Using Competitive Population Evaluation in a Differential Evolution algorithm for Dynamic Environments

Mathys C. du Plessis *, Andries P. Engelbrecht,
Department of Computer Science, University of Pretoria, Pretoria, South Africa

Abstract

This paper reports three adaptations to DynDE, an approach to using Differential Evolution to solve dynamic optimization problems. The first approach, Competitive Population Evaluation (CPE), is a multi-population DE strategy aimed at locating optima faster in the dynamic environment. This approach is based on allowing populations to compete for function evaluations based on their performance. The second approach, Reinitialization Midpoint Check (RMC), is aimed at improving the technique used by DynDE to maintain populations on different peaks in the search space. A third approach, consisting of a combination of CPE and RMC is investigated. The new strategies are empirically compared to DynDE using various problem sets. The empirical results show that the new approaches constitute an improvement over DynDE and other approaches in the literature.

Key words: Global optimization, Evolutionary computations, Dynamic Environments, Competing Populations, Moving Peaks

1. Introduction

Dynamic optimization problems are found in many real world domains, for example air traffic control, polarization mode dispersion compensation in optical fibre, and target tracking in military applications. Despite the fact that evolutionary algorithms often successfully solve static problems, dynamic optimization problems tend to pose a challenge to evolutionary algorithms (Morrison, 2004). Lack of diversity is the main drawback of most of the standard evolutionary algorithms when it comes to dynamic problems, since the algorithms tend to converge on a single optimum in the solution space and then lack the diversity to locate new optima when they appear. Differential evolution (DE) is one of the evolutionary algorithms that do not scale well to dynamic environments.

Mendes and Mohais (2005) applied DE to dynamic optimization problems by utilizing multiple populations maintained on distinct peaks in the solution space. In order to better track the movements of the peaks in the solution space, the diversity of a subset of each population was increased.

In this paper, extensions to the above approach are suggested. The first adaptation is a novel approach called Competitive Population Evaluation (CPE). CPE allows populations to compete for fitness evaluations, hence allocating fitness evaluations to the more successful populations first. The second suggested adaptation is Reinitialization Midpoint Check (RMC). RMC is an improvement on the technique used in DynDE to prevent populations converging to the same peak.

The rest of the paper is structured as follows: related work is discussed in Section 2. The DE algorithm is discussed

* Corresponding author.
Email addresses: mc.duplessis@nmmu.ac.za (Mathys C. du Plessis), engel@cs.up.ac.za (Andries P. Engelbrecht).

in Section 3. Section 4 reviews the research of Mendes and Mohais (2005). The benchmark which is used in this work is described in section 5. CPE is presented in section 6, RMC in section 7 and the combination of CPE and RMC is described in section 8. The results of a comparative investigation into the scalability of each of these approaches is given in section 9. The new approaches are compared with other research in section 10, and conclusions are drawn in section 11.

2. Related Work

Several of the earlier investigations into optimization in dynamic environments involved approaches based on Genetic Algorithms. More recently, attention has been given to other population-based optimization algorithms like Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) and Differential Evolution (DE) (Storn, 1996). A survey of the literature identified three main strategies of which at least one is present in most algorithms aimed at dynamic optimization. These strategies are:

- **Increase Diversity:** One of the main reasons why traditional population-based algorithms fail in dynamic environments is that the algorithms converge to a solution and then lack the diversity required to locate new optima once the environment has changed. Jin and Branke (2005) pointed out that most algorithms try to remedy this problem by either explicitly increasing diversity after a change in the environment is detected, or by maintaining more diversity during the entire run.
- **Memory:** Optima in the dynamic environments shift or disappear when changes in the environment occur. However, unless the changes are large, information on where optima were located before the change can still be used to locate new optima in the vicinity of old optima. Many algorithms make use of either explicit memory to store information about the location of optima, or implicit memory, mostly in the form of sub-populations that converge to a variety of optima. Since a change in the environment can lead to the global optimum being reduced to a mere local optimum and vice versa, knowing the location of all, or at least several, of the local optima can accelerate the location of the new global optimum.
- **Parallel Search:** Many algorithms employ a parallel search to locate the optima in the environment, mostly by using multiple independent populations. Several approaches take steps to ensure that sub-populations do not converge to the the same optimum and they also control how individuals are distributed among the populations.

Jin and Branke (2005) provide a survey on algorithms for dynamic optimization. These algorithms and some of the more recent advances in the field will now be discussed in terms of the strategies described above.

Cobb (1990) suggested drastically increasing the mutation rate of a GA after a change in the environment has occurred, while Vavak et al. (1997) advocated a more gradual increase. Hu and Eberhart (2002) suggested that particles in a PSO algorithm should be reinitialized when a change in the environment occurs. Approaches aimed at maintaining a high amount of diversity during the entire run include Grefenstette's Random Immigrants (Grefenstette, 1999) and refinements made by Yang (2005). These introduce random individuals into the GA's population after each generation. In contrast, Morrison (2004) made use of stationary individuals (called sentinels) that are uniformly distributed around the search space to increase diversity.

The thermodynamic GA (Mori et al., 1997), (Mori et al., 1996), (Mori et al., 1998) explicitly controls the population's diversity throughout the run by selecting individuals for the next population, not only based on their fitness, but also based on the rarity of their genes.

More recent diversity-increasing approaches include charged particles (Blackwell and Bentley, 2002), where each particle is assigned a virtual charge and then allowed to repel each other based on the laws of electrostatics. The idea of increasing diversity by reinitializing a number of individuals in a population within a hyper-sphere centered around the best individual within the population was proposed by Blackwell and Branke (2004, 2006). These individuals are called Quantum individuals and were implemented in a PSO algorithm. A similar approach, called Brownian individuals, involves the creation of individuals close to the best individual by adding a small random value sampled from a normal distribution to each component of the best individual (Mendes and Mohais, 2005). Investigations into finding an appropriate neighborhood structure for PSO (Janson and Middendorf, 2004), (Li and Dam, 2003) and an appropriate scheme for DE (Mendes and Mohais, 2005) have been conducted, since these parameters greatly affect the diversity of the population.

In some dynamic problems the shape of the solution space is either oscillating or cyclic, i.e. changes in the dynamic

environment will result in the environment returning to the same configuration at a future stage. For these problems, maintaining an explicit memory of good solutions has been found to be especially useful. Ramsey and Grefenstette (1993) made use of a knowledge base of strategies (individuals that had performed well in previous generations) that are inserted in the population when a change occurs that results in a previously seen environment.

In situations where the dynamic environment is not expected to periodically return to the same configuration, memory is mostly employed to retain information regarding the location of optima in the environments before a change has occurred. This is generally achieved by using multiple independent populations to locate various optima. A key feature of these approaches is that independent populations are allowed to search for optima in parallel. Three of the seminal algorithms in this class are Branke's Self-Organizing Scouts (SOS) (Branke et al., 2000), (Branke, 2002), (Branke and Schmeck, 2003), Oppacher and Wineberg's Shifting Balance GA (SBGA) (Oppacher and Wineberg, 1999) and Ursem's Multinational GA (MGA) (Ursem, 2000). All three of these approaches made use of some strategy to intelligently distribute individuals among the populations.

SOS makes use of a large base population that identifies optima in the search space. When an optimum is located, a small scout population is left to guard and further optimize the optimum. Individuals are distributed between the various scout populations and the base population by the algorithm. This distribution is based firstly on the fitness of the best individual in each population, and secondly, on the amount of improvement in fitness made between the previous and current generation.

While SOS aims to keep the bulk of the population in a single population searching for new optima, SBGA groups a single core around the best optimum that was found and uses smaller populations, called colonies, to search for new optima. The information contained in the colony populations is shared with the core population by means of migrant individuals that are periodically transferred from colony populations to the core population.

In contrast to SOS and SBGA, the MGA algorithm does not explicitly control the number of individuals in sub-populations. Furthermore, parent individuals for the creation of offspring for a given population are not only selected from the current population, but from all individuals. MGA uses *hill-valley detection* to form populations. This technique randomly samples points between two individuals to determine whether the two individuals are located on the same optimum. When offspring are created, hill-valley detection determines if the new individuals are to remain in the current population, whether the new individual should join another population or whether the new individual should be placed in an entirely new population.

Considerable success has been achieved in applying modern optimization algorithms, like PSO and DE, to dynamic optimization.

Parrott and Li (2004) suggested a multiple-population PSO approach to optimizing dynamic problems, call speciation. The social component of PSO provides a simple method to divide the population in sub-populations. In this algorithm, a particle is classed into a population if the Euclidean distance between the position of the particle and the best particle in the population is within a certain threshold value. The *global best* value of each particle within a population is set to the *personal best* value of the best particle. Particles can thus migrate to another population by moving too far away from the current population's best particle or by moving closer to another population's best particle.

Blackwell and Branke (2006) introduced a multiple-population PSO-based algorithm that is based on three components: Exclusion, Anti-convergence and Quantum individuals. An interesting novelty about their approach is that all populations contain the same number of individuals. The aim of having multiple populations is that each population should be positioned on its own, promising optimum in the environment. Unfortunately, populations often converge to the same peak, hence decreasing diversity. Exclusion (Blackwell and Branke, 2004) is a technique meant to prevent swarms from clustering around the same peak by means of reinitializing populations that stray within a threshold Euclidean distance from a better performing population. This threshold distance is called the exclusion radius. Anti-convergence is meant to prevent stagnation of the particles in the search space. Consequently, if it is found that all populations have converged to their respective optima, the weakest population is randomly reinitialized. Convergence is detected if all particles within a swarm fall within a threshold Euclidean distance of each other. This is called the convergence radius.

