48 of the 96 experiment in 100 dimensions.

The average percentage improvement of DynPopDE over CDE over all experiments was found to be 28.72%. The APIs per dimension were found to be 43.79%, 43.71%, 36.95%, 31.40% and -12.25% for 5, 10, 25, 50 and 100 dimensions respectively. Considerable improvements of DynPopDE over CDE were thus found for most settings of dimension, but CDE outperformed DynPopDE in 100 dimensions. The APIs per change period were found to be 19.18%, 27.58%, 23.71%, 20.64%, 24.87%, 33.16%, 37.33% and 43.30% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations, respectively. The magnitude of the improvement over CDE thus increased as the change period was increased. The APIs, in terms of the number of peaks, were found to be 21.92%, 13.01%, 31.06%, 33.86%, 35.04% and 37.44% for 5, 10, 25, 50, 100 and 200 peaks respectively. The percentage improvement increased as the number of peaks increased, with the exception of the 10 peak case. CDE used 10 sub-populations and thus had an advantage when 10 peaks were present in the environment. DynPopDE was found to be superior to CDE on the n_p standard set of experiments, with the exception of very high dimensional environments, where CDE generally performed better.

5.3.4.3 Summary for Research Question 3

This research question investigated whether DynPopDE is a significantly better algorithm than DynDE and CDE on problems where the number of optima is unknown. DynPopDE was found, generally, to perform better than DynDE and CDE on a wide range of benchmark instances. The magnitudes of improvements were large, especially when a

Table 5.5: DynPopDE vs CDE performance analysis on the n_p standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑13 ↓0**
10	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	**↑1 ↓0**	↑0 ↓0	**↑5 ↓4**
25	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
50	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
100	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
200	(2)	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
C	(6)	**↑5 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑4 ↓1**	**↑5 ↓0**	**↑4 ↓0**	**↑5 ↓0**	**↑4 ↓0**	**↑29 ↓1**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑5 ↓1**	**↑5 ↓1**	**↑5 ↓1**	**↑6 ↓0**	**↑5 ↓0**	**↑44 ↓3**
All	(12)	**↑11 ↓0**	**↑7 ↓0**	**↑7 ↓0**	**↑9 ↓2**	**↑10 ↓1**	**↑9 ↓1**	**↑11 ↓0**	**↑9 ↓0**	**↑73 ↓4**
Set.	**Max**	**10 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓1**
10	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
50	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
100	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑15 ↓0**
200	(2)	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
C	(6)	**↑4 ↓0**	**↑5 ↓0**	↑0 ↓0	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑4 ↓0**	**↑28 ↓1**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑10 ↓0**	**↑11 ↓0**	**↑6 ↓0**	**↑9 ↓0**	**↑10 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓1**
Set.	**Max**	**25 Dimensions**								
n_p 5	(2)	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	↑1 ↓1	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑1 ↓0**	**↑11 ↓2**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
100	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓0**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑14 ↓1**
C	(6)	↑1 ↓1	**↑6 ↓0**	**↑6 ↓0**	*↑0 ↓1*	**↑3 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑4 ↓0**	**↑28 ↓3**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑7 ↓1**	**↑12 ↓0**	**↑12 ↓0**	**↑6 ↓1**	**↑9 ↓0**	**↑10 ↓1**	**↑10 ↓0**	**↑10 ↓0**	**↑76 ↓3**
Set.	**Max**	**50 Dimensions**								
n_p 5	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	↑1 ↓1	**↑1 ↓0**	**↑10 ↓1**
10	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑10 ↓0**
25	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
50	(2)	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓0**
100	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
200	(2)	↑1 ↓1	**↑2 ↓0**	**↑2 ↓0**	**↑1 ↓0**	**↑1 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑2 ↓0**	**↑13 ↓1**
C	(6)	*↑0 ↓2*	**↑6 ↓0**	**↑6 ↓0**	↑0 ↓0	↑0 ↓0	**↑4 ↓0**	**↑4 ↓1**	**↑4 ↓0**	**↑24 ↓3**
S	(6)	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑48 ↓0**
All	(12)	**↑6 ↓2**	**↑12 ↓0**	**↑12 ↓0**	**↑6 ↓0**	**↑6 ↓0**	**↑10 ↓0**	**↑10 ↓1**	**↑10 ↓0**	**↑72 ↓3**
Set.	**Max**	**100 Dimensions**								
n_p 5	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	*↑3 ↓5*
10	(2)	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑2 ↓7*
25	(2)	↑0 ↓0	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	*↑3 ↓7*
50	(2)	**↑1 ↓0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	*↑6 ↓7*
100	(2)	*↑0 ↓1*	**↑1 ↓0**	↑1 ↓1	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	↑1 ↓1	↑1 ↓1	*↑5 ↓7*
200	(2)	↑1 ↓1	**↑2 ↓0**	↑0 ↓0	↑1 ↓1	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	**↑7 ↓5**
C	(6)	*↑0 ↓2*	**↑4 ↓0**	**↑5 ↓0**	**↑5 ↓0**	↑0 ↓0	**↑2 ↓0**	**↑3 ↓0**	**↑4 ↓0**	**↑23 ↓2**
S	(6)	**↑2 ↓0**	*↑1 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑3 ↓36*
All	(12)	↑2 ↓2	**↑5 ↓4**	↑5 ↓5	*↑5 ↓6*	*↑0 ↓5*	*↑2 ↓5*	*↑3 ↓6*	*↑4 ↓5*	*↑26 ↓38*
Set.	**Max**	**All Dimensions**								
n_p 5	(10)	**↑7 ↓0**	**↑9 ↓1**	**↑8 ↓1**	**↑5 ↓2**	**↑5 ↓2**	**↑4 ↓1**	**↑5 ↓2**	**↑4 ↓0**	**↑47 ↓9**
10	(10)	**↑6 ↓0**	**↑7 ↓1**	**↑7 ↓1**	**↑4 ↓3**	**↑3 ↓2**	**↑3 ↓2**	**↑4 ↓1**	**↑3 ↓1**	**↑37 ↓11**
25	(10)	**↑5 ↓0**	**↑7 ↓1**	**↑7 ↓1**	**↑6 ↓1**	**↑6 ↓1**	**↑8 ↓1**	**↑8 ↓1**	**↑9 ↓1**	**↑56 ↓7**
50	(10)	**↑7 ↓0**	**↑8 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑7 ↓1**	**↑8 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑62 ↓7**
100	(10)	**↑5 ↓2**	**↑8 ↓0**	**↑7 ↓1**	**↑6 ↓1**	**↑7 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑60 ↓8**
200	(10)	**↑6 ↓3**	**↑8 ↓0**	**↑6 ↓0**	**↑7 ↓1**	**↑7 ↓0**	**↑9 ↓1**	**↑9 ↓1**	**↑9 ↓1**	**↑61 ↓7**
C	(30)	**↑10 ↓5**	**↑22 ↓0**	**↑18 ↓0**	**↑12 ↓2**	**↑12 ↓1**	**↑18 ↓1**	**↑20 ↓1**	**↑20 ↓0**	**↑132 ↓10**
S	(30)	**↑26 ↓0**	**↑25 ↓4**	**↑24 ↓5**	**↑23 ↓7**	**↑23 ↓6**	**↑23 ↓6**	**↑24 ↓6**	**↑23 ↓5**	**↑191 ↓39**
All	(60)	**↑36 ↓5**	**↑47 ↓4**	**↑42 ↓5**	**↑35 ↓9**	**↑35 ↓7**	**↑41 ↓7**	**↑44 ↓7**	**↑43 ↓5**	**↑323 ↓49**

large number of optima was present in the environment. DynPopDE has the advantage of commencing with a single sub-population, which makes it possible for the algorithm to perform more generations between changes than DynDE and CDE when a low change period is used. DynPopDE also yielded superior results for large change periods, but did not scale well to high dimensions.

5.3.5 Research Question 4

How do DynDE, CDE and DynPopDE scale in terms of the maximum number of optima and percentage change in the environment on dynamic optimisation problems where the number of optima is unknown and fluctuates over time?

The previous two research questions focused on the performance of DynDE, CDE and DynPopDE with respect to the number of optima in the environment. The focus of this research question is on dynamic environments in which the number of optima explicitly changes over time. The extended MPB benchmark was used to evaluate the performance of the three algorithms on the $n_p(t)$ standard set as described in Section 5.3.1.2. This section describes the trends observed in the comparative performance of DynDE, CDE and DynPopDE.

The effect of the maximum number of peaks on offline error was found to be dependent on the number of dimensions and the change period. Three distinct behaviour phases can be observed as the change period is increased. The first phase is shown in Figure 5.15, which depicts the offline errors of DynDE, CDE and DynPopDE for various settings of maximum number of peaks with a 5% change in the number of peaks on the conical peak function in 10 dimensions with a change period of 500 function evaluations. CDE generally performed better than DynDE during this phase, with DynPopDE generally performing better than CDE. The offline errors of all the algorithms decreased as the maximum number of peaks was increased. This is due to the fact that at these low change periods the average chances of locating optima are increased when more optima are present in the environment.

The second phase can be observed in Figure 5.16, which shows the same information as Figure 5.15, but with a change period of 5 000. This phase is characterised by high standard deviations in the experimental results (compare the confidence bars of Figure

5.16 to that of Figure 5.17), and, while DynDE generally yielded higher offline errors than CDE and DynPopDE during this phase, neither CDE nor DynPopDE was obviously better.

The third phase occurred at a higher change period than phase two, and is characterised by lower offline errors and a clearer differentiation between DynPopDE and the other algorithms. Phase three can be observed in Figures 5.17 and 5.18, which shows the offline errors when using change periods of 25 000 and 100 000, respectively. The offline error typically increased as the maximum number of peaks was increased during phase three. DynPopDE generally performed better than DynDE and CDE, especially for high values of maximum number of peaks. DynDE and CDE exhibited similar scaling behaviour during phase three, while CDE generally performed slightly better than DynDE.

The change period ranges in which each of the three phases occur, were found to be dependent on the number of dimensions. Phase one did not clearly exist in five dimensions, but appeared for change periods of less than 5 000, in dimensions higher than five. Phase two started directly after phase one and ended with increasingly higher change periods as the number of dimensions was increased. The transition from phase two to phase three occured at a change period of about 5 000 in 5 dimensions, 10 000 in 10 dimensions, 25 000 in 25 dimensions, and exhibited a broad transition period in 50 and 100 dimensions.

The percentage change in the number of peaks had a considerable impact on the offline error of all the algorithms, with the offline errors increasing as the percentage change was increased. Figure 5.19 shows the offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks. This figure illustrates the relative performance of the algorithms during phase one, and DynPopDE accordingly yielded lower offline errors. Figures 5.20 and 5.21 shows the offline errors at change periods of 5 000 and 25 000 respectively. This corresponds to phase two, and CDE performed the best for all of the larger percentage changes. DynPopDE performed the best for percentage changes lower than 40% in Figure 5.22, which shows the offline errors at a change period of 100 000, corresponding to phase three.

This research question investigated the scalability of DynDE, CDE and DynPopDE on DOPs with fluctuating numbers of peaks. The experimental results showed that Dyn-

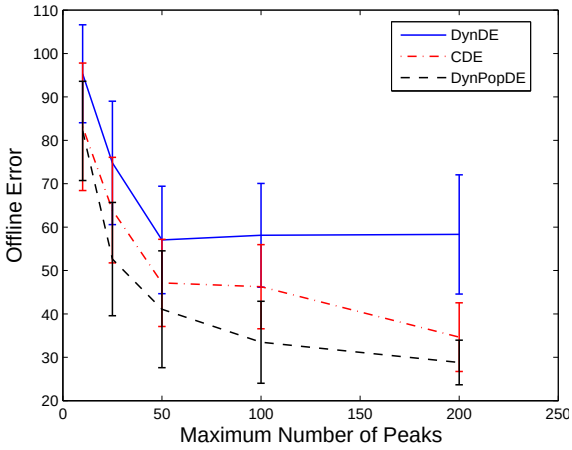

Figure 5.15: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

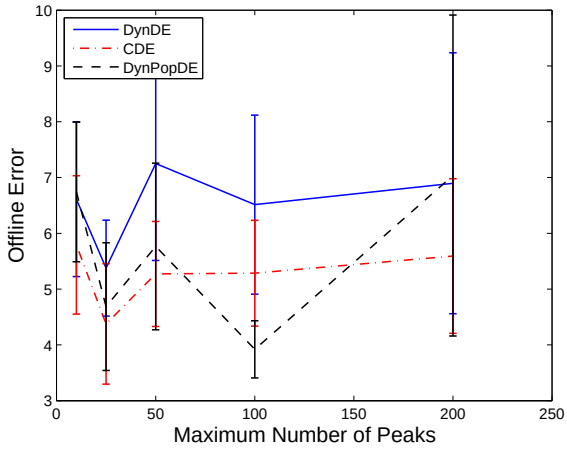

Figure 5.16: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

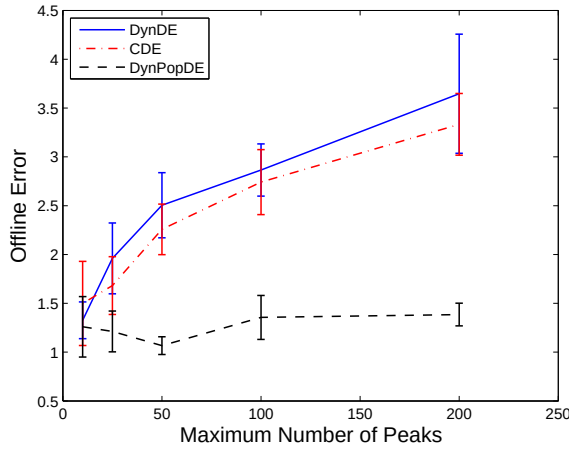

Figure 5.17: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 25 000 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

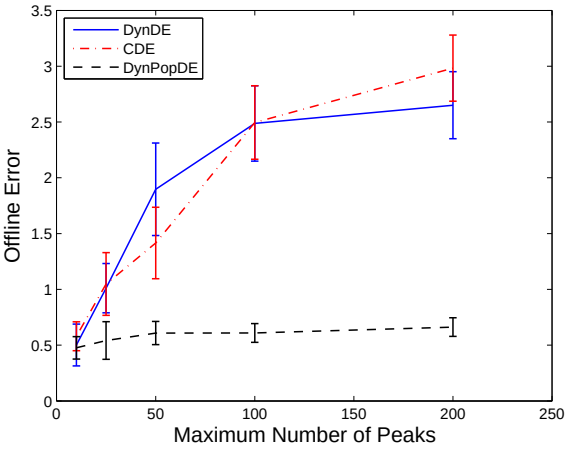

Figure 5.18: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 100 000 for various settings of maximum number of peaks in 10 dimensions with 5% change in the number of peaks

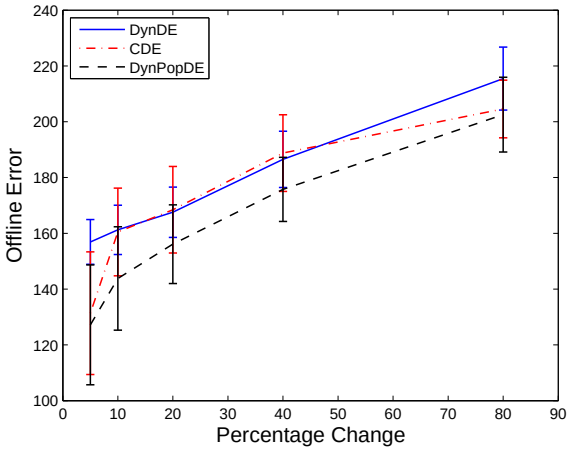

Figure 5.19: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 500 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

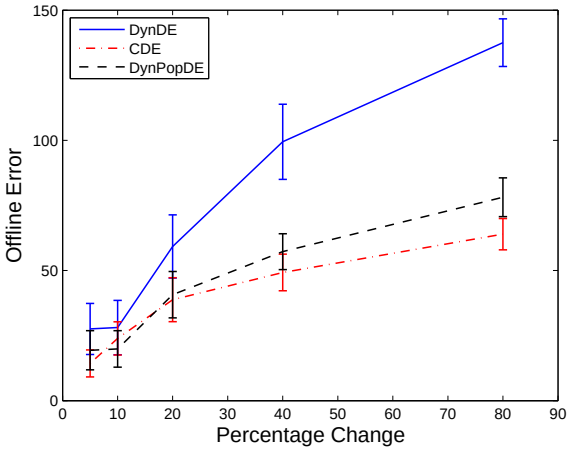

Figure 5.20: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 5 000 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

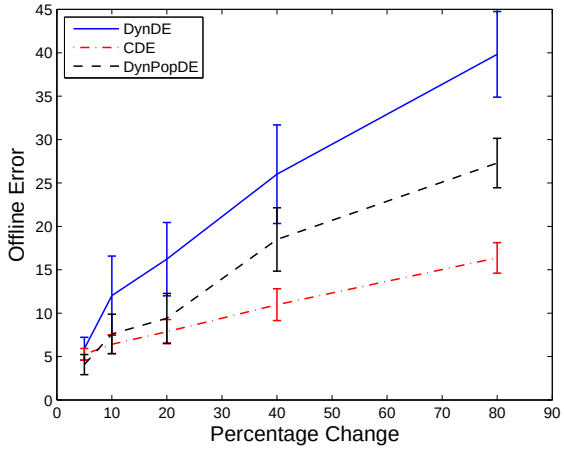

Figure 5.21: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 25 000 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

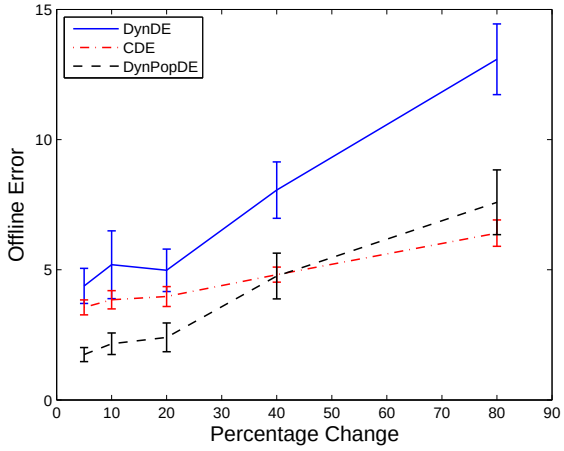

Figure 5.22: Offline errors of DynDE, CDE, and DynPopDE on the conical peak function with a change period of 100 000 for various settings of percentage change in the number of peaks in 25 dimensions with a maximum of 200 peaks

PopDE generally scaled better than DynDE and CPE at low change periods and at high change periods, but not at intermediate change periods. The focus of the next research question is to determine whether the differences between DynDE, CDE and DynPopDE are statistically significant.

