Differential Evolution for Dynamic Environments with Unknown Numbers of Optima

Mathys C. du Plessis · Andries P. Engelbrecht

Abstract This paper investigates optimization in dynamic environments where the numbers of optima are unknown or fluctuating. The authors present a novel algorithm, Dynamic Population Differential Evolution (DynPopDE), which is specifically designed for these problems. DynPopDE is a Differential Evolution based multi-population algorithm that dynamically spawns and removes populations as required. The new algorithm is evaluated on an extension of the Moving Peaks Benchmark. Comparisons with other state-of-the-art algorithms indicate that DynPopDE is an effective approach to use when the number of optima in a dynamic problem space is unknown or changing over time.

Keywords Differential Evolution · Dynamic Environments · Competing Populations · Moving Peaks · Dynamic Number of Populations

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 [23]. The lack of genetic diversity of the individuals in the population is the main drawback of most of the standard evolutionary algorithms when they are applied to dynamic problems. This is because the algorithms tend to converge on a single optimum in the solution space and then lack the diversity to locate

Mathys C. du Plessis
Department of Computing Sciences, Nelson Mandela Metropolitan University, Port Elizabeth, South Africa
Tel.: +27-41-5042076
Fax: +27-41-5042831
E-mail: mc.duplessis@nmmu.ac.za

Andries P. Engelbrecht
Department of Computer Science, University of Pretoria, Pretoria, South Africa
E-mail: engel@cs.up.ac.za

new optima when they appear. Differential Evolution (DE) is one of the evolutionary algorithms that does not scale well to dynamic environments due to a lack of diversity [34].

Most current research in applying DE to dynamic environments focuses on problems where the number of optima is known and does not change over time. This is the case with the popular Moving Peaks Benchmark (MPB) [5] and the Generalized Benchmark Generator [18]. While the phrase "varying number of peaks" is often used by researchers, this generally refers only to the fact that some peaks are obscured by others for a brief period during the algorithm's run, and not to the number of peaks fluctuating during the run [4], [19].

Previously, the authors presented Competitive Population Evaluation DE (CDE) [10], a DE based multi-population algorithm, which is based on DynDE [21], one of the first DE-based algorithms for dynamic environments. It was shown that CDE performed better than DynDE on problems where the number of optima are known [10]. In this paper, the performance of CDE on problems where the number of optima is unknown or fluctuating is investigated and compared to results found using CDE's predecessor algorithm, DynDE.

The main contribution of this work is a novel extension to CDE, namely Dynamic Population Differential Evolution (DynPopDE), which is aimed specifically at problems where the number of optima is unknown or fluctuating. DynPopDE differs from CDE in that the number of populations used is not constant over time.

The performance of DynDE, CDE and DynPopDE is evaluated by using an extension of Branke's MPB which allows for fluctuating numbers of peaks.

The rest of the paper is structured as follows: related work is described in Section 2 and the DE algorithm is discussed in Section 3. The Moving Peaks Benchmark and extensions to allow for fluctuating numbers of peaks is discussed in Section 4. Section 5 reviews the research of Mendes and Mohais on DynDE. CDE is reviewed in Section 6, while the novel approach DynPopDE is presented in Section 7. The results of an investigation into the performance of DynDE, CDE and DynPopDE are given in Section 8. DynPopDE is compared to other state-of-the-art optimization algorithms for dynamic environments in Section 9. In addition, a population spawning and removing technique suggested by Blackwell [2] is incorporated into CDE and compared to DynPopDE. Finally, conclusions are drawn in Section 10.

2 Related Work

Jin and Branke [15] provided a survey on algorithms for dynamic optimization which are discussed along with some of the more recent advances in the field in this section.

Cobb [9] suggested drastically increasing the mutation rate of a Genetic Algorithm (GA) after a change in the environment has occurred, while Vavak et al. [32] advocated a more gradual increase. Hu and Eberhart [12] suggested that particles in a Particle Swarm Optimization (PSO) algorithm should be reinitialized when a change in the environment has occurred. Approaches aimed at maintaining a high amount of diversity during the entire run include Grefenstette's Random Immigrants [11] and refinements made by Yang [33]. These introduce random individuals into the GA's population after each generation. In contrast, Morrison [23] made use of stationary individuals (called sentinels) that are uniformly distributed around the search space. These sentinels in-

crease diversity by providing an influx of genetic material from points apart from the convergence area of the GA's population.