Li et al. (2006) improved the speciation algorithm by introducing ideas from (Blackwell and Branke, 2004), namely quantum individuals to increase diversity and anti-convergence to detect stagnation and subsequently to reinitialize the worst-performing populations.

Mendes and Mohais (2005) adapted the ideas from (Blackwell and Branke, 2006) to a DE algorithm for dynamic

optimization. Their multi-population algorithm, DynDE, uses Brownian individuals to increase diversity, and exclusion to prevent populations from converging to the same peak. DynDE will be discussed in detail in later sections.

Recently, Brest et al. (2006, 2009) proposed a self-adaptive multi-population DE algorithm (*jDE*) for optimizing dynamic environments. This work focused on adapting the DE scale factor and crossover probability, but it also contained several components that are similar to other dynamic optimization algorithms. An idea similar to exclusion is used to prevent populations converging to the same optimum. An ageing metaphor is used to reinitialize populations that have stagnated on a local optimum. Each individual's age is incremented every generation. Offspring inherit the age of parents, but this age may be reduced if the offspring performs significantly better than the parent. Populations of which the best individual is too old are reinitialized. Within populations a further mechanism is used to prevent convergence. Individuals are reinitialized if the Euclidean distance between the individual and the best individual in the population is too small. The algorithm also utilizes a form of memory called an archive. The best individual is added to the archive every time a change in the environment occurs. A random individual is selected from the archive from which one of the sub-populations is generated by adding small random numbers to each of the individual's components.

The most successful non-population based algorithm is Moser and Hendtlass's Multi-Phase Multi-Individual Extremal Optimization (MMEO) algorithm (Moser and Hendtlass, 2007). Extremal Optimization (EO) (Boettcher and Percus, 1999) makes use of a single solution which is mutated, and consequently finds the optimum of the search space through hill climbing. EO was adapted to contain several individuals, each of which uses five steps to find optima. Firstly, a stepwise sampling of the solution space is performed to locate areas that potentially contain optima. From these potential points, an individual uses hill climbing to find a local optimum. During the hill climbing phase the individuals are checked to ensure that no duplicates (individuals that are optimizing the same peak) exist. If changes in the environment occur, individuals are optimized further by using hill climbing. Finally, individuals are fine tuned using finer grained hill climbing.

3. Differential Evolution

Differential Evolution (DE) is a relatively new optimization algorithm based on Darwinian evolution, created by Storn and Price (1997). In DE, mutations are made based on the spatial difference between two or more individuals added to a target vector, as opposed to other evolutionary algorithms where random mutations are generally made to individuals in the population. Several variants to the DE algorithm have been suggested, but a generic algorithm is as follows (K. Price, 2005):

(i) Randomly create I individuals to form a population.
(ii) Evaluate each individual.
(iii) Create I individuals for a trial population as follows:
  (a) Select three individuals at random, $\vec{x}_1 \neq \vec{x}_2 \neq \vec{x}_3$, from the current population.
  (b) Create a new trial vector $\vec{v}$ using:

$$\vec{v}_i = \vec{x}_1 + \mathcal{F} \cdot (\vec{x}_2 - \vec{x}_3) \tag{1}$$

  where $\mathcal{F} \in (0, \infty)$ is known as the scale factor.
  (c) Add $\vec{v}_i$ to the trial population.
(iv) For each individual $\vec{x}_i$ in the current population select the corresponding $\vec{v}_i$ in the trial population. With these two individuals, do the following:
  (a) Create offspring $\vec{u}_i$ as follows:

$$u_{i,j} = \begin{cases} v_{i,j} \text{ if} (U(0,1) \leq C_r \text{ or } j = j_{rand}) \\ x_{i,j} \text{ otherwise} \end{cases} \tag{2}$$

  where $C_r \in (0, 1)$ is the crossover probability and j_{rand} is a randomly selected index.
  (b) Evaluate the fitness of $\vec{u}_i$.
  (c) If $\vec{u}_i$ has a better fitness value than $\vec{x}_i$ then replace $\vec{x}_i$ with $\vec{u}_i$.
(v) Repeat steps 3 and 4 until a termination criterion is met.

Most variations of DE (called schemes) are based on different approaches to creating each of the temporary individuals, $\vec{v}_i$ (Storn, 1996) (see equation (1)), and how offspring are created (see equation (2)). By convention, schemes are labelled in the form DE/*a*/*b*/*c*, where *a* is the method used to select the target vector, *b* is the number of difference vectors and *c* is the method used to create offspring. The scheme used in the above algorithm is referred to as DE/rand/1/bin.

4. DynDE

DynDE is a differential evolution algorithm developed by Mendes and Mohais (2005) to solve dynamic optimization problems. The most successful versions of DynDE make use of multiple populations, exclusion and Brownian individuals to adapt DE to dynamic environments.

4.1. *Multiple populations*

Typically, a static problem space may contain several peaks or local optima. These peaks not only move around in a dynamic environment, but also change in height. This implies that it is necessary to track the movements of the not only best peak found in the problem space, since an entirely different peak may become the optimal peak once a change in the environment occurs. It is thus desirable to keep track of all the peaks. An effective method to achieve this is to maintain several independent populations of DE individuals, one on each peak. In most experiments, Mendes and Mohais used 10 populations, each containing 6 to 10 individuals.

4.2. *Exclusion*

In order to track all the peaks, it is necessary to ensure that all populations converge to different peaks. If all populations converged to the optimum peak it would defeat the purpose of having multiple populations. Mendes and Mohais used exclusion (Blackwell and Branke, 2004) to prevent populations from converging to the same peak. The approach works by comparing the best individuals from each population. If the spatial difference between any two of these individuals becomes too small, their errors are compared and the entire population of the inferior individual is randomly reinitialized. A threshold is used to determine if two individuals are too close. This is calculated as follows:

$$r_{excl} = \frac{X}{2p^{\frac{1}{d}}} \tag{3}$$

where X is the range of the d dimensions (assuming equal ranges for all dimensions), and p is the number of peaks. By studying equation (3) it can be seen that the exclusion threshold increases with the increase in the number of dimensions and decreases if the number of peaks is increased.

4.3. *Brownian individuals*

Since a change in the environment implies at least some movement of some of the peaks, it is unlikely (even if the change is small) that all of the populations will still be clustered around the optimum of their respective peaks. In order to improve relocation of the optimum of the respective peak by the individuals in the sub-populations, the diversity of each population should be increased. Mendes and Mohais successfully used Brownian individuals. In every generation a predefined number of the weakest individuals are flagged as Brownian. These individuals are then replaced by another individual created by adding a small random number sampled from a zero centered Gaussian distribution to each component of the best individual in the sub-population. A Brownian individual, $\vec{x}_{brown}$, is thus created from the best individual $\vec{x}_{best}$ using the formula

$$\vec{x}_{brown} = \vec{x}_{best} + \vec{\mathcal{N}}(0, \nu) \tag{4}$$

where ν is the standard deviation of the Gaussian distributed random number. Mendes and Mohais (Mendes and Mohais, 2005) showed that the best value of ν to use is 0.2.

4.4. *DE Scheme*

Mendes and Mohais (2005) showed that the most effective scheme to use when following their approach is DE/best/2/bin, in which each temporary individual is created using

$$\vec{v} = \vec{x}_{best} + \mathcal{F} \cdot (\vec{x}_1 + \vec{x}_2 - \vec{x}_3 - \vec{x}_4) \tag{5}$$

where $\vec{x}_1 \neq \vec{x}_2 \neq \vec{x}_3 \neq \vec{x}_4$. The temporary individuals are thus created from four random individuals and the current best individual, $\vec{x}_{best}$.

5. Moving Peaks Benchmark

Branke (June 2007) created the Moving Peaks Benchmark (MPB) in order to address the need for a single, adaptable benchmark that can be used to compare the performance of algorithms aimed at dynamic optimization problems. It has been used by several researchers (Janson and Middendorf, 2003), (Li et al., 2006), (Parrott and Li, 2004), (Trojanowski, 2007). The benchmark contains a moving peaks function and performance measures to evaluate the efficiency of an algorithm. The multi-dimensional problem space of the moving peaks function contains several peaks of variable height, width and shape. These move around with height and width changing periodically.

The MPB allows the following parameters to be set:

- Number of peaks
- Number of dimensions
- Maximum and minimum peak width
- Maximum and minimum peak height
- Change period (the number of function evaluations between successive changes in the environment)
- Change severity (how much the peaks are moved)
- Height severity (standard deviation of changes made to the height of each peak)
- Width severity (standard deviation of changes made to the width of each peak)
- Peak function
- Correlation (between successive movements of a peak)

A static basis function, $B(\vec{x})$, can optionally be added to the problem space.

The performance measure suggested by Branke and Schmeck (2002) is the **offline error**. The offline error is defined as the average of the **current errors** over the entire run, where the current error is defined as the smallest error found since the last change in the environment.

6. Competitive Population Evaluation

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

The MPB measures an algorithm's efficiency at finding solutions quickly by calculating the performance of an optimization algorithm as the average of the error value over the entire run, as opposed to only averaging errors prior to changes in the environment. An algorithm that finds solutions faster would also be beneficial in situations where evaluation criteria are only concerned with the error value prior to changes in the environments, since it would be more robust when the number of function evaluations between changes is reduced.