5.3.6 Research Question 5

Is DynPopDE more effective than DynDE and CDE on dynamic problems where the number of optima is unknown and fluctuates over time?

The offline errors yielded by DynPopDE on the $n_p(t)$ standard set were compared to those yielded by DynDE and CDE. Table 5.6 gives the performance analysis of DynPopDE compared to DynDE, stratified with respect to change period, number of dimensions, peak function, percentage change in the number of peaks, and the maximum number of peaks. Each cell gives the number of times that DynPopDE performed better than DynDE (indicated by $\uparrow$), and the number of times that DynDE performed better than DynPopDE (indicated by $\downarrow$). Results that did not differ according to a Mann-Whitney U test at a 95% confidence level were not included in the respective totals.

DynPopDE performed significantly better than DynDE in 1 289 of the 2 000 experiments and worse in only 121 experiments. The cases where DynDE performed better than DynPopDE were concentrated in 100 dimensions and at lower change periods. DynDE and DynPopDE yielded similar results at a change period of 100.

The average percentage improvement of DynPopDE over DynDE over all experiments was found to be 29.01%. The APIs per dimension were found to be 45.01%, 33.41%, 30.63%, 26.98% and 9.04% for 5, 10, 25, 50 and 100 dimensions respectively. Larger improvements in offline error were thus found in lower dimensions. DynPopDE did result in a positive API in 100 dimensions, despite the fact that DynDE performed better more often than DynPopDE. The APIs per change period were found to be 1.28%, 15.18%, 19.38%, 30.75%, 32.84%, 35.19%, 44.17% and 53.33% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The improvement of DynPopDE thus increased as the change period was increased. The API values and the fact that DynPopDE performed significantly better than DynDE in the majority of experimental cases, leads to the conclusion that DynPopDE is superior to

Table 5.6: DynPopDE vs DynDE performance analysis on the $n_p(t)$ standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
M_{np} 10	(10)	↑*0* ↓*2*	↑**7** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**56** ↓**2**
25	(10)	↑0 ↓0	↑**6** ↓**0**	↑**5** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**60** ↓**0**
50	(10)	↑*0* ↓*1*	↑**7** ↓**1**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**64** ↓**2**
100	(10)	↑**1** ↓**0**	↑**4** ↓**0**	↑**4** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**59** ↓**0**
200	(10)	↑**2** ↓**0**	↑**6** ↓**1**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**65** ↓**1**
pc 5	(10)	↑**2** ↓**0**	↑**7** ↓**1**	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**67** ↓**1**
10	(10)	↑0 ↓0	↑**5** ↓**1**	↑**4** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**59** ↓**1**
20	(10)	↑1 ↓1	↑**3** ↓**0**	↑**2** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**56** ↓**1**
40	(10)	↑0 ↓0	↑**8** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**61** ↓**0**
80	(10)	↑*0* ↓*2*	↑**7** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**61** ↓**2**
C	(25)	↑*0* ↓*1*	↑**17** ↓**0**	↑**15** ↓**0**	↑**22** ↓**0**	↑**23** ↓**0**	↑**25** ↓**0**	↑**24** ↓**0**	↑**24** ↓**0**	↑**150** ↓**1**
S	(25)	↑**3** ↓**2**	↑**13** ↓**2**	↑**13** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**154** ↓**4**
All	(50)	↑3 ↓3	↑**30** ↓**2**	↑**28** ↓**0**	↑**47** ↓**0**	↑**48** ↓**0**	↑**50** ↓**0**	↑**49** ↓**0**	↑**49** ↓**0**	↑**304** ↓**5**
Set.	**Max**	**10 Dimensions**								
M_{np} 10	(10)	↑**2** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**8** ↓**0**	↑**5** ↓**1**	↑**6** ↓**0**	↑**5** ↓**2**	↑**44** ↓**3**
25	(10)	↑**2** ↓**1**	↑**8** ↓**0**	↑**6** ↓**0**	↑**6** ↓**1**	↑**6** ↓**1**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**57** ↓**3**
50	(10)	↑1 ↓1	↑**5** ↓**0**	↑**4** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**9** ↓**1**	↑**10** ↓**0**	↑**10** ↓**0**	↑**54** ↓**2**
100	(10)	↑**3** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**65** ↓**0**
200	(10)	↑**1** ↓**0**	↑**9** ↓**0**	↑**6** ↓**1**	↑**7** ↓**1**	↑**10** ↓**0**	↑**9** ↓**1**	↑**10** ↓**0**	↑**10** ↓**0**	↑**62** ↓**3**
pc 5	(10)	↑**5** ↓**0**	↑**6** ↓**0**	↑**4** ↓**1**	↑**7** ↓**2**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**59** ↓**3**
10	(10)	↑0 ↓0	↑**7** ↓**0**	↑**4** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**7** ↓**1**	↑**9** ↓**0**	↑**9** ↓**0**	↑**51** ↓**1**
20	(10)	↑**1** ↓**0**	↑**8** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**8** ↓**1**	↑**8** ↓**1**	↑**9** ↓**0**	↑**9** ↓**1**	↑**52** ↓**3**
40	(10)	↑**2** ↓**1**	↑**7** ↓**0**	↑**8** ↓**0**	↑**5** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**9** ↓**1**	↑**56** ↓**2**
80	(10)	↑1 ↓1	↑**10** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**9** ↓**1**	↑**10** ↓**0**	↑**9** ↓**0**	↑**64** ↓**2**
C	(25)	↑**6** ↓**2**	↑**21** ↓**0**	↑**19** ↓**0**	↑**17** ↓**1**	↑**21** ↓**0**	↑**20** ↓**0**	↑**21** ↓**0**	↑**20** ↓**2**	↑**145** ↓**5**
S	(25)	↑**3** ↓**0**	↑**17** ↓**0**	↑**10** ↓**1**	↑**14** ↓**1**	↑**21** ↓**1**	↑**22** ↓**3**	↑**25** ↓**0**	↑**25** ↓**0**	↑**137** ↓**6**
All	(50)	↑**9** ↓**2**	↑**38** ↓**0**	↑**29** ↓**1**	↑**31** ↓**2**	↑**42** ↓**1**	↑**42** ↓**3**	↑**46** ↓**0**	↑**45** ↓**2**	↑**282** ↓**11**
Set.	**Max**	**25 Dimensions**								
M_{np} 10	(10)	↑0 ↓0	↑**9** ↓**0**	↑**10** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**59** ↓**0**
25	(10)	↑0 ↓0	↑**10** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**56** ↓**0**
50	(10)	↑1 ↓1	↑**8** ↓**0**	↑**8** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**59** ↓**1**
100	(10)	↑*0* ↓*1*	↑**7** ↓**0**	↑**9** ↓**1**	↑**8** ↓**0**	↑**7** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**57** ↓**2**
200	(10)	↑**2** ↓**0**	↑**4** ↓**0**	↑**7** ↓**1**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**1**	↑**8** ↓**1**	↑**9** ↓**0**	↑**51** ↓**3**
pc 5	(10)	↑**2** ↓**0**	↑**9** ↓**0**	↑**8** ↓**2**	↑**10** ↓**0**	↑**10** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**67** ↓**2**
10	(10)	↑**1** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**56** ↓**0**
20	(10)	↑0 ↓0	↑**6** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**4** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**49** ↓**0**
40	(10)	↑*0* ↓*1*	↑**9** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**9** ↓**0**	↑**9** ↓**0**	↑**54** ↓**2**
80	(10)	↑*0* ↓*1*	↑**7** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**6** ↓**0**	↑**8** ↓**1**	↑**9** ↓**0**	↑**56** ↓**2**
C	(25)	↑*0* ↓*2*	↑**18** ↓**0**	↑**25** ↓**0**	↑**22** ↓**0**	↑**21** ↓**0**	↑**18** ↓**0**	↑**25** ↓**0**	↑**24** ↓**0**	↑**153** ↓**2**
S	(25)	↑**3** ↓**0**	↑**20** ↓**0**	↑**17** ↓**2**	↑**17** ↓**0**	↑**11** ↓**0**	↑**15** ↓**1**	↑**22** ↓**1**	↑**24** ↓**0**	↑**129** ↓**4**
All	(50)	↑**3** ↓**2**	↑**38** ↓**0**	↑**42** ↓**2**	↑**39** ↓**0**	↑**32** ↓**0**	↑**33** ↓**1**	↑**47** ↓**1**	↑**48** ↓**0**	↑**282** ↓**6**
Set.	**Max**	**50 Dimensions**								
M_{np} 10	(10)	↑**1** ↓**0**	↑**8** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**8** ↓**0**	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**62** ↓**0**
25	(10)	↑**1** ↓**0**	↑**4** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**5** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**55** ↓**0**
50	(10)	↑1 ↓1	↑**7** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**7** ↓**1**	↑**6** ↓**0**	↑**6** ↓**0**	↑**10** ↓**0**	↑**56** ↓**2**
100	(10)	↑*1* ↓*3*	↑**7** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**6** ↓**0**	↑**5** ↓**0**	↑**6** ↓**1**	↑**8** ↓**0**	↑**50** ↓**4**
200	(10)	↑*0* ↓*2*	↑**6** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**6** ↓**0**	↑**44** ↓**2**
pc 5	(10)	↑**1** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**9** ↓**0**	↑**7** ↓**0**	↑**10** ↓**0**	↑**10** ↓**0**	↑**63** ↓**0**
10	(10)	↑**3** ↓**2**	↑**8** ↓**0**	↑**8** ↓**0**	↑**9** ↓**0**	↑**8** ↓**1**	↑**7** ↓**0**	↑**7** ↓**1**	↑**9** ↓**0**	↑**59** ↓**4**
20	(10)	↑*0* ↓*1*	↑**5** ↓**0**	↑**6** ↓**0**	↑**9** ↓**0**	↑**3** ↓**0**	↑**2** ↓**0**	↑**4** ↓**0**	↑**9** ↓**0**	↑**38** ↓**1**
40	(10)	↑0 ↓0	↑**6** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**8** ↓**0**	↑**6** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**52** ↓**0**
80	(10)	↑*0* ↓*3*	↑**6** ↓**0**	↑**9** ↓**0**	↑**10** ↓**0**	↑**9** ↓**0**	↑**6** ↓**0**	↑**7** ↓**0**	↑**8** ↓**0**	↑**55** ↓**3**
C	(25)	↑*0* ↓*4*	↑**12** ↓**0**	↑**21** ↓**0**	↑**25** ↓**0**	↑**21** ↓**0**	↑**16** ↓**0**	↑**17** ↓**1**	↑**23** ↓**0**	↑**135** ↓**5**
S	(25)	↑**4** ↓**2**	↑**20** ↓**0**	↑**21** ↓**0**	↑**22** ↓**0**	↑**16** ↓**1**	↑**12** ↓**0**	↑**17** ↓**0**	↑**20** ↓**0**	↑**132** ↓**3**
All	(50)	↑*4* ↓*6*	↑**32** ↓**0**	↑**42** ↓**0**	↑**47** ↓**0**	↑**37** ↓**1**	↑**28** ↓**0**	↑**34** ↓**1**	↑**43** ↓**0**	↑**267** ↓**8**
Set.	**Max**	**100 Dimensions**								
M_{np} 10	(10)	↑0 ↓0	↑*0* ↓*3*	↑*4* ↓*5*	↑5 ↓5	↑5 ↓5	↑4 ↓4	↑**3** ↓**0**	↑**8** ↓**0**	↑**29** ↓**22**
25	(10)	↑0 ↓0	↑*2* ↓*4*	↑4 ↓4	↑5 ↓5	↑**5** ↓**4**	↑**4** ↓**2**	↑**4** ↓**1**	↑**8** ↓**0**	↑**32** ↓**20**
50	(10)	↑*0* ↓*1*	↑*1* ↓*5*	↑*2* ↓*3*	↑**5** ↓**3**	↑**5** ↓**2**	↑**4** ↓**3**	↑**5** ↓**0**	↑**8** ↓**0**	↑**30** ↓**17**
100	(10)	↑*0* ↓*1*	↑*3* ↓*4*	↑**4** ↓**2**	↑**5** ↓**4**	↑**6** ↓**2**	↑**5** ↓**3**	↑**3** ↓**1**	↑**7** ↓**0**	↑**33** ↓**17**
200	(10)	↑*0* ↓*1*	↑**3** ↓**2**	↑3 ↓3	↑**5** ↓**2**	↑**5** ↓**3**	↑**4** ↓**2**	↑**5** ↓**1**	↑**5** ↓**1**	↑**30** ↓**15**
pc 5	(10)	↑0 ↓0	↑2 ↓2	↑**5** ↓**2**	↑**5** ↓**2**	↑**6** ↓**3**	↑**5** ↓**4**	↑**7** ↓**1**	↑**8** ↓**0**	↑**38** ↓**14**
10	(10)	↑0 ↓0	↑3 ↓3	↑**4** ↓**1**	↑**5** ↓**3**	↑**5** ↓**1**	↑**4** ↓**0**	↑**3** ↓**0**	↑**8** ↓**0**	↑**32** ↓**8**
20	(10)	↑0 ↓0	↑*0* ↓*4*	↑*2* ↓*4*	↑**5** ↓**4**	↑**5** ↓**2**	↑**2** ↓**1**	↑**1** ↓**0**	↑**8** ↓**0**	↑**23** ↓**15**
40	(10)	↑0 ↓0	↑*2* ↓*4*	↑*2* ↓*5*	↑5 ↓5	↑5 ↓5	↑**5** ↓**4**	↑**4** ↓**0**	↑**4** ↓**0**	↑**27** ↓**23**
80	(10)	↑*0* ↓*3*	↑*2* ↓*5*	↑*4* ↓*5*	↑5 ↓5	↑5 ↓5	↑5 ↓5	↑**5** ↓**2**	↑**8** ↓**1**	↑**34** ↓**31**
C	(25)	↑0 ↓0	↑**9** ↓**0**	↑**17** ↓**0**	↑**25** ↓**0**	↑**25** ↓**0**	↑**21** ↓**0**	↑**18** ↓**0**	↑**18** ↓**0**	↑**133** ↓**0**
S	(25)	↑*0* ↓*3*	↑*0* ↓*18*	↑*0* ↓*17*	↑*0* ↓*19*	↑*1* ↓*16*	↑*0* ↓*14*	↑*2* ↓*3*	↑**18** ↓**1**	↑*21* ↓*91*
All	(50)	↑*0* ↓*3*	↑*9* ↓*18*	↑17 ↓17	↑**25** ↓**19**	↑**26** ↓**16**	↑**21** ↓**14**	↑**20** ↓**3**	↑**36** ↓**1**	↑**154** ↓**91**
Set.	**Max**	**All Dimensions**								
M_{np} 10	(50)	↑**3** ↓**2**	↑**32** ↓**3**	↑**35** ↓**5**	↑**34** ↓**5**	↑**36** ↓**5**	↑**32** ↓**5**	↑**37** ↓**0**	↑**41** ↓**2**	↑**250** ↓**27**
25	(50)	↑**3** ↓**1**	↑**30** ↓**4**	↑**33** ↓**4**	↑**36** ↓**6**	↑**35** ↓**5**	↑**34** ↓**2**	↑**42** ↓**1**	↑**47** ↓**0**	↑**260** ↓**23**
50	(50)	↑*3* ↓*5*	↑**28** ↓**6**	↑**30** ↓**3**	↑**39** ↓**3**	↑**37** ↓**3**	↑**37** ↓**4**	↑**41** ↓**0**	↑**48** ↓**0**	↑**263** ↓**24**
100	(50)	↑5 ↓5	↑**29** ↓**4**	↑**31** ↓**3**	↑**41** ↓**4**	↑**38** ↓**2**	↑**37** ↓**3**	↑**38** ↓**2**	↑**45** ↓**0**	↑**264** ↓**23**
200	(50)	↑**5** ↓**3**	↑**28** ↓**3**	↑**29** ↓**5**	↑**39** ↓**3**	↑**39** ↓**3**	↑**34** ↓**4**	↑**38** ↓**2**	↑**40** ↓**1**	↑**252** ↓**24**
pc 5	(50)	↑**10** ↓**0**	↑**31** ↓**3**	↑**35** ↓**5**	↑**41** ↓**4**	↑**45** ↓**3**	↑**39** ↓**4**	↑**46** ↓**1**	↑**47** ↓**0**	↑**294** ↓**20**
10	(50)	↑**4** ↓**2**	↑**30** ↓**4**	↑**28** ↓**1**	↑**38** ↓**3**	↑**37** ↓**2**	↑**35** ↓**1**	↑**39** ↓**1**	↑**46** ↓**0**	↑**257** ↓**14**
20	(50)	↑2 ↓2	↑**22** ↓**4**	↑**22** ↓**4**	↑**33** ↓**4**	↑**30** ↓**3**	↑**29** ↓**2**	↑**34** ↓**0**	↑**46** ↓**1**	↑**218** ↓**20**
40	(50)	↑2 ↓2	↑**32** ↓**4**	↑**34** ↓**5**	↑**36** ↓**5**	↑**34** ↓**5**	↑**35** ↓**5**	↑**38** ↓**0**	↑**39** ↓**1**	↑**250** ↓**27**
80	(50)	↑*1* ↓*10*	↑**32** ↓**5**	↑**39** ↓**5**	↑**41** ↓**5**	↑**39** ↓**5**	↑**36** ↓**6**	↑**39** ↓**3**	↑**43** ↓**1**	↑**270** ↓**40**
C	(125)	↑*6* ↓*9*	↑**77** ↓**0**	↑**97** ↓**0**	↑**111** ↓**1**	↑**111** ↓**0**	↑**100** ↓**0**	↑**105** ↓**1**	↑**109** ↓**2**	↑**716** ↓**13**
S	(125)	↑**13** ↓**7**	↑**70** ↓**20**	↑**61** ↓**20**	↑**78** ↓**20**	↑**74** ↓**18**	↑**74** ↓**18**	↑**91** ↓**4**	↑**112** ↓**1**	↑**573** ↓**108**
All	(250)	↑**19** ↓**16**	↑**147** ↓**20**	↑**158** ↓**20**	↑**189** ↓**21**	↑**185** ↓**18**	↑**174** ↓**18**	↑**196** ↓**5**	↑**221** ↓**3**	↑**1289** ↓**121**