The thermodynamic GA [22] 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 PSO approaches include charged particles [1], 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 [3], [4]. These individuals are called Quantum individuals. 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 [21]. Investigations into finding an appropriate neighborhood structure for PSO [14], [20] and an appropriate reproduction scheme for DE [21] have been conducted, since these parameters greatly affect the diversity of the population.

Several strategies employ memory to retain information regarding the location of the 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) [6], Oppacher and Wineberg's Shifting Balance GA (SBGA) [26] and Ursem's Multinational GA (MGA) [31]. All three of these approaches make 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 the current generation.

While SOS aims to keep the bulk of the population in a single population searching for new optima, the 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 the 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, such as PSO and DE, to dynamic optimization. Parrot and Li [27] suggested a multiple-population PSO approach to optimizing dynamic problems, called speciation. The social component of PSO provides a simple method of dividing the population

into sub-populations. In this algorithm, a particle is grouped 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 [4] introduced a multiple-swarm PSO-based algorithm that is based on three components: Exclusion, Anti-convergence and Quantum individuals. An interesting novelty about their approach is that all swarms contain the same number of individuals. The aim of having multiple swarms is that each population should be positioned on its own promising optimum in the environment. Unfortunately swarms often converge to the same optimum, hence decreasing diversity. Exclusion [3] is a technique meant to prevent swarms from clustering around the same optimum by means of reinitializing swarms that stray within a threshold Euclidean distance from a better-performing swarm. 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 swarms 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.

Blackwell [2] further adapted the PSO-based algorithm of Blackwell and Branke [4] by self-adapting the number of swarms in the search space. This algorithm is aimed at situations where the number of optima in the dynamic environment is unknown. Swarms are generated when the number of free swarms that have not converged to an optimum (M_{free}) have dropped to zero. Conversely, swarms are removed if M_{free} is less than a threshold parameter, n_{excess}. A swarm is classified as converged if all particles are located within a diameter of $2r_{conv}$. The value of r_{conv} is calculated using equation 5.

Li et al. [19] improved the speciation algorithm by introducing quantum individuals [3] to increase the diversity and anti-convergence to detect stagnation and subsequently to reinitialize the worst-performing populations. This algorithm is called the Speciation-based PSO (SPSO).

Mendes and Mohais [21] adapted the ideas from Blackwell and Branke [4] 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 optimum. DynDE will be discussed in detail in later sections.

Brest et al. [7], [8] 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 from 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 an offspring individual performs significantly better than its 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.

Recently, Noroozi et al. [25] proposed a DE-based algorithm aimed at dynamic environments, referred to as CellularDE. This algorithm divides the search space into equally sized cells which are used to delineate sub-populations. A limit is placed on the maximum number of individuals that are allowed to be located in each cell (and consequently the sub-population size). Once this limit is exceeded, the worst performing individual within the overpopulated cell is reinitialized in a randomly selected cell. For each sub-population, the DE mutations and crossovers are only applied to individuals in the same or surrounding cells.

It is interesting to note that several earlier algorithms supported the number of populations changing over the course of the run (e.g. SOS [6] and MGA [31]), while later algorithms (for which better results are reported) made use of a constant number of populations (e.g. Blackwell and Branke's PSO-based approach [4], DynDE [21] and jDE [8]).

3 Differential Evolution

Differential Evolution (DE) is a relatively new optimization algorithm based on Darwinian evolution, created by Storn and Price [16], [29]. Mutations are based on the spacial difference between two or more individuals added to a target vector, as opposed to other evolutionary algorithms where mutation step sizes are generally sampled from a random distribution. Several variants to the DE algorithm have been suggested, but a generic algorithm is as follows [16]:

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

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

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

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

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

Most variations on DE (known as schemes) are based on different approaches to create each of the temporary individuals, $\overrightarrow{v}_i$ [28] (see equation (1)), and how offspring is 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 Moving Peaks Benchmark

Branke [5] 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. This has been used by several researchers [13],[24],[27],[30]. 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. The position, height and width of each peak changes periodically.