A dynamic optimization algorithm can thus be improved, not only by reducing the error, but also by making the algorithm reach its lowest error value (before a change occurs in the environment) in fewer function evaluations.

The above argument is the motivation for a novel approach named Competitive Population Evaluation (CPE). The primary goal of the new approach is not to decrease the error value found by DynDE, but rather to make the algorithm

reach the lowest error value in fewer function evaluations.

The basis for the new approach is to allocate function evaluations to populations based on their performance. The best-performing population is evolved on its own until its performance drops below that of another population. The new best performing population is then evolved on its own until its performance drops below that of another population. This allows the location of the highest peak to be discovered early, while the sub-optimum peaks are located later. This approach thus differs from DynDE in that peaks are not located in parallel, but in sequence.

In short, the Competitive Population Evaluation algorithm works as follows:

(i) At the commencement of the run or after a change in the environment occurs, allow the standard DynDE algorithm to run for two generations (two successive evaluations are needed to evaluate equation (6)).
(ii) Calculate the performance value, $\mathcal{P}$ (see equation (6)), for each population.
(iii) Evolve only the population with the highest performance value in the next generation.
(iv) Update the performance value of the population that was evolved.
(v) If no change in the environment has occurred, return to step 2.
(vi) Return to step 1 when a change in the environment occurs.

The performance of a population depends on two factors: The current fitness of the best individual in the population and the amount that the error of the best individual was reduced during the previous evaluation of the population. Let K be the number of populations and $f_k(t)$ be the fitness of the best individual in population k during generation t. The performance $\mathcal{P}$ of population k after generation t is given by:

$$\mathcal{P}(k,t) = (\Delta f_k(t) + 1)(R_k(t) + 1) \tag{6}$$
$$\Delta f_k(t) = |f_k(t) - f_k(t-1)|$$

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

$$R_k(t) = |f_k(t) - \min_{q=1,\ldots,K}\{f_q(t)\}|$$

and

$$R_k(t) = |f_k(t) - \max_{q=1,\ldots,K}\{f_q(t)\}|$$

for function minimization problems. The best performing population will thus be the population with the highest product of fitness and improvement. The motivation for the addition of 1 to the first and second term in equation (6) is to prevent a population being assigned a performance of zero. Without the addition, the least fit population would always be assigned a performance value of zero and would thus never come under consideration for evaluation. Similarly, a good performing population that happens not to have shown any improvement during a specific generation, would also be assigned a performance value of zero and would never be considered for evaluation again. The absolute values of $\Delta f_k(t)$ and $R_k(t)$ are taken to ensure that the performance values are always positive. When a population is reinitialized due to exclusion (see section 4.2), the fitness of the fittest individual is likely to be lower than before reinitialization, which would lead to $\Delta f_k(t)$ being a relatively large number. The population will be assigned a relatively large performance value, making it likely that it would be allocated fitness evaluations in the near future.

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

An advantage of CPE is that it only utilizes information that is available in normal DynDE, so that no extra function evaluations are required.

7. Reinitialization Midpoint Check

Section 4.2 explained how DynDE determines when two populations are located on the same peak, which results in the weaker population being reinitialized. This approach does not take into account the case when two peaks are located extremely close to each other, i.e. within the exclusion threshold. In these situations, one of the populations

will be reinitialized, leaving one of the peaks unpopulated. It is proposed that this problem be partially remedied by determining whether the midpoint between the best individuals in each population constitutes a higher error value than the best individuals of both populations. If this is the case, it implies that a trough exists between the two populations and that neither should be reinitialized (see Figure 1, scenario A). This approach will be referred to as the Reinitialization Midpoint Check (RMC) approach. It is apparent that this strategy will not work in all cases. Scenarios B and C of Figure 1 depict situations where multiple peaks within the exclusion threshold are not detected by a midpoint check. Scenario C further constitutes an example of where no point between the two peaks will give a higher error, thus making it impossible to detect by using any number of intermediate point checks.
This approach is similar, but simpler, than *hill-valley detection* suggested by Ursem (2000).

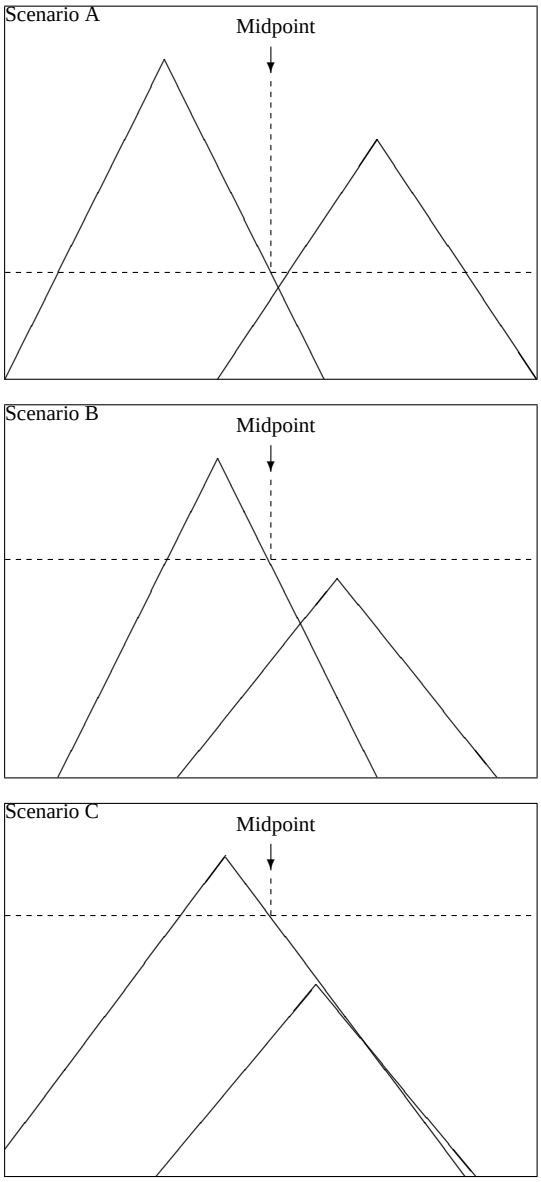

Fig. 1: Midpoint Checking Scenarios

The midpoint check approach provides a method of detecting multiple peaks within the exclusion threshold without being computationally expensive or using too many function evaluations, since only one point is evaluated.

8. Combining RMC and CPE

The RMC and CPE algorithms are mutually independent and can thus be combined in a single algorithm. This algorithm will be referred to as RMCCPE. RMCCPE thus consists of the standard DynDE algorithm with the addition of a midpoint check before reinitializing populations when they are within the exclusion threshold of each other and the evaluation of populations based on their performance.

9. Experimental Results

9.1. *Experimental Procedure*

In the following sections the empirical results comparing the proposed algorithms to the base algorithm, DynDE, will be presented. The MPB settings corresponding to Scenario 2 (Branke, June 2007) are given in Table 1. In addition,

Table 1: MPB settings.

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

the scalability of the algorithms in terms of change severity, change period, number of dimensions and different peak function were investigated, by repeating the experiments for all combinations of the selected settings listed in Table 2. Furthermore, the effect of population size was investigated for sub-population size 6 (2 Brownian individuals)

Table 2: MPB settings.

Setting	**Values**
Nr of Dimensions	5, 10, 15, 20, 25
Change period	1000, 2000, 3000, 4000, 5000
Change severity	1.0, 3.0, 5.0
Peak function	Cone, Sphere

and 10 (5 Brownian individuals). Figures 2 and 3 visually depict the difference between the conical and spherical peak functions. The most apparent difference is that, because spherical peaks do not fan out (do not have a constant gradient) as much as the conical peaks, the absolute minimum on the graph with spherical peaks is much lower than that of the conical peaks. On the other hand, the steeper slopes on the spherical graph should make the hill climbing process simpler, but only until a point close to the absolute minimum, where optimization should become harder.

The various combinations of settings resulted in a total of 300 different experiments being conducted per algorithm. Experiments were run for 500 000 function evaluations each. For each experiment the average offline error over 50 runs along with the confidence interval is reported. A two-sided Mann-Whitney U test was done for each experiment

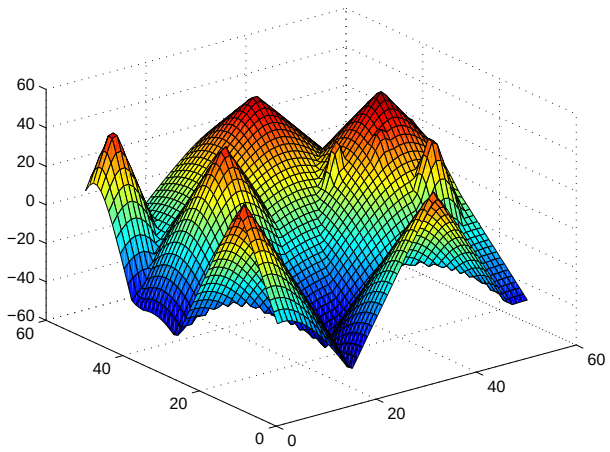

Fig. 2: The Moving Peaks Function using the conical peak function in two dimensions

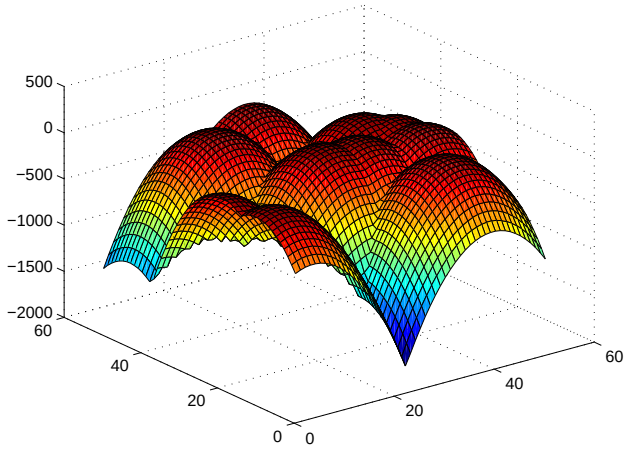

Fig. 3: The Moving Peaks Function using the spherical peak function in two dimensions

to see if the results were statistically significantly different (within a 95% confidence interval) from the corresponding DynDE results.