DynDE on problems where the number of optima fluctuates over time.

The same performance analysis that was conducted for DynPopDE versus DynDE was repeated for DynPopDE versus CDE. The results are given in Table 5.7. The analysis shows that the average offline errors of DynPopDE and CDE did not differ statistically significantly in the majority of experimental environments. DynPopDE did, however, perform statistically significantly better than CDE in 719 of the 2 000 experiments, and performed worse in only 249 experiments.

The three phases that were identified in Section 5.3.5 can be observed in Table 5.7 by looking at the clusters of shaded cells (which indicate settings where DynPopDE performed better more often than CDE). Phase one, in which DynPopDE performed better than CDE at low change periods, is visible in the change period of 100 and 500 columns. Phase two, in which CDE typically performed better than DynPopDE, is visible in the intermediate range of change periods which broadens in higher dimensions. Phase three, in which DynPopDE typically performed better than CDE at high change periods, is most visible in low dimensions.

The average percentage improvement of DynPopDE over CDE over all experiments was found to be 12.45%. The APIs per dimension were found to be 37.69%, 23.18%, 6.11%, -3.42% and -1.29% for 5, 10, 25, 50 and 100 dimensions respectively. DynPopDE was thus inferior on high dimensions, but by a small margin. The APIs per change period, were found to be 1.82%, 9.01%, 5.79%, 1.85%, 5.84%, 11.22%, 26.04% and 38.06% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. The improvement of DynPopDE thus increased as the function evaluations was increased. The APIs per setting of maximum number of peaks are 9.25%, 14.38%, 14.92%, 14.19% and 9.54% for values 5, 10, 25, 50, 100 and 200, respectively. The APIs, with regard to percentage change in the number of peaks, were found to be 22.78%, 17.25%, 11.63%, 6.84% and 3.78% for values 5%, 10%, 20%, 40% and 80%, respectively. Large improvements over CDE were thus made for the smaller percentage changes, with only minor improvements for large percentage changes.

The results presented in this section support the conclusion that, in general, DynPopDE is a more effective optimisation algorithm than DynDE and CDE on problems where the number of optima fluctuates over time. DynPopDE performed statistically

Table 5.7: DynPopDE vs CDE performance analysis on the $n_p(t)$ standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
M_{np} 10	(10)	↑0 ↓1	↑3 ↓0	↑3 ↓0	↑4 ↓0	↑6 ↓0	↑4 ↓0	↑6 ↓0	↑4 ↓0	↑30 ↓1
25	(10)	↑1 ↓0	↑1 ↓0	↑0 ↓1	↑7 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑49 ↓1
50	(10)	↑0 ↓1	↑0 ↓2	↑2 ↓0	↑7 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑49 ↓3
100	(10)	↑0 ↓1	↑2 ↓0	↑1 ↓0	↑9 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑52 ↓1
200	(10)	↑2 ↓1	↑2 ↓0	↑3 ↓0	↑8 ↓0	↑9 ↓0	↑10 ↓0	↑10 ↓0	↑10 ↓0	↑54 ↓1
pc 5	(10)	↑1 ↓1	↑4 ↓1	↑4 ↓1	↑8 ↓0	↑8 ↓0	↑8 ↓0	↑9 ↓0	↑9 ↓0	↑51 ↓3
10	(10)	↑1 ↓0	↑2 ↓1	↑2 ↓0	↑9 ↓0	↑10 ↓0	↑9 ↓0	↑9 ↓0	↑8 ↓0	↑50 ↓1
20	(10)	↑1 ↓0	↑0 ↓0	↑0 ↓0	↑9 ↓0	↑9 ↓0	↑8 ↓0	↑9 ↓0	↑9 ↓0	↑45 ↓0
40	(10)	↑0 ↓0	↑2 ↓0	↑1 ↓0	↑5 ↓0	↑9 ↓0	↑9 ↓0	↑9 ↓0	↑9 ↓0	↑44 ↓0
80	(10)	↑0 ↓3	↑0 ↓0	↑2 ↓0	↑4 ↓0	↑9 ↓0	↑10 ↓0	↑10 ↓0	↑9 ↓0	↑44 ↓3
C	(25)	↑1 ↓2	↑0 ↓0	↑2 ↓0	↑14 ↓0	↑21 ↓0	↑21 ↓0	↑21 ↓0	↑20 ↓0	↑100 ↓2
S	(25)	↑2 ↓2	↑8 ↓2	↑7 ↓1	↑21 ↓0	↑24 ↓0	↑23 ↓0	↑25 ↓0	↑24 ↓0	↑134 ↓5
All	(50)	↑3 ↓4	↑8 ↓2	↑9 ↓1	↑35 ↓0	↑45 ↓0	↑44 ↓0	↑46 ↓0	↑44 ↓0	↑234 ↓7
Set.	**Max**	**10 Dimensions**								
M_{np} 10	(10)	↑2 ↓0	↑6 ↓0	↑1 ↓0	↑3 ↓0	↑6 ↓0	↑4 ↓1	↑6 ↓0	↑5 ↓0	↑33 ↓1
25	(10)	↑1 ↓0	↑4 ↓0	↑0 ↓0	↑3 ↓0	↑4 ↓0	↑8 ↓0	↑10 ↓0	↑10 ↓0	↑40 ↓0
50	(10)	↑2 ↓0	↑3 ↓0	↑0 ↓1	↑1 ↓3	↑5 ↓0	↑9 ↓1	↑10 ↓0	↑10 ↓0	↑40 ↓5
100	(10)	↑1 ↓1	↑4 ↓1	↑3 ↓0	↑4 ↓1	↑5 ↓1	↑8 ↓0	↑10 ↓0	↑10 ↓0	↑45 ↓4
200	(10)	↑2 ↓0	↑3 ↓0	↑1 ↓0	↑1 ↓1	↑4 ↓2	↑5 ↓1	↑9 ↓0	↑9 ↓0	↑34 ↓4
pc 5	(10)	↑2 ↓0	↑4 ↓1	↑2 ↓1	↑6 ↓0	↑8 ↓0	↑9 ↓0	↑9 ↓0	↑9 ↓0	↑49 ↓2
10	(10)	↑2 ↓0	↑2 ↓0	↑0 ↓0	↑4 ↓3	↑8 ↓0	↑8 ↓1	↑9 ↓0	↑9 ↓0	↑42 ↓4
20	(10)	↑2 ↓0	↑3 ↓0	↑0 ↓0	↑1 ↓0	↑5 ↓0	↑8 ↓1	↑10 ↓0	↑9 ↓0	↑38 ↓1
40	(10)	↑1 ↓1	↑5 ↓0	↑2 ↓0	↑0 ↓1	↑2 ↓1	↑6 ↓0	↑9 ↓0	↑9 ↓0	↑34 ↓3
80	(10)	↑1 ↓0	↑6 ↓0	↑1 ↓0	↑1 ↓1	↑1 ↓2	↑3 ↓1	↑8 ↓0	↑8 ↓0	↑29 ↓4
C	(25)	↑6 ↓1	↑10 ↓0	↑1 ↓0	↑2 ↓2	↑7 ↓3	↑15 ↓0	↑21 ↓0	↑20 ↓0	↑82 ↓6
S	(25)	↑2 ↓0	↑10 ↓1	↑4 ↓1	↑10 ↓3	↑17 ↓0	↑19 ↓3	↑24 ↓0	↑24 ↓0	↑110 ↓8
All	(50)	↑8 ↓1	↑20 ↓1	↑5 ↓1	↑12 ↓5	↑24 ↓3	↑34 ↓3	↑45 ↓0	↑44 ↓0	↑192 ↓14
Set.	**Max**	**25 Dimensions**								
M_{np} 10	(10)	↑1 ↓0	↑3 ↓0	↑2 ↓0	↑1 ↓1	↑1 ↓1	↑3 ↓2	↑6 ↓0	↑5 ↓1	↑22 ↓5
25	(10)	↑0 ↓0	↑6 ↓0	↑2 ↓0	↑1 ↓2	↑1 ↓3	↑3 ↓1	↑5 ↓0	↑6 ↓0	↑24 ↓6
50	(10)	↑1 ↓0	↑5 ↓0	↑3 ↓0	↑2 ↓1	↑2 ↓1	↑3 ↓1	↑5 ↓0	↑9 ↓0	↑30 ↓3
100	(10)	↑0 ↓0	↑4 ↓0	↑4 ↓0	↑2 ↓0	↑1 ↓3	↑2 ↓3	↑5 ↓2	↑7 ↓0	↑25 ↓8
200	(10)	↑2 ↓2	↑1 ↓0	↑4 ↓0	↑1 ↓2	↑3 ↓4	↑2 ↓5	↑4 ↓4	↑6 ↓1	↑23 ↓18
pc 5	(10)	↑2 ↓1	↑4 ↓0	↑4 ↓0	↑4 ↓1	↑6 ↓0	↑7 ↓1	↑8 ↓0	↑8 ↓0	↑43 ↓3
10	(10)	↑1 ↓0	↑4 ↓0	↑2 ↓0	↑3 ↓0	↑2 ↓1	↑4 ↓2	↑7 ↓1	↑9 ↓0	↑32 ↓4
20	(10)	↑0 ↓1	↑3 ↓0	↑1 ↓0	↑0 ↓1	↑0 ↓0	↑2 ↓0	↑6 ↓0	↑8 ↓1	↑20 ↓3
40	(10)	↑0 ↓0	↑4 ↓0	↑3 ↓0	↑0 ↓1	↑0 ↓4	↑0 ↓2	↑2 ↓1	↑5 ↓0	↑14 ↓8
80	(10)	↑1 ↓0	↑4 ↓0	↑5 ↓0	↑0 ↓3	↑0 ↓7	↑0 ↓7	↑2 ↓4	↑3 ↓1	↑15 ↓22
C	(25)	↑0 ↓2	↑2 ↓0	↑4 ↓0	↑0 ↓5	↑1 ↓8	↑2 ↓8	↑8 ↓3	↑12 ↓1	↑29 ↓27
S	(25)	↑4 ↓0	↑17 ↓0	↑11 ↓0	↑7 ↓1	↑7 ↓4	↑11 ↓4	↑17 ↓3	↑21 ↓1	↑95 ↓13
All	(50)	↑4 ↓2	↑19 ↓0	↑15 ↓0	↑7 ↓6	↑8 ↓12	↑13 ↓12	↑25 ↓6	↑33 ↓2	↑124 ↓40
Set.	**Max**	**50 Dimensions**								
M_{np} 10	(10)	↑2 ↓0	↑6 ↓0	↑3 ↓0	↑0 ↓1	↑1 ↓2	↑2 ↓5	↑4 ↓2	↑5 ↓0	↑23 ↓10
25	(10)	↑0 ↓0	↑4 ↓0	↑2 ↓0	↑1 ↓2	↑2 ↓4	↑1 ↓5	↑3 ↓2	↑5 ↓0	↑18 ↓13
50	(10)	↑0 ↓0	↑6 ↓0	↑6 ↓0	↑2 ↓2	↑1 ↓4	↑1 ↓4	↑3 ↓3	↑7 ↓0	↑26 ↓13
100	(10)	↑0 ↓1	↑6 ↓0	↑2 ↓0	↑1 ↓1	↑1 ↓2	↑1 ↓5	↑0 ↓4	↑4 ↓3	↑15 ↓16
200	(10)	↑1 ↓1	↑6 ↓0	↑2 ↓0	↑2 ↓1	↑1 ↓5	↑1 ↓4	↑1 ↓7	↑3 ↓4	↑17 ↓22
pc 5	(10)	↑2 ↓1	↑7 ↓0	↑3 ↓0	↑5 ↓0	↑3 ↓3	↑5 ↓2	↑3 ↓3	↑9 ↓0	↑37 ↓9
10	(10)	↑1 ↓1	↑6 ↓0	↑4 ↓0	↑1 ↓2	↑3 ↓0	↑1 ↓3	↑4 ↓2	↑7 ↓1	↑27 ↓9
20	(10)	↑0 ↓0	↑5 ↓0	↑2 ↓0	↑0 ↓1	↑0 ↓2	↑0 ↓2	↑3 ↓0	↑5 ↓1	↑15 ↓6
40	(10)	↑0 ↓0	↑4 ↓0	↑4 ↓0	↑0 ↓2	↑0 ↓5	↑0 ↓6	↑1 ↓5	↑2 ↓1	↑11 ↓19
80	(10)	↑0 ↓0	↑6 ↓0	↑2 ↓0	↑0 ↓2	↑0 ↓7	↑0 ↓10	↑0 ↓8	↑1 ↓4	↑9 ↓31
C	(25)	↑0 ↓2	↑9 ↓0	↑5 ↓0	↑1 ↓6	↑0 ↓11	↑0 ↓14	↑0 ↓12	↑9 ↓3	↑24 ↓48
S	(25)	↑3 ↓0	↑19 ↓0	↑10 ↓0	↑5 ↓1	↑6 ↓6	↑6 ↓9	↑11 ↓6	↑15 ↓4	↑75 ↓26
All	(50)	↑3 ↓2	↑28 ↓0	↑15 ↓0	↑6 ↓7	↑6 ↓17	↑6 ↓23	↑11 ↓18	↑24 ↓7	↑99 ↓74
Set.	**Max**	**100 Dimensions**								
M_{np} 10	(10)	↑0 ↓0	↑0 ↓3	↑0 ↓5	↑0 ↓2	↑2 ↓2	↑5 ↓4	↑5 ↓3	↑5 ↓2	↑17 ↓21
25	(10)	↑0 ↓0	↑0 ↓4	↑0 ↓4	↑0 ↓3	↑0 ↓2	↑5 ↓2	↑5 ↓2	↑5 ↓2	↑15 ↓19
50	(10)	↑0 ↓1	↑0 ↓5	↑1 ↓5	↑0 ↓3	↑0 ↓3	↑3 ↓2	↑4 ↓3	↑6 ↓2	↑14 ↓24
100	(10)	↑0 ↓0	↑2 ↓4	↑1 ↓5	↑0 ↓4	↑0 ↓4	↑2 ↓3	↑4 ↓2	↑5 ↓3	↑14 ↓25
200	(10)	↑1 ↓1	↑2 ↓1	↑0 ↓3	↑0 ↓6	↑1 ↓4	↑0 ↓5	↑2 ↓2	↑4 ↓3	↑10 ↓25
pc 5	(10)	↑1 ↓0	↑2 ↓2	↑1 ↓4	↑0 ↓2	↑2 ↓0	↑2 ↓1	↑3 ↓0	↑5 ↓2	↑16 ↓11
10	(10)	↑0 ↓0	↑1 ↓3	↑0 ↓3	↑0 ↓2	↑0 ↓1	↑3 ↓1	↑4 ↓0	↑5 ↓1	↑13 ↓11
20	(10)	↑0 ↓0	↑1 ↓3	↑0 ↓5	↑0 ↓2	↑1 ↓1	↑4 ↓3	↑4 ↓3	↑5 ↓1	↑15 ↓18
40	(10)	↑0 ↓0	↑0 ↓4	↑0 ↓5	↑0 ↓5	↑0 ↓6	↑3 ↓5	↑5 ↓4	↑5 ↓4	↑13 ↓33
80	(10)	↑0 ↓2	↑0 ↓5	↑1 ↓5	↑0 ↓7	↑0 ↓7	↑3 ↓6	↑4 ↓5	↑5 ↓4	↑13 ↓41
C	(25)	↑0 ↓0	↑4 ↓0	↑2 ↓0	↑0 ↓5	↑0 ↓9	↑0 ↓14	↑1 ↓12	↑3 ↓12	↑10 ↓52
S	(25)	↑1 ↓2	↑0 ↓17	↑0 ↓22	↑0 ↓13	↑3 ↓6	↑15 ↓2	↑19 ↓0	↑22 ↓0	↑60 ↓62
All	(50)	↑1 ↓2	↑4 ↓17	↑2 ↓22	↑0 ↓18	↑3 ↓15	↑15 ↓16	↑20 ↓12	↑25 ↓12	↑70 ↓114
Set.	**Max**	**All Dimensions**								
M_{np} 10	(50)	↑5 ↓1	↑18 ↓3	↑9 ↓5	↑8 ↓4	↑16 ↓5	↑18 ↓12	↑27 ↓5	↑24 ↓3	↑125 ↓38
25	(50)	↑2 ↓0	↑15 ↓4	↑4 ↓5	↑12 ↓7	↑17 ↓9	↑27 ↓8	↑33 ↓4	↑36 ↓2	↑146 ↓39
50	(50)	↑3 ↓2	↑14 ↓7	↑12 ↓6	↑12 ↓9	↑18 ↓8	↑26 ↓8	↑32 ↓6	↑42 ↓2	↑159 ↓48
100	(50)	↑1 ↓3	↑18 ↓5	↑11 ↓5	↑16 ↓6	↑17 ↓10	↑23 ↓11	↑29 ↓8	↑36 ↓6	↑151 ↓54
200	(50)	↑8 ↓5	↑14 ↓1	↑10 ↓3	↑12 ↓10	↑18 ↓15	↑18 ↓15	↑26 ↓13	↑32 ↓8	↑138 ↓70
pc 5	(50)	↑8 ↓3	↑21 ↓4	↑14 ↓6	↑23 ↓3	↑27 ↓3	↑31 ↓4	↑32 ↓3	↑40 ↓2	↑196 ↓28
10	(50)	↑5 ↓1	↑15 ↓4	↑8 ↓3	↑17 ↓7	↑23 ↓2	↑25 ↓7	↑33 ↓3	↑38 ↓2	↑164 ↓29
20	(50)	↑3 ↓1	↑12 ↓3	↑3 ↓5	↑10 ↓4	↑15 ↓3	↑22 ↓6	↑32 ↓3	↑36 ↓3	↑133 ↓28
40	(50)	↑1 ↓1	↑15 ↓4	↑10 ↓5	↑5 ↓9	↑11 ↓16	↑18 ↓13	↑26 ↓10	↑30 ↓5	↑116 ↓63
80	(50)	↑2 ↓5	↑16 ↓5	↑11 ↓5	↑5 ↓13	↑10 ↓23	↑16 ↓24	↑24 ↓17	↑26 ↓9	↑110 ↓101
C	(125)	↑7 ↓7	↑25 ↓0	↑14 ↓0	↑17 ↓18	↑29 ↓31	↑38 ↓36	↑51 ↓27	↑64 ↓16	↑245 ↓135
S	(125)	↑12 ↓4	↑54 ↓20	↑32 ↓24	↑43 ↓18	↑57 ↓16	↑74 ↓18	↑96 ↓9	↑106 ↓5	↑474 ↓114
All	(250)	↑19 ↓11	↑79 ↓20	↑46 ↓24	↑60 ↓36	↑86 ↓47	↑112 ↓54	↑147 ↓36	↑170 ↓21	↑719 ↓249