The MPB allows the following parameters to be set:

- Number of peaks
- Number of dimensions
- Maximum and minimum peak widths
- Maximum and minimum peak heights
- 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)

The performance measure suggested by Branke et al. [6] 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.

Part of this study involves an investigation into dynamic environments in which the number of peaks fluctuates. The MPB was therefore adapted by the current authors to allow the number of peaks to change when a change occurs in the environment. For the adapted MPB, the number of peaks, $m(t)$, is now calculated as:

$$m(t) = \begin{cases} \max\{1, m(t-1) - M * U(0,1) * P\} \\ \qquad \text{if } U(0,1) < 0.5 \\ \min\{M, m(t-1) + M * U(0,1) * P\} \\ \qquad \text{otherwise} \end{cases} \tag{3}$$

where M is the maximum number of peaks and P is the maximum fraction of the maximum number of peaks that can be added or removed from the population after a change in the environment. P thus controls the severity of the change in the number of peaks. For example, a value of $P = 1$ will result in up to M peaks being added or removed, while a value of $P = 0.1$ will result in a change of up to 10% in M, where M and P are included as parameters of the benchmark function.

5 DynDE

DynDE is a differential evolution algorithm developed by Mendes and Mohais [21] to solve dynamic optimization problems. Their algorithm makes use of approaches similar to those applied by Blackwell and Branke [4] to Particle Swarm Optimization (PSO) in dynamic environments. The most successful versions of DynDE make use of multiple populations, exclusion and Brownian individuals to adapt DE to dynamic environments.

5.1 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 an entirely different peak may become the optimal peak (or one of several) once a change in the environment occurs. Consequently, not only must the movement of the best peak found in the problem space be tracked, but also the sub-optimal peaks. An effective method to achieve this is to maintain several independent populations of DE individuals, one on each peak. In their most successful experiments, Mendes and Mohais used 10 populations, each containing 6 individuals.

5.2 Exclusion

In order to track all peaks, it is necessary to ensure that all populations converge to different peaks. If all the populations converged to the optimum peak this would defeat the purpose of having multiple populations. Mendes and Mohais used exclusion to prevent populations from converging to the same peak. This 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 then randomly reinitialized. A threshold is used to determine if two individuals are too close. This is calculated as follows:

$$r_{excl} = \frac{X}{2p^{\frac{1}{d}}} \tag{4}$$

where X is the range of the d dimensions (assuming equal ranges for all dimensions), and p is the number of peaks. From equation (4) it can be seen that the exclusion threshold increases with an increase in the number of dimensions and decreases if the number of peaks is increased.

In the above approach it is assumed that the number of peaks is known. In the absence of information on the number of peaks, this study suggests that equation (4) be changed to:

$$r_{excl} = \frac{X}{2K^{\frac{1}{d}}} \tag{5}$$

where K is the number of populations, thus making the threshold dependent on the number of available populations. The same equation was used by Blackwell [2] in the self-adapting multi-swarms algorithm.

5.3 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 maximum of their respective peaks. In order to improve the 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 for this purpose. In every generation a pre-defined 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-centred Gaussian distribution to each component of the best individual in the sub-population. A Brownian individual, $\overrightarrow{x}_{brown}$, is thus created from the best individual $\overrightarrow{x}_{best}$ using the formula:

$$\overrightarrow{x}_{brown} = \overrightarrow{x}_{best} + \overrightarrow{\mathcal{N}}(0, \nu) \tag{6}$$

where ν is the standard deviation of the Gaussian distributed random number. Mendes and Mohais [21] showed that an appropriate value of ν is 0.2.

5.4 DE Scheme

Mendes and Mohais [21] showed that the most effective scheme to use when following their approach is DE/best/2/bin, where each temporary individual is created using:

$$\overrightarrow{v} = \overrightarrow{x}_{best} + \mathcal{F} \cdot (\overrightarrow{x}_1 + \overrightarrow{x}_2 - \overrightarrow{x}_3 - \overrightarrow{x}_4) \tag{7}$$

with $\overrightarrow{x}_1 \neq \overrightarrow{x}_2 \neq \overrightarrow{x}_3 \neq \overrightarrow{x}_4$. The temporary individuals are thus created from four randomly selected individuals and the current best individual, $\overrightarrow{x}_{best}$.