9.2. *DynDE Results*

Using the MPB settings shown in Table 1, Mendes and Mohais reported an offline error and confidence interval of 1.75 ± 0.032 for DynDE with a population size of 6 (2 Brownian individuals) and 1.94 ± 0.029 for DynDE with a population size of 10 with 5 Brownian individuals (Mendes and Mohais, 2005).

The DynDE algorithm was reimplemented for the purposes of this paper, as described in (Mendes and Mohais, 2005). Considerably better results were found when repeating the experiments. For the MPB settings in Table 1, an offline error of 1.28 ± 0.09 was found for experiments with a population size of 6 and 2 Brownian individuals. Experiments with a population size of 10 and 5 Brownian individuals yielded an offline error of 1.57 ± 0.09. A possible explanation for this discrepancy could be the fact that a different random number seed was used or that small imple-

mentation differences have affected the results positively. It will be assumed in this paper that an algorithm constitutes an improvement over the DynDE algorithm if it yields an offline error significantly lower than 1.57 for a population size of 10 experiment and significantly lower than 1.28 for a population size of 6 experiment.

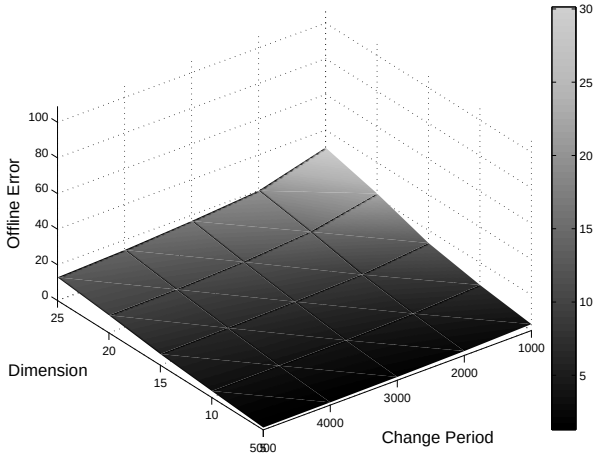

Fig. 4: Offline error of DynDE using the conical peak function with population size of 6 with change severity 1.0 for various settings of dimension and change period.

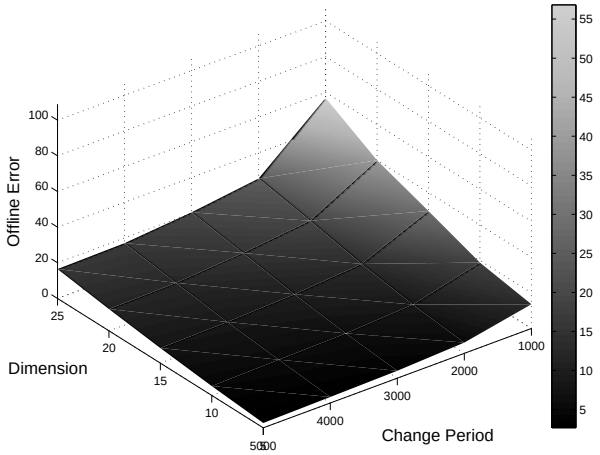

Fig. 5: Offline error of DynDE using the conical peak function with population size of 6 with change severity 3.0 for various settings of dimension and change period.

The results into the scalability of DynDE with respect to the various combinations of settings in Table 2 are summarized in Tables 3 and 4. For the sake of brevity, only the results for population size 6 in 5 dimensions are presented. Full results are available upon request. It was found that in all cases the experiments with a population size of 6 outperformed experiments with a population size of 10. A graphical representation of the DynDE results on the conical peak function can be seen in Figures 4, 5, and 6 for change severity values of 1.0, 3.0, and 5.0 respectively. The same results for the spherical peak function can be seen in Figures 7, 8, and 9. These figures show that similar trends exist for the conical and spherical peak functions, with the effect of increasing the dimension being more pronounced for the spherical peak function. Any change in the MPB settings resulted in a higher offline error being given by the DynDE algorithm. This is to be expected, since all the settings that were investigated do in fact make the problem harder. For example changes occur more frequently and are more severe. On the whole, DynDE performed better on

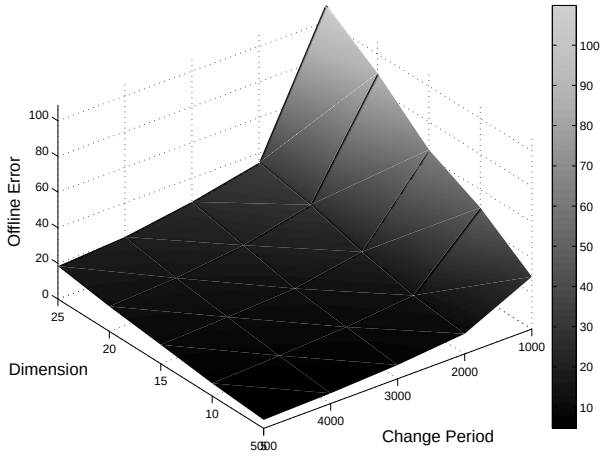

Fig. 6: Offline error of DynDE using the conical peak function with population size of 6 with change severity 5.0 for various settings of dimension and change period.

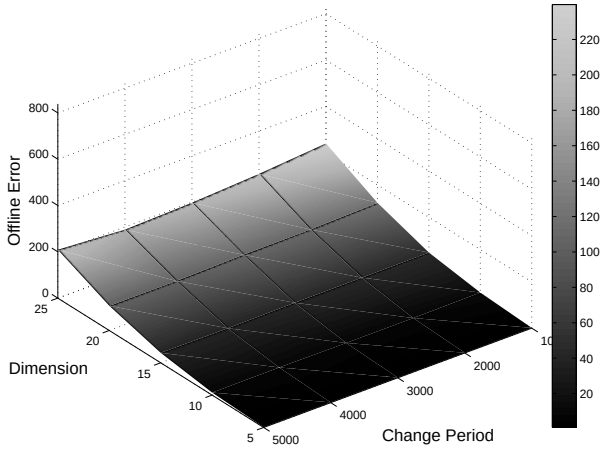

Fig. 7: Offline error of DynDE using the spherical peak function with population size of 6 with change severity 1.0 for various settings of dimension and change period.

the conical peak function than on the spherical, although lower errors were found for the spherical peak function in low dimension, low change severity experiments.

Lowering the change period and increasing the change severity had a very detrimental effect on the DynDE algorithm, especially in the 25 dimensional experiments where the offline error for the extreme combination was 109.89 ± 16.96 on the conical peak function and 839.91 ± 207.21 on the spherical peak function. Given that the minimum peak height is 30, it would appear that none of the peaks are tracked under these conditions.

It can thus be concluded that DynDE does not scale well to higher dimensions, to more frequent and to more severe changes in the environment.

9.3. *CPE Results*

Experiments were conducted to compare the Competitive Population Evaluation algorithm with the standard DynDE algorithm. The outcomes of the Mann-Whitney U tests comparing the CPE results with those of DynDE are listed in Tables 3 and 4. It was found that most experiments yielded a statistically significant difference (i.e. have a Mann-Whitney U test p-value smaller than 0.05). The general trend of the effect of varying the MPB setting was the same

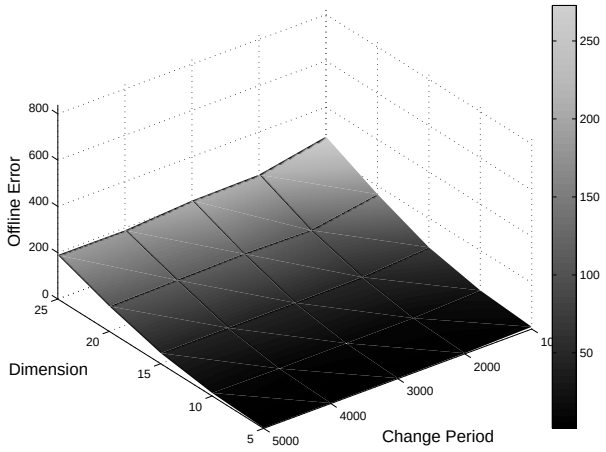

Fig. 8: Offline error of DynDE using the spherical peak function with population size of 6 with change severity 3.0 for various settings of dimension and change period.