significantly better more often than both the other algorithms. Furthermore, with the exception of high dimensional problems, DynPopDE yielded large percentage improvements over DynDE and CDE.

5.3.7 Research Question 6

What is the convergence profile of DynPopDE?

The previous sections found that DynPopDE is, generally, a better optimisation algorithm than DynDE and CDE on the environments which are the focus of this chapter. This research question compares the convergence behaviour, in terms of current error, offline error and diversity, of DynPopDE to that of DynDE and CDE. This research question also investigates the number of sub-populations that DynPopDE uses on various environments.

Figure 5.23 gives the offline and current errors of DynDE, CDE and DynPopDE on the MPB using the Scenario 2 settings (i.e. a change period of 5 000 in five dimensions). All values are averaged over 30 repeats and the first 50 000 function evaluations (10 changes in the environment) are depicted. The average number of sub-populations used by DynPopDE is included. DynPopDE commenced with a single sub-population, but the number of sub-populations quickly increased to about eight after 5 000 function evaluations, and stabilised at about 14 after 50 000 function evaluations. The number of sub-populations was higher than the number of peaks in this environment. However, this is not necessarily a negative occurrence, as only the best sub-population is evolved at any given time through the competitive population evaluation process.

The initial small number of sub-populations enabled the offline error of DynPopDE to decrease noticeably faster than that of DynDE and CDE, at the commencement of the optimisation process. The difference between the offline errors of the three algorithms decreased as time passed. CDE yielded a lower final offline error than DynPopDE on this environment, so CDE's offline error eventually dropped to a value lower than that of DynPopDE.

DynPopDE perpetually creates and removes sub-populations. DynPopDE should consequently have more sub-populations that have not converged to an optimum than DynDE and CDE at any given time. These sub-populations caused DynPopDE to have higher current errors than DynDE and CDE, immediately after changes in the environment (compare

Figure 5.23 to Figure 4.34). DynPopDE's current errors dropped very quickly after changes and subsequently followed similar trends to those of DynDE and CDE.

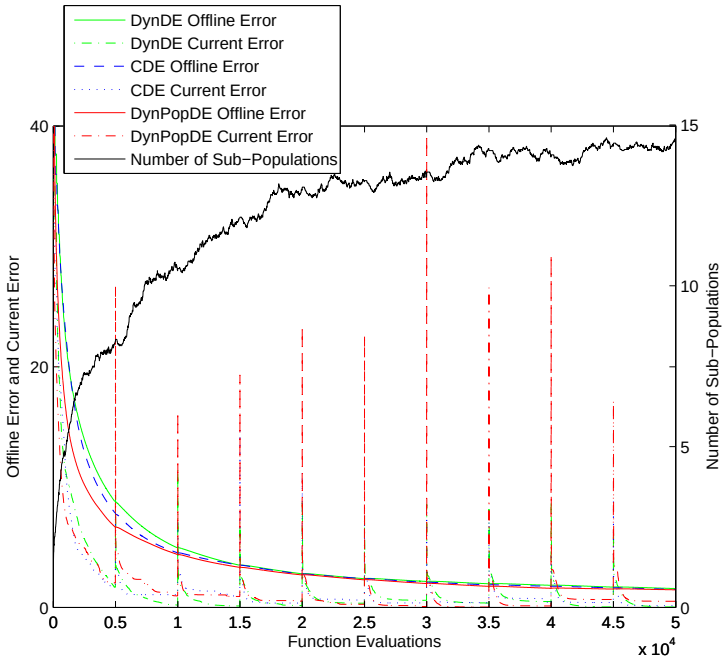

Figure 5.23: Offline errors and current errors of DynDE, CDE and DynPopDE on the MPB with the Scenario 2 settings.

Figure 5.24 gives the diversity and average sub-population diversity (calculated using equation (4.8) on page 156) of DynDE, CDE and DynPopDE for the same environment and period used in Figure 5.23. DynPopDE's diversity commences at a lower value than that of DynDE and CDE due the small number of initial sub-populations. However, the diversity value increases to a value very similar to that of DynDE and CDE after the first 1 000 function evaluations. The average sub-population diversity decreases faster than that of DynDE and CDE, as all initial function evaluations are allocated to a single sub-population. The fact that the average sub-population diversity of DynPopDE is very similar to that of CDE for most of the depicted period, shows the similarity in the behaviour of the two algorithms.

The previous sections found than DynPopDE is especially effective on problems with a large number of peaks when a high change period is used. Figure 5.25 gives the offline and current errors of DynDE, CDE and DynPopDE on the MPB, with the conical peak

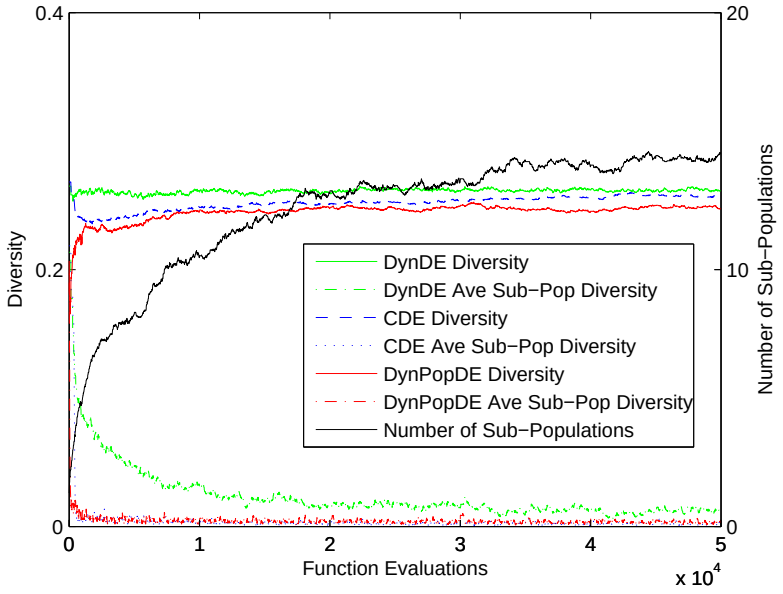

Figure 5.24: Diversity profiles of DynDE, CDE and DynPopDE on the MPB with the Scenario 2 settings.

function in five dimensions, with 100 peaks and a change period of 100 000. The average number of sub-populations used by DynPopDE is included. The first 300 000 function evaluations are shown.

The number of sub-populations used by DynPopDE was greater in Figure 5.25 than the number used in Figure 5.23. DynPopDE thus created more sub-populations in order to track more of the peaks. The benefit of creating more sub-populations is clear from the resulting offline error. The current errors of DynDE and CDE reached a plateau relatively quickly after a change in the environment, as the sub-populations converged to sub-optimal peaks. The current error of DynPopDE continued to decrease and tended towards zero as more sub-populations were created and new optima were consequently discovered. DynPopDE's offline error reached a value close to zero after 50 000 function evaluations. DynPopDE thus benefited greatly from having a large number of function evaluations available. This explains the trend, observed in previous sections, that DynPopDE was comparatively more effective than CDE and DynDE on problems with a large change period.

A high change period, however, does not guarantee a low offline error for DynPopDE.

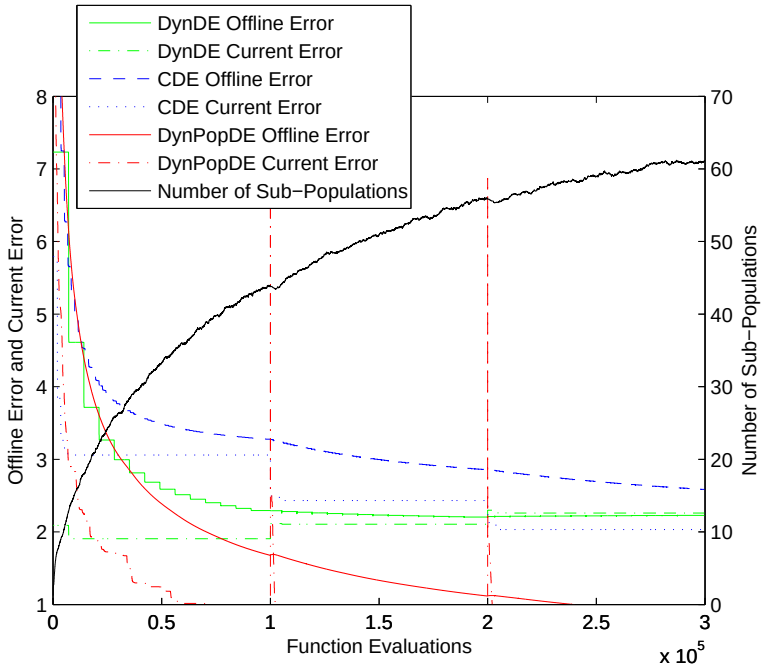

Figure 5.25: Offline errors and current errors of DynDE, CDE and DynPopDE on the MPB with the conical peak function in five dimensions with 100 peaks and a change period of 100 000

Figure 5.26 gives the offline and current errors of DynDE, CDE and DynPopDE over the first 300 000 function evaluations on the MPB with the conical peak function in 100 dimensions with 100 peaks and a change period of 100 000. The average number of sub-populations used by DynPopDE is included. The figure shows that the current errors of DynPopDE and CDE followed a very similar trend in high dimensions. DynPopDE's offline and current errors were initially slightly lower than those of CDE, but reached similar values after the second change in the environment. The number of sub-populations that was created by DynPopDE in 100 dimensions was considerably lower than that created in five dimensions (compare Figures 5.26 and 5.25). This suggests that the sub-population spawning process of DynPopDE is less effective in high dimensional environments.