6 Competitive Population Evaluation Differential Evolution

CDE [10] is an algorithm based on DynDE, containing two additional components, namely: Competitive Population Evaluation and Reinitialization Midpoint Check.

6.1 Competitive Population Evaluation (CPE)

The basis for the CPE approach [10] is to allocate function evaluations to populations based on performance. The best-performing population is evolved on its own until its performance drops below that of another population. At this point another population will be identified as the best-performing population, which is then evolved on its own until its performance drops. This allows the location of the highest peak to be discovered early, while the suboptimum peaks are located later. This approach thus differs from DynDE in that peaks are located sequentially rather than in parallel.
In short, the Competitive Population Evaluation algorithm works as follows:

1. At the commencement of the running of the algorithm or after a change in the environment occurs, allow the standard DynDE algorithm to run for two generations.
2. Calculate the performance value, $\mathcal{P}$, for each population.

3. Evolve only the population with the highest performance value in the next generation.
4. Update the performance value of the population that evolved.
5. If no change in the environment has occurred, return to step 2.
6. 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:

$$\begin{aligned} \mathcal{P}(k,t) &= (\Delta f_k(t) + 1)(R_k(t) + 1) \\ \Delta f_k(t) &= |f_k(t) - f_k(t-1)| \end{aligned} \tag{8}$$

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, for function minimization problems,

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

The absolute values of $\Delta f_k(t)$ and $R_k(t)$ are taken to ensure that the performance values are always positive. The best-performing population will thus be the population with the highest product of fitness and improvement. Equation (8) explains why the standard DynDE algorithm is allowed to run for two generations after a change in the environment; since two successive evaluations are required to evaluate the equation.

By competitively choosing the better-performing populations to evolve before other populations, the lowest error value could be reached sooner. This is beneficial when a low error is required during the entire run of the algorithm. Alternatively, when the lowest error value is only required just before changes occur in the environment, the algorithm should perform better on environments with more frequent changes. This technique has the added advantage that better-performing populations will receive more function evaluations. These would otherwise have been wasted on finding the maximum of the sub-optimal peaks. The overall error value should consequently also be reduced.

Changes in the environment are detected by periodically re-evaluating the best individual found since the last change in the environment.

An advantage of CPE is that only information that is available in normal DynDE is utilized, thus requiring no additional function evaluations.

6.2 Reinitialization Midpoint Check

Section 5.2 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. The Reinitialization Midpoint

Check (RMC) approach partially remedies this problem 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. This strategy will not work in all situations, but 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.

7 DynPopDE

In this section a novel approach specifically designed to address problems with unknown or a fluctuating number of peaks is introduced. Dynamic Population Differential Evolution (DynPopDE) is based on CDE, but with three new extensions, namely spawning new populations, removing populations, and the introduction of a penalty factor on performance.

7.1 Spawning populations

As is illustrated in section 8, even if the number of peaks is known, creating an equal number of populations as peaks is not an effective strategy. When the total number of peaks is unknown, choosing the number of populations to use would be, at best, an educated guess. It is therefore suggested that the number of populations should not be a parameter of the algorithm, but that populations should be spawned as needed. The question that must be answered is: When should new populations be spawned? Blackwell [2] spawned populations based on the number of swarms that have not converged to within a threshold radius. In contrast, DynPopDE makes use of information inherent to CDE to determine when populations should be spawned.

CDE is based on allocating processing time and function evaluations to populations based on performance. A detection scheme is proposed to indicate when evolution has reached a point of little or no improvement in error for current populations. In this context, this point will be referred to as stagnation. When stagnation is detected, DynPopDE introduces a new population of random individuals. Equation (8) shows that the performance of an individual is calculated as the product of its current fitness and the improvement made in fitness during the previous generation. It is suggested that it would be a meaningful indicator of stagnation if all of the current populations received a zero improvement of fitness after their last respective function evaluations. Let $\mathcal{K}$ be the set of current populations. We define a function, $\Upsilon(t)$, that is true if stagnation occurred, as follows:

$$\Upsilon(t) = \begin{cases} true \text{ if}(\Delta f_k(t) = 0) \ \forall \ k \in \mathcal{K} \\ false \text{ otherwise} \end{cases} \tag{9}$$

Note that this approach does not guarantee that stagnation of all populations has permanently occurred, but it does indicate that function evaluations are not effectively used by the current populations, and may thus be employed to detect more peaks in the environment.