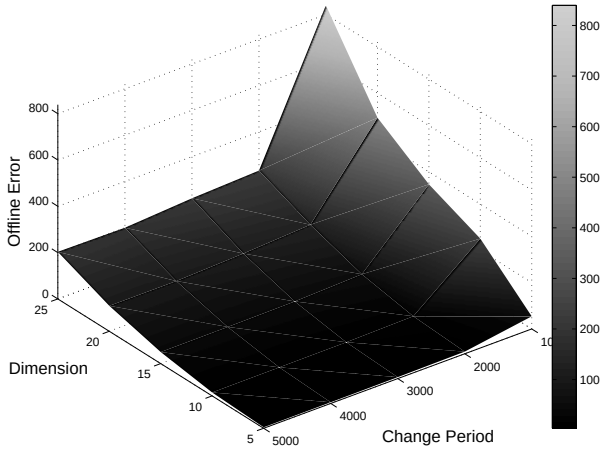

Fig. 9: Offline error of DynDE using the spherical peak function with population size of 6 with change severity 5.0 for various settings of dimension and change period.

for CPE as for DynDE.

CPE yielded a considerable improvement over normal DynDE for all instances using the conical peak function. Considering only the population size 6 experiments, the average improvement is 28.44%, 31.16%, 33.77%, 39.61%, and 42.85% in 5, 10, 15, 20, and 25 dimensions respectively. Figures 10, 11 and 12 graphicly depict the percentage improvement of CPE over DynDE when using the conical function change frequencies of 1.0, 3.0, and 5.0 respectively.

For the spherical peak function experiments, CPE did not yield a statistically significant improvement over normal DynDE for all instances. In the low dimensional, low change severity experiments, the results tend not to be statistically significant. In contrast, significant improvements over DynDE were found in higher dimensions and higher change severity experiments. Considering only the population size 6 experiments, the average improvements are 26.74%, 21.36%, 21.47%, 23.77%, and 23.44% in 5, 10, 15, 20, and 25 dimensions respectively.

It is encouraging that the mentioned improvements are especially pronounced in the more challenging high dimensional, low change period and high change severity problems. In 25 dimensions with a change period of 1000 and change severity of 5, an improvement over standard DynDE of 67.25% was found using the conical peak function and 67.68% using the spherical peak function.

Table 3: Offline error using conical peak function. Change Severity, Change Period, Percentage Improvement over DynDE (% Imp) and Mann-Whitney U test result (p-val)

Change Severity	Change Period	Offline Error	% Imp	p-val	Change Severity	Change Period	Offline Error	% Imp	p-val
DynDE					**RMC**				
1	1000	3.63 ± 0.10	-	-	1	1000	3.49 ± 0.07	3.88 %	0.04
1	2000	2.43 ± 0.10	-	-	1	2000	2.29 ± 0.07	5.59 %	**0.06**
1	3000	1.85 ± 0.09	-	-	1	3000	1.66 ± 0.06	10.35 %	0.00
1	4000	1.47 ± 0.09	-	-	1	4000	1.32 ± 0.07	10.20 %	0.02
1	5000	1.28 ± 0.09	-	-	1	5000	1.14 ± 0.05	11.08 %	0.02
3	1000	13.60 ± 0.96	-	-	3	1000	13.19 ± 0.89	2.97 %	**0.57**
3	2000	6.02 ± 0.16	-	-	3	2000	5.71 ± 0.16	5.16 %	0.01
3	3000	4.10 ± 0.13	-	-	3	3000	4.22 ± 0.14	-3.06 %	**0.34**
3	4000	3.36 ± 0.13	-	-	3	4000	3.17 ± 0.12	5.65 %	0.02
3	5000	2.68 ± 0.12	-	-	3	5000	2.65 ± 0.13	1.31 %	**0.42**
5	1000	29.39 ± 2.21	-	-	5	1000	28.14 ± 2.35	4.25 %	**0.50**
5	2000	11.30 ± 0.29	-	-	5	2000	10.88 ± 0.31	3.71 %	**0.07**
5	3000	7.74 ± 0.25	-	-	5	3000	7.30 ± 0.19	5.68 %	0.01
5	4000	5.77 ± 0.18	-	-	5	4000	5.76 ± 0.17	0.09 %	**0.85**
5	5000	4.76 ± 0.22	-	-	5	5000	4.77 ± 0.17	-0.26 %	**0.65**
CPE					**RMCCPE**				
1	1000	2.85 ± 0.09	21.41 %	0.00	1	1000	2.75 ± 0.07	24.31 %	0.00
1	2000	1.87 ± 0.09	22.99 %	0.00	1	2000	1.72 ± 0.08	29.28 %	0.00
1	3000	1.49 ± 0.11	19.48 %	0.00	1	3000	1.31 ± 0.08	28.97 %	0.00
1	4000	1.20 ± 0.08	18.30 %	0.00	1	4000	1.16 ± 0.07	20.95 %	0.00
1	5000	1.09 ± 0.12	15.31 %	0.00	1	5000	0.92 ± 0.07	28.37 %	0.00
3	1000	7.64 ± 0.20	43.78 %	0.00	3	1000	7.32 ± 0.14	46.17 %	0.00
3	2000	4.27 ± 0.13	29.15 %	0.00	3	2000	4.13 ± 0.12	31.41 %	0.00
3	3000	3.06 ± 0.12	25.26 %	0.00	3	3000	2.98 ± 0.12	27.18 %	0.00
3	4000	2.46 ± 0.13	26.80 %	0.00	3	4000	2.39 ± 0.11	28.96 %	0.00
3	5000	2.01 ± 0.10	25.13 %	0.00	3	5000	1.98 ± 0.10	26.22 %	0.00
5	1000	14.99 ± 0.54	48.99 %	0.00	5	1000	14.81 ± 0.60	49.61 %	0.00
5	2000	7.46 ± 0.21	34.00 %	0.00	5	2000	7.19 ± 0.21	36.35 %	0.00
5	3000	5.13 ± 0.21	33.76 %	0.00	5	3000	4.96 ± 0.16	35.97 %	0.00
5	4000	4.10 ± 0.16	28.94 %	0.00	5	4000	3.92 ± 0.16	31.96 %	0.00
5	5000	3.18 ± 0.11	33.22 %	0.00	5	5000	3.26 ± 0.18	31.46 %	0.00

9.4. *RMC Results*

The results of experiments conducted to investigate the effectiveness of the RMC approach are listed in Tables 3 and 4. The outcomes of the Mann-Whitney U tests comparing the RMC results with those of DynDE indicate that, on the whole, only experiments in 5 dimensions yield a statistically significantly different result from DynDE. Furthermore,

Table 4: Offline error using spherical peak function. Change Severity, Change Period, Percentage Improvement over DynDE (% Imp) and Mann-Whitney U test result (p-val)

Change Severity	Change Period	Offline Error	% Imp	p-val	Change Severity	Change Period	Offline Error	% Imp	p-val
DynDE					**RMC**				
1	1000	1.66 ± 0.14	-	-	1	1000	1.30 ± 0.07	21.46 %	0.00
1	2000	1.20 ± 0.10	-	-	1	2000	1.00 ± 0.08	16.87 %	0.01
1	3000	1.03 ± 0.11	-	-	1	3000	0.86 ± 0.07	16.30 %	0.02
1	4000	1.04 ± 0.09	-	-	1	4000	0.76 ± 0.07	27.27 %	0.00
1	5000	0.91 ± 0.10	-	-	1	5000	0.74 ± 0.07	18.75 %	0.01
3	1000	9.66 ± 1.03	-	-	3	1000	9.58 ± 1.00	0.84 %	**0.80**
3	2000	3.27 ± 0.10	-	-	3	2000	2.89 ± 0.05	11.75 %	0.00
3	3000	2.44 ± 0.10	-	-	3	3000	2.15 ± 0.06	11.83 %	0.00
3	4000	1.91 ± 0.07	-	-	3	4000	1.76 ± 0.07	7.71 %	0.00
3	5000	1.70 ± 0.08	-	-	3	5000	1.53 ± 0.07	10.12 %	0.00
5	1000	54.49 ± 11.02	-	-	5	1000	65.70 ± 11.65	-20.58 %	**0.59**
5	2000	9.29 ± 0.16	-	-	5	2000	8.93 ± 0.18	3.81 %	0.00
5	3000	6.13 ± 0.11	-	-	5	3000	5.84 ± 0.06	4.69 %	0.00
5	4000	4.79 ± 0.10	-	-	5	4000	4.56 ± 0.07	4.67 %	0.00
5	5000	3.95 ± 0.07	-	-	5	5000	3.83 ± 0.09	3.24 %	0.00
CPE					**RMCCPE**				
1	1000	1.31 ± 0.10	21.29 %	0.00	1	1000	1.12 ± 0.08	32.37 %	0.00
1	2000	1.06 ± 0.10	11.66 %	**0.05**	1	2000	0.85 ± 0.08	29.16 %	0.00
1	3000	0.90 ± 0.08	12.14 %	**0.15**	1	3000	0.74 ± 0.07	28.24 %	0.00
1	4000	0.95 ± 0.10	9.13 %	**0.10**	1	4000	0.75 ± 0.08	28.34 %	0.00
1	5000	0.94 ± 0.10	-3.24 %	**0.61**	1	5000	0.66 ± 0.08	26.92 %	0.00
3	1000	4.16 ± 0.10	56.87 %	0.00	3	1000	3.61 ± 0.08	62.63 %	0.00
3	2000	2.42 ± 0.09	25.95 %	0.00	3	2000	2.10 ± 0.06	35.71 %	0.00
3	3000	1.85 ± 0.09	24.18 %	0.00	3	3000	1.59 ± 0.07	34.68 %	0.00
3	4000	1.55 ± 0.09	18.58 %	0.00	3	4000	1.29 ± 0.06	32.25 %	0.00
3	5000	1.34 ± 0.08	21.04 %	0.00	3	5000	1.15 ± 0.07	32.33 %	0.00
5	1000	15.14 ± 1.21	72.21 %	0.00	5	1000	14.92 ± 1.15	72.62 %	0.00
5	2000	5.89 ± 0.11	36.61 %	0.00	5	2000	5.54 ± 0.06	40.30 %	0.00
5	3000	4.12 ± 0.07	32.69 %	0.00	5	3000	3.95 ± 0.06	35.49 %	0.00
5	4000	3.22 ± 0.07	32.83 %	0.00	5	4000	3.08 ± 0.06	35.65 %	0.00
5	5000	2.80 ± 0.07	29.22 %	0.00	5	5000	2.61 ± 0.06	34.03 %	0.00

only some of the 5 dimensional results are statistically significant, but they show a considerable improvement over DynDE. Improvements are more pronounced when using the spherical peak function; an average improvement of 20.13% was found over DynDE for 5 dimensional experiments with change severity of 1 and population size of 6. For all high dimensional experiments almost none of the results are statistically significant. A possible explanation for this fact could be that peaks are less likely to appear within the exclusion threshold of each other in higher