The change period and number of dimensions were found to influence the number of sub-populations that DynPopDE creates. Figures 5.27 to 5.32 show how the number of sub-populations changes over time for various numbers of peaks being present in the environment. All figures give data collected over 30 repeats on the conical MPB function. Figures 5.27 and 5.28 give the number of sub-populations in five dimensional environments

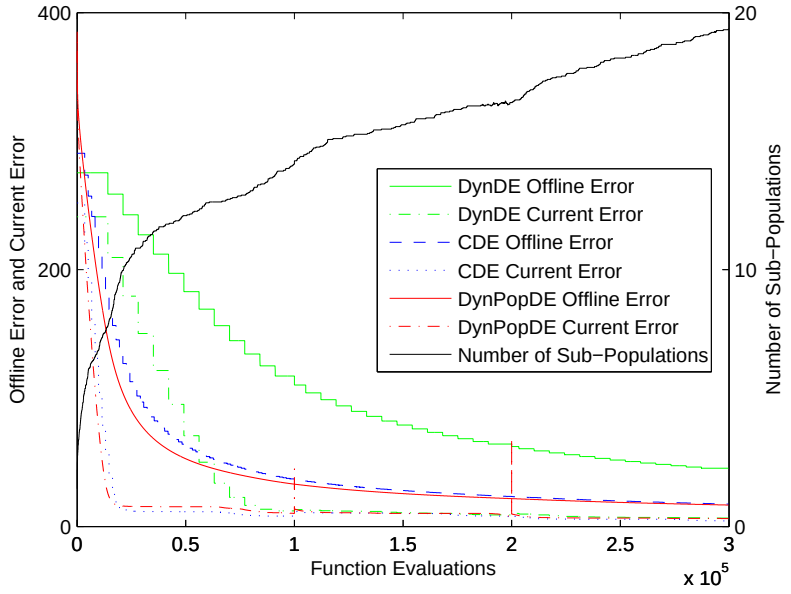

Figure 5.26: Offline errors and current errors of DynDE, CDE and DynPopDE on the MPB with the conical peak function in 100 dimensions with 100 peaks and a change period of 100 000

with a change period of 5 000 and 100 000, respectively. Figures 5.29 and 5.30 give the same information in 25 dimensions, while Figures 5.31 and 5.32 give the same information in 100 dimensions.

A considerable differentiation can be observed in the number of sub-populations created by DynPopDE for the various settings of number of peaks in five dimensions. A comparatively small number of sub-populations was created when a small number of peaks was present, but the number of sub-populations was typically greater than the number of peaks. A larger number of sub-populations was created when larger numbers of peaks were present in the environment, but the peaks typically outnumbered the sub-populations. This is consistent with the findings of research question 1, which showed that it is generally better to use a smaller number of sub-populations than the number of peaks.

DynPopDE created more sub-populations for five dimensional environments containing a large number of peaks when a change period of 100 000 was used than when a change period of 5 000 was used. A large change period enables the effective evolution of more sub-populations, making DynPopDE's choice of number of sub-populations sensible.

A distinctly different trend can be seen in Figure 5.29 as opposed to Figure 5.27. Dyn-

PopDE created a very similar number of sub-populations for the different settings of the number of peaks in the 25 dimensional case. This trend continues in Figure 5.31, which shows the 100 dimensional case. DynPopDE did not spawn sub-populations appropriately for the various environments, which results in DynPopDE's inferior performance in comparison to CDE in 100 dimensions.

The ineffective differentiation between the number of peak settings can be explained by considering the mechanism used by DynPopDE to detect stagnation. Equation (5.1) defines the function, $S(t)$, which must be true in order for DynPopDE to spawn a new sub-population. $S(t)$ is true when the change in fitness is equal to zero for all sub-populations. Ideally, this should occur when all sub-populations have converged to their respective optima. The high dimensional environments cause DynPopDE to converge slowly, which in turn causes $S(t)$ to be true infrequently. DynPopDE, consequently, creates too few sub-populations in high dimensional environments.

The problem is compounded by the sub-population removal mechanism, which is incorporated into the exclusion process. Exclusion occurs only when sub-populations converge to the same optima. However when the sub-populations converge slowly, the sub-populations infrequently converge to the same optima before changes occur in the environment. This means that unnecessary sub-populations are not removed, as can be seen in Figures 5.29 and 5.31, where more sub-populations were present than the number of peaks for the environments with low numbers of peaks.

DynPopDE's ineffective population spawning problem in high dimensions is alleviated by increasing the change period. Figures 5.30 and 5.32 give the number of sub-populations per number of peak setting in 25 and 100 dimensions, respectively, using a change period of 100 000. A comparison of these figures with Figures 5.29 and 5.31 shows that DynPopDE created a number of sub-populations proportional to the number of peaks that was present, when a high change period was used. The higher change period made it more likely that sub-populations would converge to optima, which assisted in detecting stagnation. The sub-population removal process was also improved by a high change period, as sub-populations converged to the same optima and were subsequently removed through the exclusion process. Note that DynPopDE was still less effective at creating different numbers of sub-populations for different numbers of peaks in 100 dimensions

than in 25. This explains why DynPopDE outperformed CDE only in environments with a high number of peaks and high change period in 100 dimensions.

DynPopDE can respond to changes in the environment when the number of optima fluctuates over time. The relationship between the number of peaks and number of sub-populations that was found in a typical run of the DynPopDE algorithm is illustrated in Figure 5.33. The environment used to produce this figure is the conical function in five dimensions. The maximum number of peaks was set to 50 and the percentage change in the number of peaks was set to 20%. Changes occurred once every 5 000 function evaluations. It is clear that the number of sub-populations increases when the number of peaks increases, but remains less than the number of peaks when the number of peaks is large. This is consistent with results found in the unknown number of peaks experiments which indicated that the number of sub-populations should be kept relatively small when many peaks are present.

The region between 100 000 and 130 000 function evaluations in Figure 5.33 represents a period during which the number of sub-populations outnumbered the number of peaks. In fact, for a brief period there were eight sub-populations present while there was only one peak in the environment. This is not necessarily a negative occurrence when the total number of sub-populations is small, since few sub-populations equate to more available function evaluations. Redundant sub-populations can assist in peak discovery and increase diversity, which is useful when new peaks are introduced. As long as the number of redundant sub-populations is small, the function evaluation overhead can be justified.

The existence of the three phases (refer to Sections 5.3.5 and 5.3.6) provides an insight into the comparative effectiveness of the DynPopDE algorithm. DynPopDE was effective at low change periods (phase one) because it commences with a single sub-population which resulted in more generations between changes in the environment.

DynPopDE was generally less effective than CDE during phase two. This phase was most evident in high dimensions on mid-range change periods. This corresponds to the environments in which, according to the discussion in the current section, it was found that DynPopDE's sub-population spawning and removal processes are not effective. DynPopDE's comparatively poor performance during phase two can thus be ascribed to the inability of the algorithm to respond effectively to the changes in numbers of peaks.

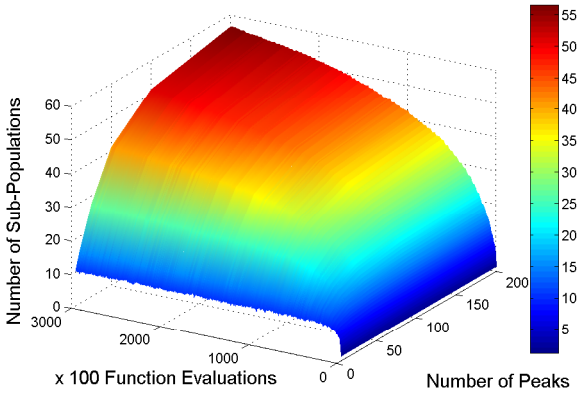

Figure 5.27: Number of sub-populations used by DynPopDE on the conical peak function in five dimensions with a change period of 5 000.

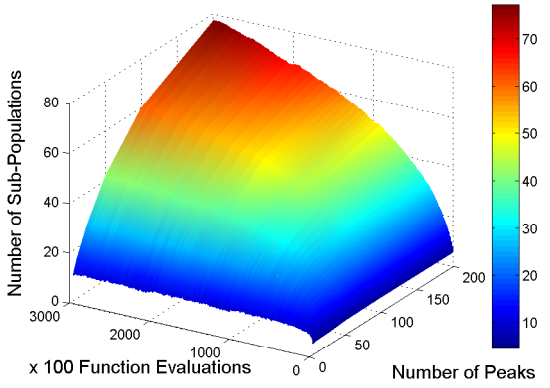

Figure 5.28: Number of sub-populations used by DynPopDE on the conical peak function in five dimensions with a change period of 100 000.

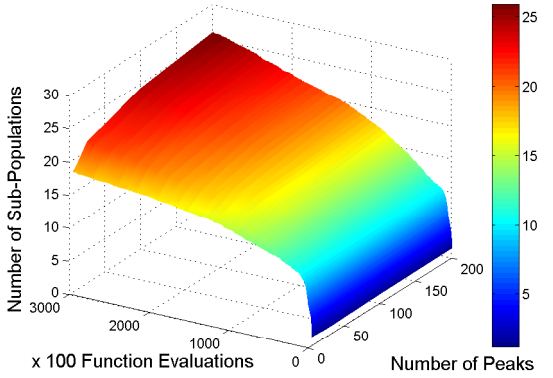

Figure 5.29: Number of sub-populations used by DynPopDE on the conical peak function in 25 dimensions with a change period of 5 000.

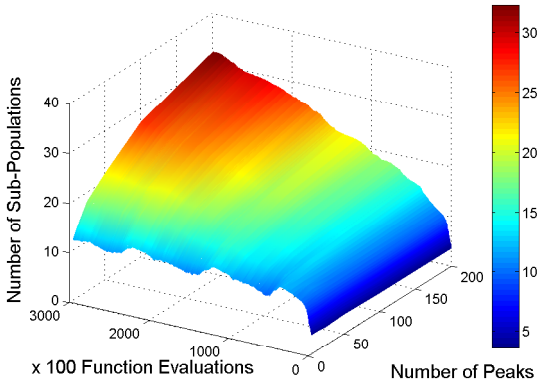

Figure 5.30: Number of sub-populations used by DynPopDE on the conical peak function in 25 dimensions with a change period of 100 000.

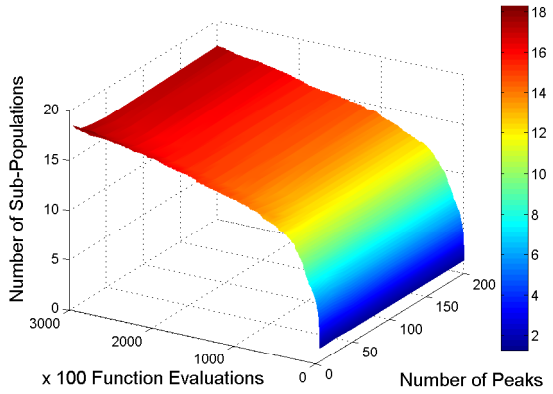

Figure 5.31: Number of sub-populations used by DynPopDE on the conical peak function in 100 dimensions with a change period of 5 000.

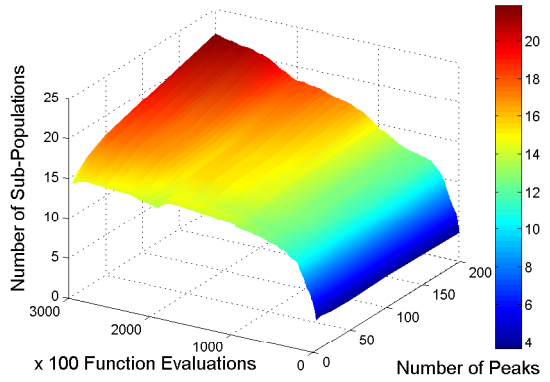

Figure 5.32: Number of sub-populations used by DynPopDE on the conical peak function in 100 dimensions with a change period of 100 000.

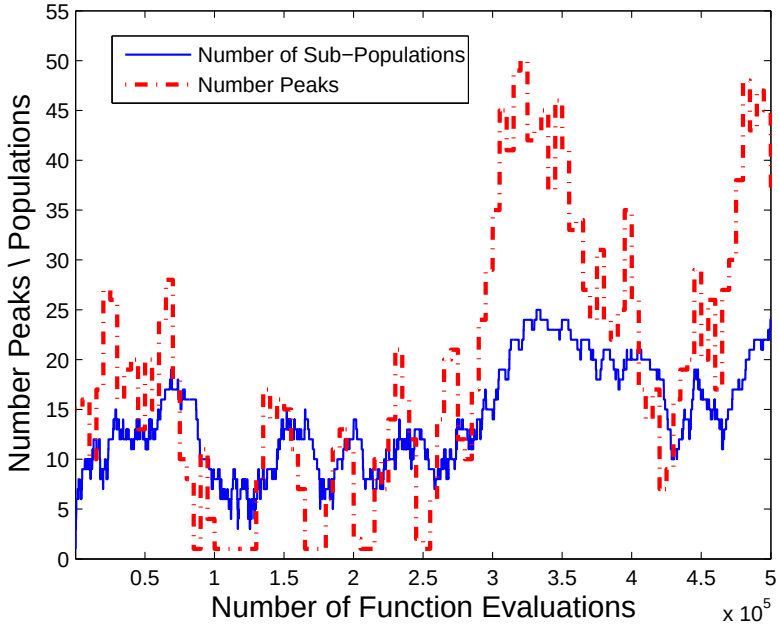

Figure 5.33: Comparison between the number of peaks and the number of populations in a typical execution of the DynPopDE algorithm on the conical peak function in five dimensions with a change period of 5 000, a maximum of 50 peaks and a 20% change in the number of peaks.

DynPopDE performed better than CDE during phase three, which occurred in low dimensions and at high change periods. This corresponds to environments in which DynPopDE effectively created sub-populations based on the number of peaks that were present. DynPopDE thus requires a sufficient number of function evaluations between changes in the environment to respond effectively to changes in the number of optima.

5.3.8 Research Question 7

Is the process of spawning and removing sub-populations as used in DynPopDE more effective than the process used in MPSO2?

This chapter introduced the sub-population spawning and removal processes that were incorporated into CDE to form DynPopDE. This research question investigates whether other spawning and removal processes, namely the approaches used in MPSO2, are a better choice for incorporation into CDE, and subsequently could yield better results.

The MPSO2 algorithm of Blackwell [2007] was discussed in Section 3.3.3.2. A dis-

advantage of MPSO2 is that it has a parameter, n_{excess}, that must be manually tuned. Blackwell [2007] did, however, find that for all values of $n_{excess} > 5$ the algorithm demonstrates identical behaviour to each other without significantly degrading performance. Despite the previously mentioned disadvantage associated with the MPSO2 algorithm, the reported results suggest that the swarm spawning and removing approach followed by Blackwell [2007] could potentially be more effective than the technique used in DynPopDE. This possibility was investigated by incorporating the approach to spawn and remove populations as used in MPSO2 into CDE. A sub-population is thus created when there are no available free sub-populations, i.e. all the current sub-populations had converged to within a diameter of $2r_{conv}$, where r_{conv} is calculated using the same equation as r_{excl} (equation (4.1)). The worst performing free sub-population is removed when the number of free sub-populations exceeds n_{excess}. The new algorithm is referred to as MPSO2CDE.

Experiments were conducted on the $n_p(t)$ standard set of experiments, using the same experimental procedure as for DynDE, CDE and DynPopDE. The parameter n_{excess} was set to $n_{excess} = 10$, as this value was found to be within the range where the algorithm exhibits identical behaviour [du Plessis and Engelbrecht, 2012a].

The performance analysis of MPSO2CDE compared to DynPopDE is given in Table 5.8. MPSO2CDE performed statistically significantly better than DynPopDE in only 64 of the 2 000 experiments, while it performed worse in 1 609 experiments. MPSO2CDE did, however, perform better more often than DynPopDE in experiments that used a change period of 100. The improvement over DynPopDE on these environments is minor as only 36 of the 250 experiments yielded statistically significantly different results.

The results of experiments that were conducted to answer this research question show that the population spawning and removal components of MPSO2, when incorporated into CDE, do not work as effectively as the spawning and removal components of DynPopDE.

5.3.9 Research Question 8

Is DynPopDE more effective than CDE and DynDE on the set of environments used in Chapter 4?