This approach allows DynPopDE to commence with only a single population and adapt to an appropriate number of populations.

7.2 Removing populations

The previous section explained how new populations are spawned when necessary. However, it is possible that equation (9) detects stagnation incorrectly, and more populations than necessary may be created. Furthermore, in problems where the number of peaks fluctuate, it would be desirable to remove superfluous populations when the number of peaks decreases. It thus becomes necessary to detect redundant populations when the number of populations outnumber the number of peaks.

In DynDE a population is reinitialized when the spatial difference between the population and a more fit population drops below the exclusion threshold and, in CDE, the midpoint check does not detect two distinct optima. It is reasonable to assume that when redundant populations are present, these populations will perpetually be reinitialized and will not converge to specific peaks. Consequently, redundant populations can be detected by finding populations that are successively reinitialized without reaching a point of apparent convergence. A population, k, will be discarded when it is flagged for reinitialization due to exclusion, and

$$\Delta f_k(t) \neq 0 \tag{10}$$

Once again, this approach does not guarantee the removal of a population only when the number of populations outnumbers the peaks, since more than one population may optimize the same peak even when more peaks than populations are present. However, since no information is lost by this process (populations are only removed when flagged for reinitialization), new populations can effectively be reintroduced by the spawning process when necessary.

7.3 Penalty Factor

The detection of stagnation is essential to DynPopDE's population-spawning process. In order to detect stagnation effectively, it is necessary to distribute function evaluations more uniformly among all the populations.

The final component of DynPopDE is the introduction of a penalty factor into equation (8) to penalize populations for successively receiving the highest performance value without showing any improvement in fitness.

The performance with penalty, $\mathcal{P}_{pen}(k,t)$, is calculated as:

$$\mathcal{P}_{pen}(k,t) = \begin{cases} \frac{\mathcal{P}(k,t)}{pen_k(t)} & \text{if}(pen_k(t) > 0) \\ \mathcal{P}(k,t) & \text{otherwise} \end{cases} \tag{11}$$

where the penalty factor, $pen_k(t)$, of population k is calculated as:

$$pen_k(t) = \begin{cases} pen_k(t-1) + 1 & \text{if}(\Delta f_k(t) = 0) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

The penalty factor is thus reset to zero as soon as an improvement of fitness is found.

This approach is contrary to the spirit of CDE where populations are favoured for performing well, and function evaluations are not wasted on suboptimal peaks. In fact, it will be shown in Section 8 that when a penalty factor is introduced into CDE without the spawning and population removal components, a decrease in performance

is observed. There is thus an intrinsic cost in detecting stagnation which takes the form of a necessary allocation of more function evaluations to weaker populations to ensure a reasonable chance to stagnate. It will be shown that DynPopDE is significantly improved by the incorporation of the penalty factor, since this improves the detection of stagnation. Pseudo-code of the DynPopDE algorithms is given in Algorithm 1.

Algorithm 1 ***DynPopDE Pseudo-code***

Create one sub-population of random individuals
while *termination criterion not met* ***begin***
    if *Change detected* ***or*** *First generation*
    begin
        Reevaluate fitness of all individuals
        count := 0
    end
    if *count* $<$ *2*
    begin
        for all *sub-populations*
        begin
            Create offspring using DE/best/2/bin
            Insert better performing offspring into sub-population
        end
    end
    else begin
        for all *sub-populations calculate* $\mathcal{P}_{pen}$
        Create offspring using DE/best/2/bin for sub-population with highest $\mathcal{P}_{pen}$
        Insert better performing offspring into sub-population
    end
    if Υ ***then*** *introduce new random sub-population*
    Remove exclusion sub-populations subject to RMC
    Reinitialize excluded sub-populations for which $\Delta f = 0$
    Evaluate fitness of populations reinitialized by exclusion
    Create Brownian individuals
    count := count + 1
end

8 Experimental Results

In order to investigate performance of DynDE and CDE and to validate DynPopDE, a comparative study into the performance of DynPopDE, CDE and DynDE was conducted over a large number of combinations of settings of the MPB. In Section 8.2 the effects of having an unknown number of optima is investigated. Fluctuating numbers of optima are investigated in Section 8.3.