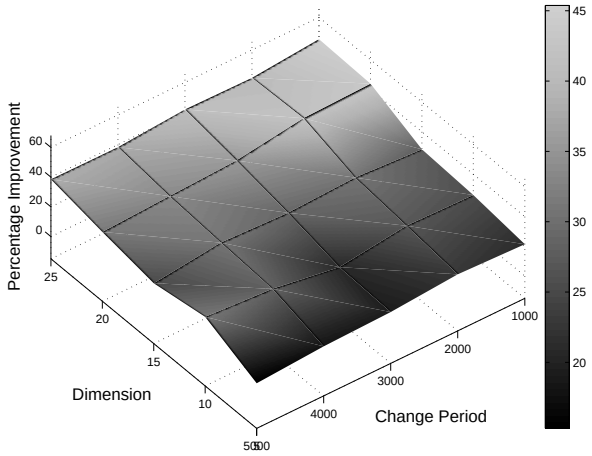

Fig. 10: Percentage improvement of CPE over DynDE using the conical peak function with population size of 6 with change severity 1.0 for various settings of dimension and change period.

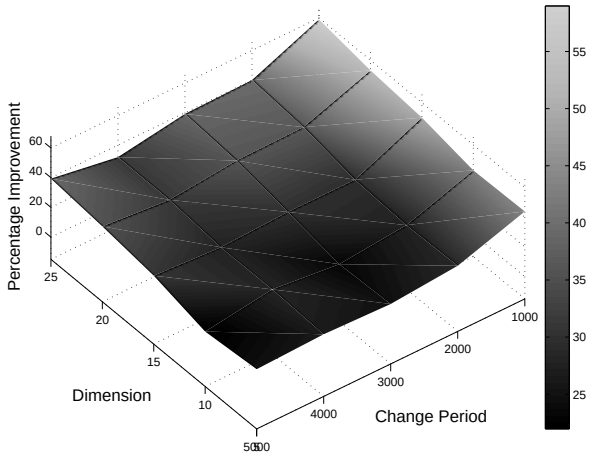

Fig. 11: Percentage improvement of CPE over DynDE using the conical peak function with population size of 6 with change severity 3.0 for various settings of dimension and change period.

dimensions, or that more cases, as depicted in Figure 1, scenarios B and C, occur. Encouragingly, only 2 of the 300 experiments yielded a result statistically significantly worse than DynDE. As with standard DynDE, the population size 6 experiments outperformed the population size 10 experiments.

Figure 13 graphically depicts the percentage improvement of RMC over DynDE in 5 dimensions for the conical peak function.

9.5. *RMCCPE Results*

The results for the combined RMCCPE approach are summarized in Tables 3 to 4. Using a Mann-Whitney U test, it was found that all conical peak function results and most spherical peak function results were statistically significantly different from DynDE. In all cases, RMCCPE performed better than DynDE. Once again, the population size 6 experiments outperformed the population size 10 experiments. Figures 14, 15, and 16 graphicly depict the percentage improvement of RMCCPE over DynDE when using the conical function change frequencies of 1.0, 3.0, and 5.0 respectively. Considering only the population size 6 experiments on the conical peak function, the average improvements over DynDE are 31.81%, 31.21%, 33.31%, 40.51%, and 42.97% in 5, 10, 15, 20, and 25 dimensions

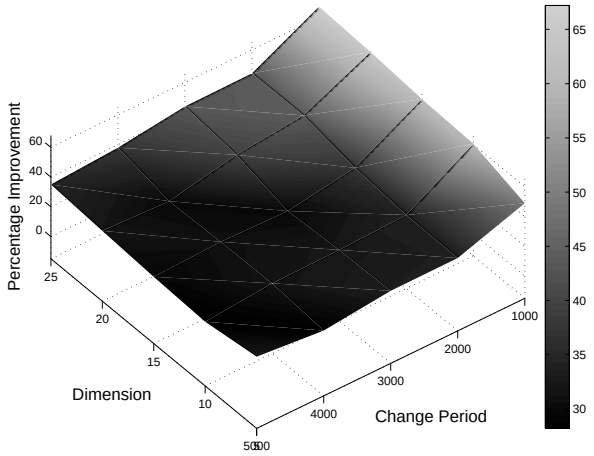

Fig. 12: Percentage improvement of CPE over DynDE using the conical peak function with population size of 6 with change severity 5.0 for various settings of dimension and change period.

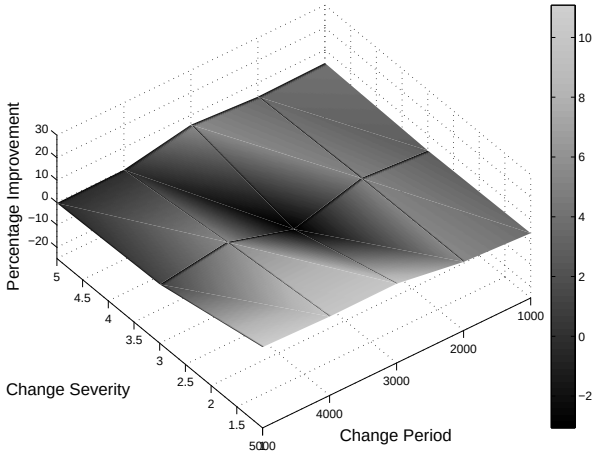

Fig. 13: Percentage improvement over DynDE of the Midpoint Check approach in 5 dimensions for various values for change period and change severity using a population size of 6 on the conical peak function.

respectively. For the population size 6 experiments, the spherical peak function resulted in improvements of 37.38%, 21.42%, 20.56%, 24.85%, and 25.31% in 5, 10, 15, 20, and 25 dimensions respectively.

In order to determine whether RMCCPE constitutes an improvement over its sub-components RMC and CPE, Mann-Whitney U tests were conducted comparing RMC to RMCCPE and CPE to RMCCPE. The results are omitted for the sake of brevity. It was found that in virtually all cases RMCCPE constituted a statistically significant improvement over RMC. This result is not surprising, considering that CPE generally outperformed RMC.

In the comparison between CPE and RMCCPE it was found that statistically significant improvements of RMCCPE over CPE were isolated mainly to 5 dimensional experiments. This is consistent with the fact that the improvement of RMC over DynDE was also mainly in the low dimensional experiments. In the 300 experiments investigated it was found that RMCCPE yielded a statistically significant improvement over CPE in 41 of the cases, while CPE was only statistically significantly better than RMCCPE in 5 of the cases.

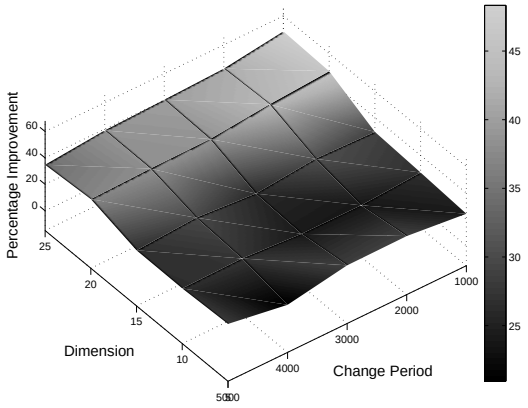

Fig. 14: Percentage improvement of RMCCPE over DynDE using the conical peak function with population size of 6 with change severity 1.0 for various settings of dimension and change period.

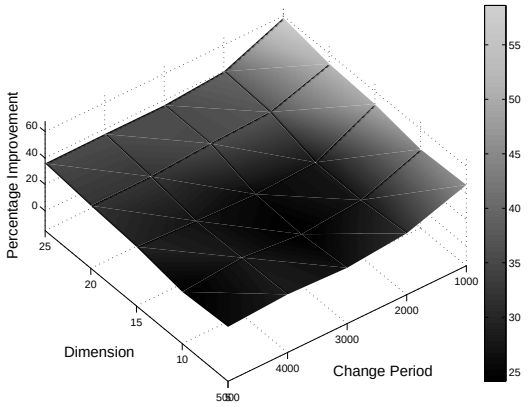

Fig. 15: Percentage improvement of RMCCPE over DynDE using the conical peak function with population size of 6 with change severity 3.0 for various settings of dimension and change period.