The experiments conducted in this chapter, up to this point, have all been on the MPB or the extended MPB, as these provide the functionality to vary or fluctuate the number of

Table 5.8: MPSO2CDE vs DynPopDE performance analysis on the $n_p(t)$ standard set

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
n_p 10	(10)	↑**3** ↓**0**	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑3 ↓69
25	(10)	↑**2** ↓**0**	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑2 ↓66
50	(10)	↑0 ↓1	↑0 ↓6	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
100	(10)	↑**1** ↓**0**	↑2 ↓6	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑3 ↓64
200	(10)	↑1 ↓1	↑1 ↓5	↑0 ↓7	↑0 ↓9	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑2 ↓60
pc 5	(10)	↑1 ↓1	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓71
10	(10)	↑0 ↓1	↑1 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓68
20	(10)	↑**1** ↓**0**	↑0 ↓6	↑0 ↓8	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓63
40	(10)	↑**3** ↓**0**	↑1 ↓6	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑4 ↓63
80	(10)	↑**2** ↓**0**	↑1 ↓3	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑3 ↓59
C	(25)	↑**2** ↓**0**	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑2 ↓173
S	(25)	↑**5** ↓**2**	↑3 ↓10	↑0 ↓17	↑0 ↓24	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑8 ↓151
All	(50)	↑**7** ↓**2**	↑3 ↓33	↑0 ↓42	↑0 ↓49	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑10 ↓324
Set.	**Max**	**10 Dimensions**								
n_p 10	(10)	↑0 ↓1	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓69
25	(10)	↑**1** ↓**0**	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓68
50	(10)	↑0 ↓2	↑0 ↓6	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
100	(10)	↑0 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
200	(10)	↑0 ↓0	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓66
pc 5	(10)	↑0 ↓1	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓70
10	(10)	↑0 ↓2	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓72
20	(10)	↑0 ↓1	↑0 ↓9	↑0 ↓9	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
40	(10)	↑0 ↓0	↑0 ↓5	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
80	(10)	↑**1** ↓**0**	↑0 ↓3	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓62
C	(25)	↑1 ↓2	↑0 ↓21	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑1 ↓173
S	(25)	↑0 ↓2	↑0 ↓15	↑0 ↓23	↑0 ↓24	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓164
All	(50)	↑1 ↓4	↑0 ↓36	↑0 ↓48	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑1 ↓337
Set.	**Max**	**25 Dimensions**								
n_p 10	(10)	↑**1** ↓**0**	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓70
25	(10)	↑**1** ↓**0**	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓70
50	(10)	↑1 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓68
100	(10)	↑0 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
200	(10)	↑0 ↓1	↑0 ↓4	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
pc 5	(10)	↑0 ↓3	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓73
10	(10)	↑0 ↓0	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
20	(10)	↑0 ↓0	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
40	(10)	↑**2** ↓**0**	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑2 ↓67
80	(10)	↑**1** ↓**0**	↑0 ↓6	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑1 ↓66
C	(25)	↑0 ↓0	↑0 ↓16	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓166
S	(25)	↑3 ↓3	↑0 ↓22	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑3 ↓175
All	(50)	↑3 ↓3	↑0 ↓38	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑3 ↓341
Set.	**Max**	**50 Dimensions**								
n_p 10	(10)	↑0 ↓0	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓67
25	(10)	↑0 ↓0	↑0 ↓6	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓66
50	(10)	↑0 ↓1	↑0 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
100	(10)	↑0 ↓0	↑0 ↓9	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
200	(10)	↑0 ↓1	↑0 ↓8	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
pc 5	(10)	↑0 ↓2	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓71
10	(10)	↑0 ↓0	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓69
20	(10)	↑0 ↓0	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓68
40	(10)	↑0 ↓0	↑0 ↓6	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓65
80	(10)	↑0 ↓0	↑0 ↓5	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓64
C	(25)	↑0 ↓0	↑0 ↓12	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓160
S	(25)	↑0 ↓2	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓177
All	(50)	↑0 ↓2	↑0 ↓37	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓337
Set.	**Max**	**100 Dimensions**								
n_p 10	(10)	↑**2** ↓**0**	↑1 ↓2	↑1 ↓5	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑4 ↓57
25	(10)	↑**2** ↓**0**	↑**5** ↓**2**	↑2 ↓5	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑9 ↓56
50	(10)	↑**3** ↓**0**	↑**5** ↓**4**	↑2 ↓4	↑0 ↓9	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑10 ↓56
100	(10)	↑**5** ↓**0**	↑**5** ↓**3**	↑4 ↓5	↑1 ↓7	↑0 ↓10	↑0 ↓10	↑0 ↓9	↑0 ↓10	↑15 ↓54
200	(10)	↑**2** ↓**0**	↑**5** ↓**3**	↑4 ↓4	↑1 ↓5	↑0 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓9	↑12 ↓47
pc 5	(10)	↑**1** ↓**0**	↑4 ↓4	↑2 ↓5	↑0 ↓8	↑0 ↓9	↑0 ↓8	↑0 ↓9	↑0 ↓9	↑7 ↓52
10	(10)	↑**5** ↓**0**	↑4 ↓4	↑0 ↓4	↑0 ↓8	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑9 ↓55
20	(10)	↑**2** ↓**0**	↑**4** ↓**3**	↑2 ↓5	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑8 ↓57
40	(10)	↑**1** ↓**0**	↑**4** ↓**2**	↑4 ↓5	↑0 ↓8	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑9 ↓55
80	(10)	↑**5** ↓**0**	↑**5** ↓**1**	↑**5** ↓**4**	↑2 ↓7	↑0 ↓9	↑0 ↓10	↑0 ↓10	↑0 ↓10	↑17 ↓51
C	(25)	↑0 ↓0	↑0 ↓14	↑0 ↓23	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓25	↑0 ↓162
S	(25)	↑**14** ↓**0**	↑**21** ↓**0**	↑**13** ↓**0**	↑2 ↓15	↑0 ↓22	↑0 ↓23	↑0 ↓24	↑0 ↓24	↑50 ↓108
All	(50)	↑**14** ↓**0**	↑**21** ↓**14**	↑13 ↓23	↑2 ↓40	↑0 ↓47	↑0 ↓48	↑0 ↓49	↑0 ↓49	↑50 ↓270
Set.	**Max**	**All Dimensions**								
n_p 10	(50)	↑**6** ↓**1**	↑1 ↓36	↑1 ↓45	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑8 ↓332
25	(50)	↑**6** ↓**0**	↑5 ↓33	↑2 ↓44	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑13 ↓326
50	(50)	↑4 ↓5	↑5 ↓30	↑2 ↓41	↑0 ↓49	↑0 ↓50	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑11 ↓324
100	(50)	↑**6** ↓**2**	↑7 ↓32	↑4 ↓42	↑1 ↓46	↑0 ↓50	↑0 ↓50	↑0 ↓49	↑0 ↓50	↑18 ↓321
200	(50)	↑3 ↓3	↑6 ↓27	↑4 ↓39	↑1 ↓44	↑0 ↓45	↑0 ↓49	↑0 ↓50	↑0 ↓49	↑14 ↓306
pc 5	(50)	↑2 ↓7	↑4 ↓42	↑2 ↓45	↑0 ↓48	↑0 ↓49	↑0 ↓48	↑0 ↓49	↑0 ↓49	↑8 ↓337
10	(50)	↑**5** ↓**3**	↑5 ↓39	↑0 ↓44	↑0 ↓48	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑10 ↓332
20	(50)	↑**3** ↓**1**	↑4 ↓33	↑2 ↓42	↑0 ↓48	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑9 ↓323
40	(50)	↑**6** ↓**0**	↑5 ↓26	↑4 ↓41	↑0 ↓48	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑15 ↓315
80	(50)	↑**9** ↓**0**	↑6 ↓18	↑5 ↓39	↑2 ↓46	↑0 ↓49	↑0 ↓50	↑0 ↓50	↑0 ↓50	↑22 ↓302
C	(125)	↑**3** ↓**2**	↑0 ↓86	↑0 ↓121	↑0 ↓125	↑0 ↓125	↑0 ↓125	↑0 ↓125	↑0 ↓125	↑3 ↓834
S	(125)	↑**22** ↓**9**	↑24 ↓72	↑13 ↓90	↑2 ↓113	↑0 ↓120	↑0 ↓123	↑0 ↓124	↑0 ↓124	↑61 ↓775
All	(250)	↑**25** ↓**11**	↑24 ↓158	↑13 ↓211	↑2 ↓238	↑0 ↓245	↑0 ↓248	↑0 ↓249	↑0 ↓249	↑64 ↓1609

peaks. This research question investigates the performance of DynPopDE in comparision to DynDE and CDE on the standard set used in Chapter 4, since the number of optima in the functions of that set is also not known to the algorithm.

The standard set of experiments which was defined in Section 4.6.2 contains functions from both the MPB and the GDBG. The MPB environments contained 10 peaks in all the experiments conducted in Chapter 4, but the GDBG functions had various numbers of optima. For example, F_{1b} has 50 optima, and F_3 to F_6 have a large number of optima (refer to Section 2.5.4). DynPopDE was executed on the standard set for all variations of settings listed in Table 4.3. This is the same set of 2 160 experiments used to evaluate the algorithms in the previous chapter.

A performance analysis was conducted to determine the number of experimental cases where DynPopDE performed statistically significantly better than DynDE and CDE. The analysis showed that DynPopDE performed better than DynDE in 867 of the 2 160 experiments while performing worse in 886 (the full performance analysis is given in Appendix C). DynPopDE thus performed better than DynDE in fewer than half of the environments.

The performance analysis comparison between DynPopDE and CDE is given in Tables 5.9 and 5.10. DynPopDE performed significantly better than CDE in 367 of the 2 160 experiments, while CDE performed better than DynPopDE in 1 257 cases. The tables show that the cases where DynPopDE performed better more often than CDE were concentrated in the MPB environments, with the exception of environments that used a low change period. DynPopDE performed better than CDE on functions F_2 and F_3 for a high change period in five dimensions, but DynPopDE was inferior to CDE in the vast majority of GDBG environments in high dimensions for high change periods. The cases where DynPopDE performed better than CDE on the MPB functions were mainly isolated to the spherical peak function in high dimensions.

The average percentage improvement of DynPopDE over CDE, over all experiments, was found to be -11.50%. The APIs per dimension were found to be -10.15%, -12.01%, -11.04%, -8.09%, -16.19% and -10.07% for 5, 10, 25, 50 and 100 dimensions respectively. The APIs per change period, were found to be 2.37%, 2.89%, -0.82%, -14.98%, -18.87%, -21.64%, -21.38% and -19.56% for change periods of 100, 500, 1 000, 5 000, 10 000, 25 000, 50 000 and 100 000 function evaluations respectively. DynPopDE was thus better than

Table 5.9: DynPopDE vs CDE performance analysis - Part 1

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**5 Dimensions**								
		MPB								
C_s 1	(2)	↑**2** ↓**0**	↑**1** ↓**0**	↑0 ↓0	*↑0 ↓1*	*↑0 ↓1*	↑**1** ↓**0**	*↑0 ↓1*	↑0 ↓0	↑**4** ↓**3**
5	(2)	↑0 ↓0	↑**2** ↓**0**	↑**1** ↓**0**	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓6*
10	(2)	↑0 ↓0	↑**1** ↓**0**	↑**2** ↓**0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓7*
20	(2)	*↑0 ↓1*	↑**1** ↓**0**	↑**2** ↓**0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓6*
40	(2)	*↑0 ↓2*	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑**6** ↓**2**
80	(2)	*↑0 ↓2*	*↑0 ↓1*	↑0 ↓0	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**2** ↓**0**	↑**7** ↓**3**
C	(6)	*↑1 ↓2*	↑**3** ↓**0**	↑**3** ↓**0**	↑2 ↓2	*↑1 ↓2*	↑0 ↓0	↑0 ↓0	↑**1** ↓**0**	↑**11** ↓**6**
S	(6)	*↑1 ↓3*	↑**3** ↓**1**	↑**3** ↓**0**	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓3*	*↑1 ↓4*	*↑1 ↓3*	*↑15 ↓21*
		GDBG								
F_{1a}	(6)	↑**4** ↓**0**	↑**6** ↓**0**	↑**4** ↓**0**	↑**3** ↓**2**	↑1 ↓1	↑0 ↓0	*↑0 ↓1*	↑0 ↓0	↑**18** ↓**4**
F_{1b}	(6)	↑**3** ↓**0**	↑**3** ↓**2**	↑3 ↓3	*↑0 ↓5*	*↑0 ↓6*	*↑1 ↓4*	↑2 ↓2	↑**2** ↓**1**	*↑14 ↓23*
F_2	(6)	↑**2** ↓**0**	↑**5** ↓**0**	↑**5** ↓**1**	↑**4** ↓**2**	↑**4** ↓**2**	↑3 ↓3	↑1 ↓1	↑**1** ↓**0**	↑**25** ↓**9**
F_3	(6)	↑0 ↓0	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓1*	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**5** ↓**0**	↑**20** ↓**11**
F_4	(6)	↑**1** ↓**0**	↑**2** ↓**1**	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑3 ↓33*
F_5	(6)	↑**2** ↓**0**	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑2 ↓42*
F_6	(6)	↑0 ↓0	*↑0 ↓2*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓38*
T_1	(7)	↑**4** ↓**0**	*↑1 ↓5*	*↑0 ↓6*	*↑0 ↓7*	*↑0 ↓5*	*↑1 ↓4*	*↑1 ↓3*	*↑1 ↓3*	*↑8 ↓33*
T_2	(7)	↑**1** ↓**0**	↑**4** ↓**2**	↑3 ↓3	*↑2 ↓4*	*↑3 ↓4*	*↑2 ↓4*	*↑1 ↓3*	*↑1 ↓3*	*↑17 ↓23*
T_3	(7)	↑**1** ↓**0**	↑**4** ↓**2**	↑3 ↓3	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓3*	*↑1 ↓3*	*↑1 ↓3*	*↑16 ↓21*
T_4	(7)	↑**5** ↓**0**	*↑2 ↓3*	*↑1 ↓5*	*↑0 ↓6*	*↑1 ↓6*	*↑1 ↓5*	*↑2 ↓5*	*↑2 ↓3*	*↑14 ↓33*
T_5	(7)	↑0 ↓0	↑**3** ↓**1**	↑3 ↓3	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑16 ↓23*
T_6	(7)	↑**1** ↓**0**	↑2 ↓2	*↑2 ↓5*	*↑1 ↓4*	*↑2 ↓4*	*↑1 ↓5*	*↑1 ↓4*	*↑1 ↓3*	*↑11 ↓27*
All	(54)	↑**14** ↓**5**	↑**22** ↓**16**	*↑18 ↓25*	*↑11 ↓32*	*↑13 ↓33*	*↑11 ↓28*	*↑9 ↓26*	*↑10 ↓22*	*↑108 ↓187*
Set.	**Max**	**10 Dimensions**								
		MPB								
C_s 1	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**10** ↓**0**
5	(2)	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**10** ↓**3**
10	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑5 ↓9*
20	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓2*	*↑0 ↓1*	*↑4 ↓8*
40	(2)	*↑0 ↓1*	↑**1** ↓**0**	↑**2** ↓**0**	↑0 ↓0	*↑0 ↓2*	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑3 ↓6*
80	(2)	*↑0 ↓1*	↑0 ↓0	↑0 ↓0	↑0 ↓0	↑**2** ↓**0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	*↑2 ↓4*
C	(6)	↑**2** ↓**0**	↑**4** ↓**0**	↑**3** ↓**0**	*↑0 ↓3*	*↑1 ↓4*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑10 ↓12*
S	(6)	↑**3** ↓**2**	↑**5** ↓**0**	↑**5** ↓**0**	↑**2** ↓**1**	↑3 ↓3	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓4*	↑**24** ↓**18**
		GDBG								
F_{1a}	(6)	↑**4** ↓**0**	↑**3** ↓**2**	↑3 ↓3	*↑1 ↓3*	*↑0 ↓3*	*↑0 ↓3*	*↑0 ↓1*	↑0 ↓0	*↑11 ↓15*
F_{1b}	(6)	↑**2** ↓**0**	*↑1 ↓2*	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓6*	*↑1 ↓5*	*↑0 ↓4*	↑2 ↓2	*↑7 ↓27*
F_2	(6)	↑**3** ↓**0**	↑**2** ↓**1**	↑**3** ↓**2**	*↑2 ↓3*	*↑2 ↓3*	*↑2 ↓3*	*↑2 ↓4*	*↑1 ↓4*	*↑17 ↓20*
F_3	(6)	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓3*	↑**2** ↓**1**	*↑2 ↓35*
F_4	(6)	↑**3** ↓**0**	↑**2** ↓**1**	↑**3** ↓**2**	*↑2 ↓3*	*↑2 ↓3*	*↑2 ↓4*	*↑2 ↓3*	*↑2 ↓4*	*↑18 ↓20*
F_5	(6)	↑**1** ↓**0**	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑1 ↓41*
F_6	(6)	↑0 ↓0	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓40*
T_1	(7)	↑**5** ↓**0**	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓5*	*↑1 ↓4*	*↑6 ↓44*
T_2	(7)	↑**3** ↓**1**	↑3 ↓3	↑3 ↓3	*↑0 ↓4*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑9 ↓30*
T_3	(7)	↑1 ↓1	*↑2 ↓3*	↑**4** ↓**3**	*↑2 ↓3*	*↑2 ↓4*	↑3 ↓3	*↑2 ↓3*	*↑1 ↓3*	*↑17 ↓23*
T_4	(7)	↑**3** ↓**1**	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑2 ↓4*	*↑5 ↓44*
T_5	(7)	*↑0 ↓1*	↑**2** ↓**1**	↑**3** ↓**2**	*↑3 ↓4*	*↑2 ↓4*	*↑2 ↓4*	*↑2 ↓3*	↑**2** ↓**2**	*↑16 ↓21*
T_6	(7)	↑**1** ↓**0**	↑1 ↓1	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓5*	*↑1 ↓5*	*↑3 ↓36*
All	(54)	↑**18** ↓**6**	*↑17 ↓20*	*↑18 ↓27*	*↑7 ↓36*	*↑8 ↓40*	*↑7 ↓38*	*↑6 ↓33*	*↑9 ↓28*	*↑90 ↓228*
Set.	**Max**	**25 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**10** ↓**0**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**1** ↓**0**	↑**9** ↓**2**
10	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑1 ↓1	↑**9** ↓**4**
20	(2)	↑**1** ↓**0**	↑0 ↓0	↑**2** ↓**0**	*↑0 ↓1*	*↑0 ↓1*	*↑0 ↓1*	↑1 ↓1	*↑0 ↓1*	*↑4 ↓5*
40	(2)	↑0 ↓0	↑0 ↓0	↑**1** ↓**0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑1 ↓9*
80	(2)	*↑0 ↓1*	↑**1** ↓**0**	↑**1** ↓**0**	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓2*	*↑0 ↓1*	*↑2 ↓10*
C	(6)	↑0 ↓0	↑**3** ↓**0**	↑**3** ↓**0**	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓3*	*↑0 ↓2*	*↑6 ↓19*
S	(6)	↑**3** ↓**1**	↑**4** ↓**0**	↑**6** ↓**0**	↑**3** ↓**2**	↑**3** ↓**2**	↑**3** ↓**2**	↑**4** ↓**2**	↑**3** ↓**2**	↑**29** ↓**11**
		GDBG								
F_{1a}	(6)	↑**3** ↓**0**	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓4*	*↑0 ↓3*	*↑0 ↓3*	*↑0 ↓3*	*↑3 ↓24*
F_{1b}	(6)	↑**3** ↓**0**	*↑1 ↓2*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓3*	*↑4 ↓31*
F_2	(6)	↑**6** ↓**0**	↑**2** ↓**0**	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	*↑8 ↓25*
F_3	(6)	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓45*
F_4	(6)	↑**5** ↓**0**	↑**1** ↓**0**	*↑0 ↓2*	*↑0 ↓5*	*↑1 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑1 ↓5*	*↑8 ↓27*
F_5	(6)	*↑0 ↓2*	*↑0 ↓4*	*↑0 ↓5*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓41*
F_6	(6)	↑0 ↓0	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓42*
T_1	(7)	↑**4** ↓**0**	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑4 ↓47*
T_2	(7)	↑**2** ↓**1**	*↑1 ↓3*	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓6*	*↑0 ↓6*	*↑0 ↓5*	*↑3 ↓40*
T_3	(7)	↑**2** ↓**1**	*↑0 ↓3*	*↑0 ↓3*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓5*	*↑0 ↓4*	*↑2 ↓31*
T_4	(7)	↑**4** ↓**2**	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑4 ↓49*
T_5	(7)	↑**3** ↓**2**	↑**3** ↓**2**	*↑0 ↓3*	*↑0 ↓5*	*↑1 ↓4*	*↑0 ↓4*	*↑0 ↓4*	*↑1 ↓3*	*↑8 ↓27*
T_6	(7)	↑**2** ↓**0**	*↑0 ↓1*	*↑0 ↓5*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑0 ↓7*	*↑2 ↓41*
All	(54)	↑**20** ↓**7**	*↑11 ↓19*	*↑9 ↓30*	*↑3 ↓44*	*↑4 ↓44*	*↑3 ↓43*	*↑4 ↓41*	*↑4 ↓37*	*↑58 ↓265*