8.1 Experimental Procedure

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 Mann-

Whitney U test was done for relevant experiments to determine if differences between the results of the respective algorithms were statistically significant. The algorithms all used the following parameter values (following results obtained by Mendes and Mohais [21]): Sub-population size = 6; Number of Brownian individuals per sub-population = 2; Brownian radius, $\nu = 0.2$.

8.2 Unknown Number of Peaks Experiments

DynDE, CDE and DynPopDE were compared on problems where the number of peaks is unknown. This was done by investigating several variations of MPB settings that correspond to Scenario 2 [5] (see Table 1).

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 algorithms were run for various settings of the number of peaks. In addition, the scalability of the algorithms in terms of the change periods and the number of dimensions were investigated by repeating the experiments for all combinations of the selected settings listed in Table 2.

Table 2 MPB settings for unknown number of peaks experiments.

Setting	**Values**
Nr of Dimensions	5, 10, 15
Change period	1000, 2000, 3000, 4000, 5000
Number of Peaks	5, 10, 20, 40, 80, 160

As a baseline, experiments were conducted to determine the offline error of DynDE and CDE when the number of peaks is known, i.e. the same number of populations as peaks were used. The results of these experiments can be found in Table 3. Similar trends were found in 10 and 15 dimensions. Consequently, the 10 and 15 dimensional results are omitted here. Full results are available from the authors. A visual depiction of the offline error of DynDE is provided in Figure 1.

The results show that both CDE and DynDE yielded extremely large errors for higher numbers of peaks.
Since the ideal number of populations is not known when an unknown number of peaks

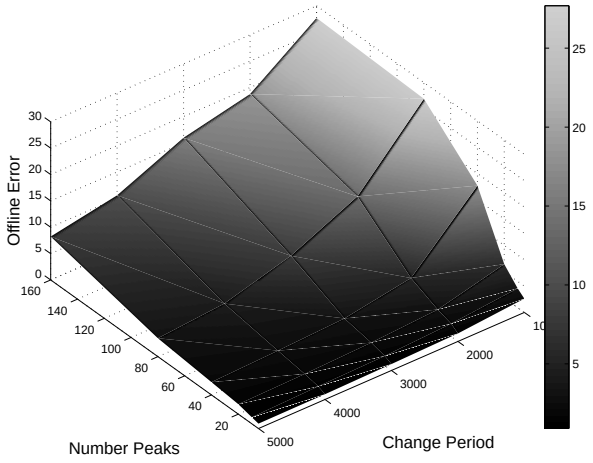

Fig. 1 Offline error of DynDE for different change periods in 5 dimensions when the same number of populations as number of peaks is used.

exists in the problem space, the same experiments were repeated with both DynDE and CDE using 10 populations in all cases. This number was found to produce reasonable results for the purpose of a baseline comparison. Results are given in Table 3. Figure 2 depicts the DynDE results in 5 dimensions when using 10 populations. Comparing Figures 1 and 2 highlights the dissimilar behaviour that was found by using a constant number of sub-populations. The effect of reducing the change period is less pronounced with respect to increasing the number of peaks, and the increase in offline error, due to increasing the number of peaks, levels off for number of peaks greater than about 40.