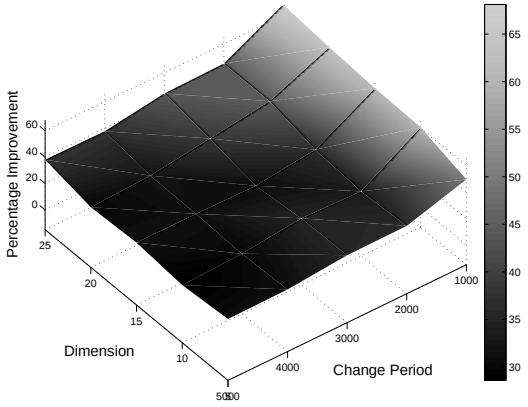

Fig. 16: Percentage improvement of RMCCPE over DynDE using the conical peak function with population size of 6 with change severity 5.0 for various settings of dimension and change period.

10. Comparison with other approaches

It has been shown in this paper that using RMC and CPE yielded better results than DynDE alone. In this section the combined RMCCPE algorithm will be compared to other approaches in the literature.

Using the MPB benchmark, the most successful algorithm in the literature is the work of Moser and Hendtlass (2007), called Multi-Phase Multi-Individual Extremal Optimisation (MMEO). Using this algorithm to solve the MPB with the settings in Table 1 resulted in an offline error of 0.66 ± 0.20. This result is superior to the result found using RMCCPE which was 0.92 ± 0.07. However, Moser and Hendtlass point out that MMEO is not expected to be very scalable with regard to increasing the number of dimensions. Through personal communication with Moser (2007), it was established that MMEO yields an offline error of 2.67 ± 0.17 in 10 dimensions and an offline error of 5.09 ± 0.36 was found in 15 dimensions. In the same experiments, RMCCPE resulted in an offline error of 2.70 ± 0.23 and 4.55 ± 0.29, respectively. RMCCPE is thus on a par with MMEO in 10 dimensions and outperforms MMEO in 15 dimensions.

After MMEO, the best reported results using the MPB are those of Blackwell and Branke (2006). This Particle Swarm based approach incorporates many of the same principles as DynDE, for example multiple populations and exclusion. An offline error of 1.75 ± 0.06 was reported for the MPB settings in Table 1. RMCCPE produced a 47.52% lower offline error. For the high change severity of 3 experiments, Blackwell and Branke found an offline error of 3.00 ± 0.06 as opposed to the RMCCPE result of 1.98 ± 0.10; and for the change severity of 5 experiments, Blackwell and Branke found an offline error of 4.24 ± 0.10, as opposed to the RMCCPE result of 3.26 ± 0.18. In 10 dimensional experiments Blackwell and Branke found an offline error of 4.17 ± 0.15 while using RMCCPE resulted in an offline error of 2.70 ± 0.23 (a result that is 35.29% better).

The 2009 Congress on Evolutionary Computation (CEC2009) ran a dynamic optimization competition. This competition used the Generalized Benchmark Generator (GBG) of Li et al. (2008), (Li and Yang, 2008) to compare the results of competing algorithms. GBG consists of six test functions which are optimized for six different change types. A seventh change type wherein the dimension of the functions is changed is included in the benchmark. The evaluation criteria used by the GBG is the average best error just before a change occurs in the environment and an overall performance mark based on the relative best fitness sampled at specific intervals.

The winning algorithm of this competition was *jDE*, developed by Brest et al. (2009). *jDE* was implemented by the present authors in order to compare its performance to the algorithms discussed in this paper.

RMCCPE was run on the GBG with the settings recommended for the CEC2009 competition (the dynamic dimension change type experiments were omitted, since the algorithms presented here have not been adapted for those types of problems). RMCCPE yielded results inferior to *jDE*. Since the recommended change period of the GBG is relatively high (100 000 function evaluations compared to 5000 which is standardly used by the MPB), the relative scalability of *jDE*, DynDE and RMCCPE with respect to change period, was investigated. Table 5 lists the results of this investigation. The number of test cases for which each algorithm performed the best out of the 42 is listed per change period along with the average error over all the problems and the overall performance mark (as calculated without the dynamic dimension problems). It can be seen that, while *jDE* is considerably superior in large change period experiments, it is outperformed by DynDE and RMCCPE in the small change period experiments.

Full results of *jDE* and RMCCPE on the GBG with a change period of 5000 can be seen in Tables 6 and 7 respectively. *jDE* performed better than RMCCPE on only one of the test cases, function 3 with change type 5.

jDE was run on the MPB for the various settings listed in Table 2 for the cone peak function. The results of the 5 dimensional experiments can be seen in Table 8.

Comparing these results to those found using RMCCPE in Table 3 shows that RMCCPE performed better than *jDE* on the MPB.

It can thus be seen that the RMCCPE approach presented here compares favourably with some of the state of the art approaches in the literature.

11. Discussion and Conclusions

CPE is a novel extension to DynDE that yields significantly better results, especially for high change frequency and high change severity problems. Noteworthy improvements were also found for the more complicated high dimensional problems.

Table 5: jDE compared to DynDE and RMCCPE for different values of Change Period (**CP**)

Change Period	**Measure**	**JDE**	**DynDE**	**RMCCPE**
5000	Number best	1	13	28
	Average error	653.00	453.33	408.43
	Performance	10.18	13.58	14.99
10000	Number best	11	15	16
	Average error	508.56	357.61	348.03
	Performance	17.43	16.77	17.15
20000	Number best	26	7	9
	Average error	318.40	303.05	303.31
	Performance	30.08	19.66	19.39
40000	Number best	33	8	1
	Average error	176.38	268.03	273.32
	Performance	43.35	22.68	21.96
100000	Number best	34	8	0
	Average error	98.87	238.05	243.19
	Performance	59.83	27.56	26.51

RMC resulted in an improvement in only some of the tested scenarios and only in the low dimensional cases. In comparison with CPE, the improvement yielded by RMC was also much smaller. Despite this, RMC is still considered to be a worthwhile addition to DynDE, because it is relatively simple to implement and, with the exception of only two cases, none of the statistically significant results indicated that it makes the standard DynDE algorithm worse.

On average, the combination of RMC and CPE is in several cases more successful than each of the approaches on its own. Experiments comparing the performance of RMCCPE to normal DynDE indicated that large improvements were in problems with high change frequency and severity, making RMCCPE a more viable option.

Comparison with other approaches showed that although RMCCPE is not the best algorithm for large change period problems, RMCCPE outperforms most other algorithms in the literature on the more difficult small change period problems.

RMCCPE does not depend on any intrinsic DE behaviour. Future work could include studying the effect of applying RMCCPE to other multi-population optimization algorithms for dynamic environments.

Table 6: jDE results on GBG with change period of 5000

Problem	Peaks	Errors	T_1	T_2	T_3	T_4	T_5	T_6
1	10	Avg_best	1.54	2.54	1.97	2.04	6.45	3.25
		Avg_worst	57.07	73.60	61.70	86.55	69.63	88.36
		Avg_mean	21.74	33.73	26.45	44.58	34.95	54.41
		STD	4.57	3.10	5.38	5.25	3.40	3.31
	50	Avg_best	2.14	4.00	2.94	2.36	6.16	4.53
		Avg_worst	59.30	69.80	63.24	86.64	64.24	87.62
		Avg_mean	24.97	32.67	28.19	45.89	29.21	53.52
		STD	2.94	2.74	3.85	4.18	1.94	2.96
2		Avg_best	19.28	96.01	93.27	10.95	148.55	25.94
		Avg_worst	367.85	595.92	594.00	673.75	568.52	872.35
		Avg_mean	137.07	422.64	433.87	270.64	424.51	434.12
		STD	34.77	38.56	21.57	31.06	15.89	27.74
3		Avg_best	481.20	769.86	764.85	583.61	796.10	621.76
		Avg_worst	1011.81	1353.86	1333.80	1744.39	1226.60	2065.57
		Avg_mean	837.50	1142.60	1111.75	1038.42	1059.47	1222.77
		STD	34.89	41.77	22.82	43.23	41.21	49.99
4		Avg_best	21.12	95.50	96.41	16.48	258.44	26.68
		Avg_worst	450.37	681.08	659.30	751.09	632.53	911.72
		Avg_mean	222.74	514.47	498.62	335.64	499.27	478.13
		STD	35.68	16.76	10.61	30.88	25.87	27.14
5		Avg_best	109.76	198.41	175.44	19.21	274.70	86.31
		Avg_worst	1730.24	1940.88	1964.50	1945.49	1906.54	2056.61
		Avg_mean	1171.46	1434.35	1402.92	1080.19	1337.72	1335.96
		STD	176.27	62.37	74.37	97.01	71.81	120.00
6		Avg_best	43.77	264.16	199.72	24.46	315.63	113.61
		Avg_worst	656.36	1107.51	1148.87	1097.80	1159.24	1476.75
		Avg_mean	356.93	853.89	878.82	595.29	890.16	870.92
		STD	82.87	28.44	25.00	76.97	33.47	59.23