Table 5.10: DynPopDE vs CDE performance analysis - Part 2

Cp		100	500	1000	5000	10000	25000	50000	100000	Total
Set.	**Max**	**50 Dimensions**								
		MPB								
C_s 1	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**1** ↓**0**	↑**11** ↓**0**
5	(2)	↑**1** ↓**0**	↑**2** ↓**0**	↑**2** ↓**0**	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**10** ↓**4**
10	(2)	↑0 ↓0	↑**2** ↓**0**	↑**2** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**1** ↓**0**	↑**9** ↓**4**
20	(2)	↑**1** ↓**0**	↑0 ↓0	↑**1** ↓**0**	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑**7** ↓**5**
40	(2)	↑0 ↓0	↑**1** ↓**0**	↑**1** ↓**0**	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*2* ↓*9*
80	(2)	↑*0* ↓*1*	↑**1** ↓**0**	↑0 ↓0	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*1* ↓*11*
C	(6)	↑0 ↓0	↑**3** ↓**0**	↑**3** ↓**0**	↑*1* ↓*4*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*4*	↑*7* ↓*23*
S	(6)	↑**3** ↓**1**	↑**5** ↓**0**	↑**5** ↓**0**	↑**4** ↓**1**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**2**	↑**4** ↓**2**	↑**33** ↓**10**
		GDBG								
F_{1a}	(6)	↑**3** ↓**0**	↑1 ↓1	↑*0* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*4*	↑*0* ↓*4*	↑*4* ↓*29*
F_{1b}	(6)	↑**4** ↓**0**	↑*1* ↓*2*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*4*	↑*5* ↓*34*
F_2	(6)	↑**6** ↓**0**	↑**2** ↓**1**	↑*0* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*8* ↓*34*
F_3	(6)	↑*0* ↓*2*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*43*
F_4	(6)	↑**6** ↓**0**	↑**1** ↓**0**	↑*0* ↓*2*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*7* ↓*32*
F_5	(6)	↑**1** ↓**0**	↑*0* ↓*1*	↑*0* ↓*3*	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*5*	↑*1* ↓*30*
F_6	(6)	↑0 ↓0	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*42*
T_1	(7)	↑**4** ↓**0**	↑*0* ↓*5*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*4* ↓*47*
T_2	(7)	↑**2** ↓**1**	↑*1* ↓*2*	↑*0* ↓*5*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*6*	↑*0* ↓*6*	↑*3* ↓*41*
T_3	(7)	↑**3** ↓**0**	↑*0* ↓*2*	↑*0* ↓*3*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*6*	↑*0* ↓*5*	↑*0* ↓*5*	↑*3* ↓*35*
T_4	(7)	↑**4** ↓**0**	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*4* ↓*45*
T_5	(7)	↑**4** ↓**1**	↑2 ↓2	↑*0* ↓*3*	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*6*	↑*6* ↓*38*
T_6	(7)	↑**3** ↓**0**	↑**2** ↓**1**	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*6*	↑*5* ↓*38*
All	(54)	↑**23** ↓**3**	↑*13* ↓*16*	↑*8* ↓*28*	↑*5* ↓*45*	↑*4* ↓*49*	↑*4* ↓*47*	↑*4* ↓*46*	↑*4* ↓*43*	↑*65* ↓*277*
Set.	**Max**	**100 Dimensions**								
		MPB								
C_s 1	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑*0* ↓*1*	↑**1** ↓**0**	↑*0* ↓*1*	↑*0* ↓*2*	↑**1** ↓**0**	↑*4* ↓*6*
5	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑1 ↓1	↑*0* ↓*1*	↑*0* ↓*2*	↑**1** ↓**0**	↑*0* ↓*1*	↑*4* ↓*7*
10	(2)	↑0 ↓0	↑1 ↓1	↑1 ↓1	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*2* ↓*11*
20	(2)	↑0 ↓0	↑*0* ↓*1*	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*12*
40	(2)	↑*0* ↓*1*	↑1 ↓1	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*1* ↓*13*
80	(2)	↑*0* ↓*1*	↑1 ↓1	↑1 ↓1	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*0* ↓*2*	↑*2* ↓*13*
C	(6)	↑0 ↓0	↑**5** ↓**0**	↑**4** ↓**0**	↑*1* ↓*3*	↑*0* ↓*4*	↑*0* ↓*5*	↑*0* ↓*5*	↑*0* ↓*4*	↑*10* ↓*21*
S	(6)	↑*0* ↓*2*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*5*	↑*0* ↓*6*	↑*1* ↓*5*	↑*1* ↓*5*	↑*3* ↓*41*
		GDBG								
F_{1a}	(6)	↑**1** ↓**0**	↑**1** ↓**0**	↑*0* ↓*3*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*2* ↓*33*
F_{1b}	(6)	↑**5** ↓**0**	↑0 ↓0	↑*0* ↓*4*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*5* ↓*34*
F_2	(6)	↑**6** ↓**0**	↑**2** ↓**0**	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*8* ↓*35*
F_3	(6)	↑*0* ↓*1*	↑*0* ↓*2*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*38*
F_4	(6)	↑**6** ↓**0**	↑**2** ↓**1**	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*8* ↓*36*
F_5	(6)	↑**2** ↓**1**	↑**3** ↓**0**	↑**3** ↓**0**	↑*1* ↓*3*	↑*0* ↓*4*	↑*0* ↓*4*	↑*0* ↓*4*	↑*0* ↓*4*	↑*9* ↓*20*
F_6	(6)	↑**1** ↓**0**	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*1* ↓*42*
T_1	(7)	↑**3** ↓**0**	↑*0* ↓*2*	↑*1* ↓*6*	↑*0* ↓*6*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*4* ↓*42*
T_2	(7)	↑**4** ↓**1**	↑*1* ↓*2*	↑*0* ↓*5*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*5* ↓*43*
T_3	(7)	↑**4** ↓**0**	↑**2** ↓**1**	↑*0* ↓*4*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*6* ↓*40*
T_4	(7)	↑**5** ↓**0**	↑**3** ↓**2**	↑*1* ↓*4*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*7*	↑*9* ↓*41*
T_5	(7)	↑**2** ↓**1**	↑**2** ↓**1**	↑*1* ↓*4*	↑*1* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*6* ↓*36*
T_6	(7)	↑**3** ↓**0**	↑*0* ↓*1*	↑*0* ↓*5*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*0* ↓*6*	↑*3* ↓*36*
All	(54)	↑**21** ↓**4**	↑*13* ↓*15*	↑*7* ↓*34*	↑*2* ↓*48*	↑*1* ↓*49*	↑*0* ↓*51*	↑*1* ↓*50*	↑*1* ↓*49*	↑*46* ↓*300*
Set.	**Max**	**All Dimensions**								
		MPB								
C_s 1	(10)	↑**6** ↓**0**	↑**8** ↓**1**	↑**6** ↓**1**	↑**4** ↓**2**	↑**4** ↓**1**	↑**4** ↓**1**	↑3 ↓3	↑**4** ↓**0**	↑**39** ↓**9**
5	(10)	↑**4** ↓**0**	↑**9** ↓**1**	↑**6** ↓**1**	↑**4** ↓**3**	↑*3* ↓*6*	↑*3* ↓*6*	↑**4** ↓**2**	↑3 ↓3	↑**36** ↓**22**
10	(10)	↑**1** ↓**0**	↑**8** ↓**1**	↑**9** ↓**1**	↑*2* ↓*6*	↑*2* ↓*8*	↑*2* ↓*7*	↑*2* ↓*6*	↑*2* ↓*6*	↑*28* ↓*35*
20	(10)	↑**2** ↓**1**	↑**3** ↓**1**	↑**7** ↓**1**	↑*1* ↓*7*	↑*1* ↓*7*	↑*1* ↓*6*	↑*2* ↓*7*	↑*1* ↓*6*	↑*18* ↓*36*
40	(10)	↑*0* ↓*4*	↑**4** ↓**1**	↑**5** ↓**1**	↑*2* ↓*5*	↑*2* ↓*8*	↑*0* ↓*7*	↑*0* ↓*7*	↑*0* ↓*6*	↑*13* ↓*39*
80	(10)	↑*0* ↓*6*	↑**3** ↓**2**	↑**2** ↓**1**	↑*2* ↓*6*	↑*3* ↓*6*	↑*1* ↓*7*	↑*1* ↓*7*	↑*2* ↓*6*	↑*14* ↓*41*
C	(30)	↑**3** ↓**2**	↑**18** ↓**0**	↑**16** ↓**0**	↑*4* ↓*16*	↑*2* ↓*20*	↑*0* ↓*17*	↑*0* ↓*15*	↑*1* ↓*11*	↑*44* ↓*81*
S	(30)	↑**10** ↓**9**	↑**17** ↓**7**	↑**19** ↓**6**	↑*11* ↓*13*	↑*13* ↓*16*	↑*11* ↓*17*	↑*12* ↓*17*	↑*11* ↓*16*	↑**104** ↓**101**
		GDBG								
F_{1a}	(30)	↑**15** ↓**0**	↑**11** ↓**5**	↑*7* ↓*13*	↑*4* ↓*22*	↑*1* ↓*20*	↑*0* ↓*17*	↑*0* ↓*15*	↑*0* ↓*13*	↑*38* ↓*105*
F_{1b}	(30)	↑**17** ↓**0**	↑*6* ↓*8*	↑*4* ↓*20*	↑*0* ↓*28*	↑*0* ↓*29*	↑*2* ↓*26*	↑*2* ↓*22*	↑*4* ↓*16*	↑*35* ↓*149*
F_2	(30)	↑**23** ↓**0**	↑**13** ↓**2**	↑*8* ↓*13*	↑*6* ↓*21*	↑*6* ↓*22*	↑*5* ↓*23*	↑*3* ↓*22*	↑*2* ↓*20*	↑*66* ↓*123*
F_3	(30)	↑*0* ↓*11*	↑*0* ↓*20*	↑*0* ↓*29*	↑*0* ↓*25*	↑*5* ↓*24*	↑*5* ↓*23*	↑*5* ↓*21*	↑*7* ↓*19*	↑*22* ↓*172*
F_4	(30)	↑**21** ↓**0**	↑**8** ↓**3**	↑*3* ↓*14*	↑*2* ↓*25*	↑*3* ↓*26*	↑*2* ↓*27*	↑*2* ↓*26*	↑*3* ↓*27*	↑*44* ↓*148*
F_5	(30)	↑**6** ↓**3**	↑*3* ↓*16*	↑*3* ↓*20*	↑*1* ↓*25*	↑*0* ↓*28*	↑*0* ↓*27*	↑*0* ↓*28*	↑*0* ↓*27*	↑*13* ↓*174*
F_6	(30)	↑**1** ↓**0**	↑*0* ↓*25*	↑*0* ↓*29*	↑*0* ↓*30*	↑*0* ↓*30*	↑*0* ↓*30*	↑*0* ↓*30*	↑*0* ↓*30*	↑*1* ↓*204*
T_1	(35)	↑**20** ↓**0**	↑*1* ↓*24*	↑*1* ↓*33*	↑*0* ↓*34*	↑*0* ↓*33*	↑*1* ↓*32*	↑*1* ↓*29*	↑*2* ↓*28*	↑*26* ↓*213*
T_2	(35)	↑**12** ↓**4**	↑*10* ↓*12*	↑*6* ↓*21*	↑*2* ↓*29*	↑*3* ↓*29*	↑*2* ↓*29*	↑*1* ↓*27*	↑*1* ↓*26*	↑*37* ↓*177*
T_3	(35)	↑**11** ↓**2**	↑*8* ↓*11*	↑*7* ↓*16*	↑*4* ↓*25*	↑*4* ↓*27*	↑*5* ↓*24*	↑*3* ↓*23*	↑*2* ↓*22*	↑*44* ↓*150*
T_4	(35)	↑**21** ↓**3**	↑*5* ↓*19*	↑*2* ↓*29*	↑*0* ↓*34*	↑*1* ↓*34*	↑*1* ↓*33*	↑*2* ↓*32*	↑*4* ↓*28*	↑*36* ↓*212*
T_5	(35)	↑**9** ↓**5**	↑**12** ↓**7**	↑*7* ↓*15*	↑*6* ↓*24*	↑*5* ↓*25*	↑*4* ↓*24*	↑*4* ↓*24*	↑*5* ↓*21*	↑*52* ↓*145*
T_6	(35)	↑**10** ↓**0**	↑*5* ↓*6*	↑*2* ↓*24*	↑*1* ↓*30*	↑*2* ↓*31*	↑*1* ↓*31*	↑*1* ↓*29*	↑*2* ↓*27*	↑*24* ↓*178*
All	(270)	↑**96** ↓**25**	↑*76* ↓*86*	↑*60* ↓*144*	↑*28* ↓*205*	↑*30* ↓*215*	↑*25* ↓*207*	↑*24* ↓*196*	↑*28* ↓*179*	↑*367* ↓*1257*

CDE only in very low change period environments, and the magnitudes of the improvements were small.