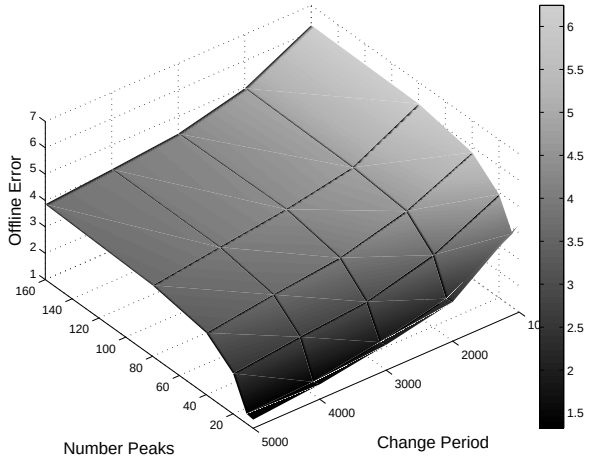

Fig. 2 Offline error of DynDE for different change periods in 5 dimensions when 10 populations are used.

Table 3 Offline error in 5 dimensional experiments with various numbers of peaks (**#P**) and change periods (**CP**) of DynDE with equal population size and number of peaks (**DynDE**), DynDE with population size 10 (**DynDE 10**), CDE with equal population size and number of peaks (**CDE**), CDE with population size 10 (**CDE 10**), CDE with penalty factor (**CDE 10 Pen**), and the result of a Mann-Whitney U test comparing DynDE with population size 10 to CDE with population size 10 (**p-Val**). Values that are below a 95% confidence level are in boldface.

#P	CP	DynDE	DynDE 10	CDE	CDE 10	CDE 10 Pen	p-Val
5	1000	2.63 ± 0.16	4.36 ± 0.13	1.82 ± 0.11	2.83 ± 0.10	3.14 ± 0.08	0.00
	2000	1.53 ± 0.12	2.49 ± 0.09	1.22 ± 0.13	1.72 ± 0.06	2.02 ± 0.07	0.00
	3000	1.23 ± 0.13	1.96 ± 0.09	0.90 ± 0.09	1.27 ± 0.06	1.61 ± 0.10	0.00
	4000	0.99 ± 0.13	1.56 ± 0.12	0.82 ± 0.08	1.05 ± 0.07	1.39 ± 0.07	0.00
	5000	0.91 ± 0.15	1.33 ± 0.08	0.76 ± 0.14	0.86 ± 0.05	1.22 ± 0.11	0.00
10	1000	3.66 ± 0.09	3.80 ± 0.09	2.80 ± 0.08	2.75 ± 0.08	2.90 ± 0.08	0.00
	2000	2.47 ± 0.09	2.43 ± 0.09	1.72 ± 0.07	1.75 ± 0.08	1.94 ± 0.07	0.00
	3000	1.84 ± 0.09	1.88 ± 0.11	1.34 ± 0.07	1.36 ± 0.08	1.57 ± 0.11	0.00
	4000	1.58 ± 0.11	1.48 ± 0.08	1.11 ± 0.08	1.15 ± 0.11	1.35 ± 0.11	0.00
	5000	1.23 ± 0.07	1.36 ± 0.10	1.00 ± 0.08	1.04 ± 0.10	1.22 ± 0.11	0.00
20	1000	6.45 ± 0.39	4.92 ± 0.10	4.04 ± 0.06	4.07 ± 0.08	4.28 ± 0.10	0.00
	2000	3.34 ± 0.08	3.72 ± 0.14	2.62 ± 0.05	3.10 ± 0.08	3.42 ± 0.11	0.00
	3000	2.56 ± 0.07	3.05 ± 0.13	1.96 ± 0.05	2.75 ± 0.13	2.93 ± 0.10	0.00
	4000	2.12 ± 0.07	2.82 ± 0.13	1.59 ± 0.06	2.50 ± 0.13	2.61 ± 0.14	0.00
	5000	1.87 ± 0.06	2.53 ± 0.13	1.42 ± 0.05	2.35 ± 0.13	2.56 ± 0.20	**0.07**
40	1000	17.69 ± 2.48	5.79 ± 0.10	18.69 ± 2.45	4.95 ± 0.09	5.11 ± 0.09	0.00
	2000	5.67 ± 0.29	4.66 ± 0.16	3.52 ± 0.06	4.08 ± 0.14	4.04 ± 0.12	0.00
	3000	3.51 ± 0.08	3.99 ± 0.14	2.71 ± 0.05	3.55 ± 0.12	3.91 ± 0.16	0.00
	4000	2.90 ± 0.07	3.69 ± 0.18	2.32 ± 0.06	3.25 ± 0.17	3.61 ± 0.16	0.00
	5000	2.42 ± 0.06	3.34 ± 0.18	1.99 ± 0.05	3.20 ± 0.17	3.33 ± 0.18	**0.29**
80	1000	26.91 ± 1.93	6.18 ± 0.10	25.27 ± 2.12	5.36 ± 0.10	5.47 ± 0.10	0.00
	2000	13.89 ± 1.71	4.86 ± 0.12	13.21 ± 1.62	4.31 ± 0.14	4.63 ± 0.14	0.00
	3000	8.13 ± 0.95	4.38 ± 0.11	3.92 ± 0.11	3.94 ± 0.14	4.24 ± 0.15	0.00
	4000	4.57 ± 0.21	4.04 ± 0.18	2.99 ± 0.05	3.79 ± 0.19	3.96 ± 0.16	0.04
	5000	3.49 ± 0.09	3.69 ± 0.19	2.68 ± 0.06	3.61 ± 0.19	3.74 ± 0.17	**0.47**
160	1000	27.69 ± 1.05	6.25 ± 0.11	28.26 ± 0.99	5.37 ± 0.10	5.49 ± 0.10	0.00
	2000	18.77 ± 1.34	4.95 ± 0.11	19.03 ± 1.29	4.45 ± 0.11	4.65 ± 0.14	0.00
	3000	15.98 ± 1.18	4.34 ± 0.14	15.21 ± 1.25	3.92 ± 0.13	4.25 ± 0.16	0.00
	4000	10.55 ± 1.08	4.08 ± 0.15	11.03 ± 1.16	3.87 ± 0.14	4.07 ± 0.19	**0.06**
	5000	8.23 ± 0.85	3.86 ± 0.21	4.76 ± 0.26	3.61 ± 0.13	3.63 ± 0.14	**0.06**