Table 7: RMCCPE results on GBG with change period of 5000

Problem	Peaks	Errors	T_1	T_2	T_3	T_4	T_5	T_6
1	10	Avg_best	4.05	2.75	2.29	4.64	4.96	4.40
		Avg_worst	31.25	42.65	41.05	54.22	38.55	61.04
		Avg_mean	11.86	14.35	15.57	25.11	15.95	29.21
		STD	0.95	1.19	5.02	2.17	3.00	1.31
	50	Avg_best	4.51	2.67	2.75	4.35	4.02	4.12
		Avg_worst	39.07	36.70	37.13	54.78	33.97	60.89
		Avg_mean	15.65	14.98	17.21	26.10	14.02	30.72
		STD	1.57	1.74	4.61	1.58	0.96	1.95
2		Avg_best	46.17	94.61	92.19	48.40	163.98	53.30
		Avg_worst	172.01	457.32	464.76	267.19	467.40	448.18
		Avg_mean	96.93	272.16	294.96	130.22	349.23	197.22
		STD	5.60	29.60	16.93	16.58	9.83	25.61
3		Avg_best	654.92	929.17	922.50	700.58	907.05	752.48
		Avg_worst	1091.19	1301.19	1289.14	1693.59	1294.80	1995.49
		Avg_mean	825.86	1106.20	1086.99	1031.94	1078.78	1206.80
		STD	31.80	12.06	11.23	21.29	35.04	40.59
4		Avg_best	60.19	115.70	129.11	47.28	199.43	70.21
		Avg_worst	212.03	570.32	563.09	319.83	555.09	560.89
		Avg_mean	118.82	369.13	379.98	148.92	440.37	247.82
		STD	6.90	37.49	17.27	14.51	12.65	24.96
5		Avg_best	141.32	184.94	185.79	132.78	229.91	165.95
		Avg_worst	446.09	577.36	1726.06	703.61	1868.12	2051.77
		Avg_mean	289.50	386.09	467.87	302.53	547.07	520.73
		STD	18.46	14.03	57.23	20.98	72.83	67.91
6		Avg_best	72.00	116.48	119.46	73.93	163.43	100.28
		Avg_worst	271.22	906.92	1029.84	751.81	1068.21	1295.35
		Avg_mean	146.88	432.05	643.52	239.23	731.73	498.66
		STD	16.33	65.11	67.80	71.12	50.62	98.75

Table 8: jDE offline error using conical peak function for different values of Change Severity and Change Period

Change Sev.	**Change Period**	**Offline Error**	**Change Sev.**	**Change Period**	**Offline Error**	**Change Sev.**	**Change Period**	**Offline Error**
1	1000	13.75 ± 0.86	3	1000	19.99 ± 0.68	5	1000	24.78 ± 0.65
1	2000	8.97 ± 0.46	3	2000	15.83 ± 0.68	5	2000	19.28 ± 0.68
1	3000	7.06 ± 0.43	3	3000	13.15 ± 0.48	5	3000	16.87 ± 0.64
1	4000	6.79 ± 0.46	3	4000	11.22 ± 0.49	5	4000	14.50 ± 0.58
1	5000	5.88 ± 0.31	3	5000	10.00 ± 0.37	5	5000	12.74 ± 0.52

References

Blackwell, T., Branke, J., 2004. Multiswarm optimization in dynamic environments. Applications of Evolutionary Computing 3005, 489–500.

Blackwell, T., Branke, J., 2006. Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation 10 (4), 459–472.

Blackwell, T. M., Bentley, P. J., 2002. Dynamic search with charged swarms. In: et al., W. B. L. (Ed.), Genetic and Evolutionary Computation Conference. Morgan Kaufmann, pp. 19–26.

Boettcher, S., Percus, A. G., 13-17 July 1999. Extremal optimization: Methods derived from co-evolution. In: Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., Smith, R. E. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference. Vol. 1. Morgan Kaufmann, Orlando, Florida, USA, pp. 825–832.

Branke, J., 2002. Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell, MA, USA.

Branke, J., June 2007. The moving peaks benchmark. http://www.aifb.uni-karlsruhe.de/ ˜jbr/MovPeaks/.

Branke, J., Kaussler, T., Schmidt, C., Schmeck, H., 2000. A multi-population approach to dynamic optimization problems. In: Adaptive Computing in Design and Manufacturing. Springer, pp. 299–308.

Branke, J., Schmeck, H., 2002. Designing evolutionary algorithms for dynamic optimization problems. In: Tsutsui, S., Ghosh, A. (Eds.), Theory and Application of Evolutionary Computation: Recent Trends. Springer, pp. 239–262.

Branke, J., Schmeck, H., 2003. Designing evolutionary algorithms for dynamic optimization problems, 239–262.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V., Dec. 2006. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. Evolutionary Computation, IEEE Transactions on 10 (6), 646–657.

Brest, J., Zamuda, A., Boškovic, B., Maučec, M. S., Žumer, V., 2009. Dynamic optimization using self-adaptive differential evolution. In: CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation. IEEE Press, Piscataway, NJ, USA, pp. 415–422.

Cobb, H. G., 1990. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Navy Center for Applied Research in Artificial Intelligence, Technical Report AIC-90-001.

Grefenstette, J. J., 1999. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In: Congress on Evolutionary Computation. Vol. 3. IEEE, pp. 2031–2038.

Hu, X., Eberhart, R., 2002. Adaptive particle swarm optimisation: detection and response to dynamic systems. In: Congress on Evolutionary Computation. pp. 1666–1670.

Janson, S., Middendorf, M., 2003. A hierachical particle swarm optimizer. In: Congress on Evolutionary Computation. IEEE, pp. 770–776.

Janson, S., Middendorf, M., 2004. A hierachical particle swarm optimizer for dynamc optimization problems. In: Raidl, G. R. (Ed.), Applications of evolutionary computing. Vol. 3005 of LNCS. Springer, pp. 513–524.

Jin, Y., Branke, J., 2005. Evolutionary optimization in uncertain environments - a survey. IEEE Transactions on Evolutionary Computation 9 (3), 303–317.

K. Price, R. Storn, J. L., 2005. Differential evolution - A practical approach to global optimization. Springer.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: International Conference on Neural Networks. IEEE, pp. 1942–1948.

Li, C., Yang, S., 2008. A generalized approach to construct benchmark problems for dynamic optimization. In: SEAL '08: Proceedings of the 7th International Conference on Simulated Evolution and Learning. Springer-Verlag, Berlin, Heidelberg, pp. 391–400.

Li, C., Yang, S., Nguyen, T. T., Yu, E. L., Yao, X., Jin, Y., g. Beyer, H., Suganthan, P. N., 2008. Benchmark Generator for CEC2009 Competition on Dynamic Optimization. University of Leicester, University of Birmingham, Nanyang Technological University, Technical Report.

Li, X., Branke, J., Blackwell, T., 2006. Particle swarm with speciation and adaptation in a dynamic environment. In: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, New York, NY, USA, pp. 51–58.

Li, X., Dam, K. H., 2003. Comparing particle swarms for tracking extrema in dynamic environments. In: Congress on

Evolutionary Computation. Vol. 3. IEEE, pp. 1772– 1779.
Mendes, R., Mohais, A., 2005. DynDE: a differential evolution for dynamic optimization problems. In: Congress on Evolutionary Computation. IEEE, pp. 2808–2815.
Mori, N., Imanishi, S., Kita, H., Nishikawa, Y., 1997. Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. In: Bäck, T. (Ed.), International Conference on Genetic Algorithms. Morgan Kaufmann, pp. 299–306.
Mori, N., Kita, H., Nishikawa, Y., 1996. Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In: Voigt, H.-M. (Ed.), Parallel Problem Solving from Nature. No. 1141 in LNCS. Springer Verlag Berlin, pp. 513–522.
Mori, N., Kita, H., Nishikawa, Y., 1998. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In: Eiben, A. E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (Eds.), Parallel Problem Solving from Nature. No. 1498 in LNCS. Springer, pp. 149–158.
Morrison, R., 2004. Designing Evolutionary Algorithms for Dynamic Environments. Springer.
Moser, I., May 2007. Personal communication.
Moser, I., Hendtlass, T., 2007. A simple and efficient multi-component algorithm for solving dynamic function optimisation problems. In: Congress on Evolutionary Computation. IEEE, pp. 252–259.
Oppacher, F., Wineberg, M., 1999. The shifting balance genetic algorithm: Improving the GA in a dynamic environment. In: et al., W. B. (Ed.), Genetic and Evolutionary Computation Conference (GECCO). Vol. 1. Morgan Kaufmann, San Francisco, pp. 504 – 510.
Parrott, D., Li, X., 2004. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In: Congress on Evolutionary Computation. IEEE, pp. 98–103.
Ramsey, C. L., Grefenstette, J. J., 1993. Case-based initialization of genetic algorithms. In: Forrest, S. (Ed.), International Conference on Genetic Algorithms. Morgan Kaufmann, pp. 84–91.
Storn, R., 1996. On the usage of differential evolution for function optimization. In: Biennial Conference of the North American Fuzzy Information Processing Society. IEEE, pp. 519–523.
Storn, R., Price, K., 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359.
Trojanowski, K., 2007. B-cell algorithm as a parallel approach to optimization of moving peaks benchmark tasks. In: 6th International Conference on Computer Information Systems and Industrial Management Applications. CISIM '07, pp. 143–148.
Ursem, R. K., 2000. Multinational GA optimization techniques in dynamic environments. In: Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.-G. (Eds.), Genetic and Evolutionary Computation Conference. Morgan Kaufmann, pp. 19–26.
Vavak, F., Jukes, K., Fogarty, T. C., 1997. Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search. In: Bäck, T. (Ed.), International Conference on Genetic Algorithms. Morgan Kaufmann, pp. 719–726.
Yang, S., 2005. Memory-based immigrants for genetic algorithms in dynamic environments. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM, New York, NY, USA, pp. 1115–1122.