The results of this section indicate that the performance of DynPopDE is strongly dependent on the underlying function. DynPopDE performed better than CDE over a range of settings for the number of optima on the MPB (as proven in research questions 3 and 5). DynPopDE did, however, not scale well to other functions. This is a serious weakness of the algorithm, as the results in this section show that DynPopDE was generally outperformed by CDE on the majority of functions. CDE thus performed better than DynPopDE without fine-tuning CDE's parameter controlling the number of sub-populations for each function.

5.4 Comparison to Other Approaches

This section compares DynPopDE to the published results of other algorithms on variations of Scenario 2 of the MPB in terms of the number of peaks in the environments. Section 4.7.2 described four change detection strategies, that were incorporated into CDE, to ensure a fair comparison with results reported by other researchers. Two of these strategies, Det_{n_k-best} and $Det_{n_k-local}$, which only test for changes once every n_k generations, are not appropriate for incorporation into DynPopDE, since the number of sub-populations varies over time. A detection strategy that only tests for changes every n_k generations would thus be erratic, and would not ensure a uniform temporal sampling of the fitness landscape. DynPopDE was consequently tested using only the remaining two strategies, Det_{best} and Det_{local}.

Table 5.11 lists the offline errors and 95% confidence intervals of DynPopDE using the automatic change detection strategy, the Det_{best} detection strategy and the Det_{local} detection strategy. Several settings for the number of peaks were tested, but the other Scenario 2 settings (refer to Table 2.1) were left unchanged. A total of 500 000 function evaluations was performed for each of the 30 repeats on each of the environments.

The introduction of the detection strategies resulted in only minor changes in DynPopDE's average offline errors. The differences in offline error was statistically significant (according to a Mann-Whitney U test) in only one instance, which occurred when using

the Det_{local} detection strategy (printed in italics in Table 5.11).

Table 5.11: DynPopDE using various detection strategies

n_p	**Automatic**	Det_{best}	**p-val**	Det_{local}	**p-val**
1	0.53 ± 0.07	0.50 ± 0.06	0.562	0.52 ± 0.08	0.820
5	0.55 ± 0.04	0.65 ± 0.13	0.752	0.54 ± 0.04	0.719
10	0.82 ± 0.07	0.81 ± 0.07	0.775	0.76 ± 0.07	0.236
20	1.08 ± 0.06	1.13 ± 0.07	0.307	1.14 ± 0.07	0.286
30	1.31 ± 0.05	1.35 ± 0.07	0.676	1.38 ± 0.05	0.112
40	1.39 ± 0.05	1.44 ± 0.05	0.173	*1.48 ± 0.06*	0.031
50	1.58 ± 0.04	1.59 ± 0.08	0.654	1.56 ± 0.06	0.612
100	1.75 ± 0.06	1.82 ± 0.06	0.273	1.78 ± 0.05	0.719
200	1.83 ± 0.05	1.84 ± 0.05	0.959	1.85 ± 0.04	0.532

Table 5.12 lists the published results of 13 of the algorithms that were discussed in Section 3.3 on the variations of the MBP Scenario 2. The 95% confidence intervals were calculated from the reported standard errors or standard deviations in cases where the confidence interval was not reported. Each result was compared to the relevant DynPopDE result to determine which is better. Results were considered to be similar if the confidence intervals overlapped, i.e. neither algorithm was considered better than the other. Offline errors that were lower than the corresponding DynPopDE results are printed in italics. Offline errors that are higher than DynPopDE's (i.e. DynPopDE performed better) are printed in boldface in shaded cells.

Table 5.12: Reported offline errors on various numbers of peaks of various algorithms

n_p	**MMEO**	**HJEO**	**LSEO**	**CESO**	**ESCA**	**MPSO**	**CPSO**
1	**11.30 ± 6.98**	**7.08 ± 3.90**	**7.47 ± 3.88**	**1.04 ± 0.00**	**0.98 ± 0.00**	**5.07 ± 0.33**	*0.14 ± 0.03*
5	N/A	N/A	N/A	N/A	N/A	**1.81 ± 0.14**	0.72 ± 0.08
10	0.66 ± 0.39	*0.25 ± 0.20*	*0.25 ± 0.16*	**1.38 ± 0.04**	**1.54 ± 0.02**	**1.80 ± 0.12**	**1.06 ± 0.07**
20	0.90 ± 0.31	*0.39 ± 0.20*	*0.40 ± 0.22*	**1.72 ± 0.04**	**1.89 ± 0.08**	**2.42 ± 0.14**	**1.59 ± 0.06**
30	1.06 ± 0.27	*0.49 ± 0.18*	*0.49 ± 0.20*	*1.24 ± 0.02*	**1.52 ± 0.04**	**2.48 ± 0.14**	**1.58 ± 0.05**
40	1.18 ± 0.31	*0.56 ± 0.18*	*0.56 ± 0.18*	*1.30 ± 0.04*	**1.61 ± 0.04**	**2.55 ± 0.14**	1.51 ± 0.03
50	*1.23 ± 0.22*	*0.58 ± 0.18*	*0.59 ± 0.20*	*1.45 ± 0.02*	1.67 ± 0.04	**2.50 ± 0.12**	1.54 ± 0.03
100	*1.38 ± 0.18*	*0.66 ± 0.14*	*0.66 ± 0.14*	*1.28 ± 0.04*	*1.61 ± 0.06*	**2.36 ± 0.08**	*1.41 ± 0.02*
200	N/A	N/A	N/A	N/A	N/A	**2.26 ± 0.06**	*1.24 ± 0.02*
n_p	**HMSO**	**MSO**	**Cellular DE**	**Cellular PSO**	**SPSO**	**MPSO2**	**CDE**
1	**0.87 ± 0.05**	0.56 ± 0.04	**1.53 ± 0.07**	**5.23 ± 0.47**	N/A	N/A	**0.84 ± 0.11**
5	**1.18 ± 0.04**	**1.06 ± 0.06**	**1.50 ± 0.04**	**1.09 ± 0.22**	**1.98 ± 0.05**	N/A	0.55 ± 0.08
10	**1.42 ± 0.04**	**1.51 ± 0.04**	**1.64 ± 0.03**	**1.14 ± 0.13**	**1.98 ± 0.05**	**1.77 ± 0.05**	0.70 ± 0.11
20	**1.50 ± 0.06**	**1.89 ± 0.04**	**2.46 ± 0.05**	**2.20 ± 0.12**	N/A	N/A	**2.11 ± 0.19**
30	**1.65 ± 0.04**	**2.03 ± 0.06**	**2.62 ± 0.05**	**2.67 ± 0.13**	N/A	N/A	**2.74 ± 0.23**
40	**1.65 ± 0.05**	**2.04 ± 0.06**	**2.76 ± 0.05**	**2.70 ± 0.13**	N/A	N/A	**2.87 ± 0.29**
50	1.66 ± 0.02	**2.08 ± 0.02**	**2.75 ± 0.05**	**2.77 ± 0.13**	**3.47 ± 0.06**	N/A	**3.12 ± 0.20**
100	*1.68 ± 0.03*	**2.14 ± 0.02**	**2.73 ± 0.03**	**2.91 ± 0.14**	**3.60 ± 0.07**	N/A	**3.16 ± 0.24**
200	*1.71 ± 0.02*	**2.11 ± 0.03**	**2.61 ± 0.02**	**3.14 ± 0.12**	**3.47 ± 0.04**	**2.37 ± 0.03**	**3.42 ± 0.23**

The comparison of CDE to each of the tabulated algorithms is briefly discussed here.

MMEO [Moser and Hendtlass, 2007] MMEO is an algorithm based on extremal op-

timisation and searches for optima in parallel. The algorithm detects changes by re-evaluating all the best solutions in the search space and is thus comparable to the Det_{local} detection strategy. DynPopDE yielded a lower offline error than MMEO when a single peak was used. MMEO yielded a lower offline error than CDE on all other experiments. The confidence intervals of the two algorithms overlap for environments that used fewer than 50 peaks and MMEO, accordingly, cannot conclusively be considered superior to DynPopDE on these environments.

HJEO [Moser, 2007] HJEO is an extension of MMEO which incorporates a Hooke-Jeeves local search. HJEO performed better than DynPopDE on all reported cases, except when a single peak was present in the environment, in which case DynPopDE performed better than HJEO.

LSEO [Moser and Chiong, 2010] The local search component of MMEO was further improved in the LSEO algorithm. LSEO performed better than CDE on all reported cases, except when a single peak was present in the environment, in which case DynPopDE performed better than LSEO.

CESO [Lung and Dumitrescu, 2007] CESO uses a single DE population and a single PSO population to solve DOPs. Changes are detected by re-evaluating the best individual in the DE population and is thus comparable to the Det_{best} detection strategy. DynPopDE performed better than CESO in all environments that had fewer than 30 peaks. CESO performed better than DynPopDE when 30 or more peaks are present in the environment.

ESCA [Lung and Dumitrescu, 2010] ESCA is an adaptation of CESO which employs two DE populations in combination with the PSO population. DynPopDE performed better than ESCA in all environments that had less than 50 peaks. ESCA performed better than DynPopDE when 50 or more peaks are present in the environment.

MPSO [Blackwell and Branke, 2006] MPSO is a multi-swarm algorithm based on PSO. MPSO uses the Det_{local} detection strategy, and is consequently comparable to DynPopDE using the Det_{local} detection strategy. DynPopDE performed better than MPSO in all reported cases.

CPSO [Yang and Li, 2010] CPSO clusters particles of a PSO algorithm into sub-swarms to track optima in a dynamic environment. CPSO detects changes using the Det_{best} detection strategy and is thus roughly comparable to DynPopDE using the Det_{best} strategy. CPSO performed better than DynPopDE when a single peak was used, but overlapping confidence intervals were found when five peaks were present in the environment. DynPopDE performed better than CPSO when 10, 20 and 30 peaks were present. Overlapping confidence intervals were found when using 40 and 50 peaks. CPSO performed better than DynPopDE when 100 and 200 peaks were present in the environment.

HMSO [Kamosi *et al.*, 2010a] HMSO is an extension of MPSO which hibernates unproductive sub-swarms. HMSO uses the Det_{best} change detection strategy, and can be compared to DynPopDE using Det_{best}. DynPopDE performed better than HMSO in all experiments with fewer than 50 peaks. Overlapping confidence intervals were found when using 50 peaks, while HMSO performed better than DynPopDE in environments with 100 and 200 peaks.

MSO [Kamosi *et al.*, 2010b] MSO is a PSO-based algorithm that utilises a dynamic number of swarms to locate optima in a dynamic environment. Kamosi *et al.* [2010b] do not specify the change detection strategy that is used in MSO, but it is assumed that the Det_{best} strategy that was used in HMSO (which was developed by the same authors) is also used in MSO. DynPopDE performed better than MSO in all reported cases.

Cellular DE [Noroozi *et al.*, 2011] Cellular DE creates sub-populations by dividing the search space into equally sized cells. Cellular DE employs the Det_{local} change detection strategy and is thus compared to DynPopDE using the same strategy. DynPopDE performed better than Cellular DE in all reported cases.

Cellular PSO [Hashemi and Meybodi, 2009a] Cellular PSO creates sub-swarms by dividing the search space into equally sized cells. Cellular PSO uses the Det_{local} change detecting strategy and is thus roughly comparable to DynPopDE using the Det_{local} strategy. DynPopDE performed better than Cellular PSO in all cases.

SPSO [Li *et al.*, 2006] SPSO is a multi-swarm PSO based algorithm designed specifically for situations where the number of peaks is unknown. SPSO detects changes by re-evaluating the five best particles, and is thus comparable to DynPopDE using the Det_{local} strategy. DynPopDE performed better than SPSO in all reported cases.

MPSO2 [Blackwell, 2007] MPSO2 is a multi-swarm PSO based algorithm which self-adapts the number of swarms. MPSO2 detects changes using the Det_{local} strategy. DynPopDE performed better than MPSO2 in both the reported environments.

CDE The Det_{best} detection strategy was used in CDE to determine its performance on the environments used in this section. DynPopDE performed better than CDE when a single peak was used, and when more then 10 peaks were present in the environment. Overlapping confidence intervals were found when using 5 and 10 peaks.

This section compared DynPopDE results with the reported results of other algorithms. DynPopDE performed better than each of the discussed algorithms on at least one of the experimental environments. DynPopDE yielded better results on all environments than several of the algorithms (MPSO, Cellular DE, Cellular PSO, SPSO and MPSO2). The results presented in this section indicate that CDE compares favourably with the state-of-the-art algorithms in the literature.

5.5 Conclusions

This chapter introduced a new algorithm, named DynPopDE, which is aimed at problems in which the number of optima is unknown or fluctuating. DynPopDE spawns and removes sub-populations in order to find an effective number of sub-populations. A sub-population is created when the algorithm detects that all existing sub-populations have stagnated. The sub-population removal process is incorporated into the exclusion component, which removes a sub-population when it strays too close to another sub-population. DynPopDE has the advantage that it removes the need to tune the parameter controlling the number of sub-populations.

The performances of DynDE, CDE and DynPopDE were investigated on problems where the number of optima are unknown or unknown and fluctuating over time. The

extended MPB was used to simulate environments in which the number of optima fluctuate over time.

The first major result that was found in this chapter is that it is typically not advisable to use the same number of populations as the number of optima in DynDE and CDE when the number of populations is large. Better results were obtained when a relatively small number of populations was used. Knowing the number of optima in an environment in advance is thus generally not beneficial.

Research questions 2 and 3 investigated environments with various numbers of optima. CDE performed significantly better than DynDE over a wide range of experiments when the number of optima were unknown. DynPopDE generally performed better than DynDE and CDE on these environments, but was typically less effective than CDE in high dimensional environments. DynPopDE creates more sub-populations when a large number of optima is present than when a small number of optima is present. Experimental results showed that high dimensions and low change periods diminish DynPopDE's ability to distinguish effectively between environments with different numbers of optima.

Research questions 4 and 5 focused on environments in which the number of optima fluctuate over time. Fluctuating the number of optima during the course of the optimisation process had a negative effect on the results of DynDE and CDE. CDE did, however, still perform significantly better than DynDE. The experimental results showed that DynPopDE performed better than CDE when the number of optima fluctuates. Three distinct phases, with respect to change period, were identified in DynPopDE's comparative performance. DynPopDE performed better than CDE in phases one and three, which occur at very low and very high change periods. DynPopDE performed similarly or worse than CDE during phase two, which occurs at intermediate change periods. The range of change periods representing phase two was found to increase as the number of dimensions increases.

The spawning and removal processes used in DynPopDE were compared to the approaches used in MPSO2. The algorithm that used MPSO2's sub-population creation and removal components performed significantly worse than DynPopDE over a wide range of fluctuating number of peak experiments. The processes used in MPSO2 are thus not appropriate for use in conjunction with CDE.

DynPopDE was found to compare favourably to the reported results of several state-of-the-art algorithms on various settings of number of peaks on the MPB.

The comparison of DynPopDE and CDE on the standard set of experiments used in Chapter 4 showed that DynPopDE does not scale well to other functions. Despite the fact that DynPopDE performed well on a wide range of numbers of optima on the MPB functions, DynPopDE's poor performance on the GDBG functions leads to the conclusion that DynPopDE is not a generally applicable algorithm.

The following chapter investigates the incorporation of self-adaptive control parameters into CDE and DynPopDE.