When using 10 populations, a much lower offline error is found for a high number of peaks for both DynDE and CDE. Figure 3 plots the average offline error of DynDE and CDE per number of peaks for situations when an equal number of populations as peaks are used and when a constant number of 10 peaks are used. An interesting observation can be made from Figure 3. Much better results are found when a constant number of 10 peaks are used. This indicates that, even if the number of peaks is known, it is better to use a lower number of populations for a large number of peaks. A plausible explanation for this phenomenon is that while it is desirable to track all peaks, the low number of function evaluations that are available between changes in the environment makes using a large number of populations infeasible.

Note that for number of peaks equal to 5, CDE with matching number of peaks and populations outperformed CDE with 10 populations. This was the only situation where knowing the number of peaks was advantageous.

In all cases CDE outperformed DynDE. The Mann-Whitney U test results listed in Tables 3 indicate that CDE performed statistically significantly better than DynDE in all but 5 of the 5 dimensional experiments. A statistically significant improvement of CDE over DynDE was found in all the 10 and 15 dimensional experiments.

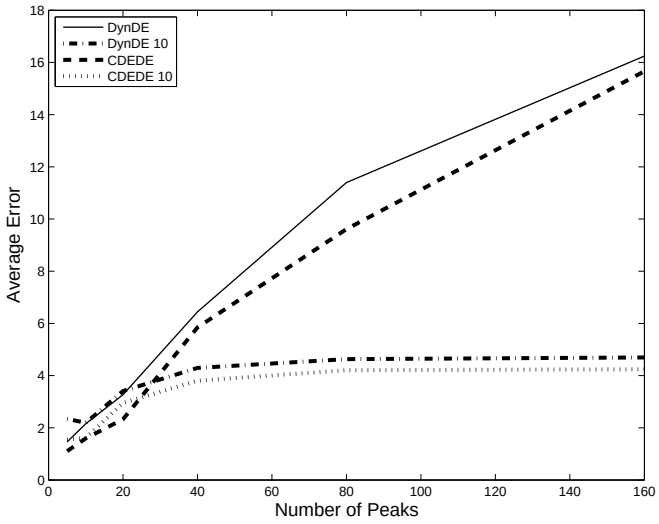

Fig. 3 Average Offline error of DynDE and CDE when the same number of populations as peaks are used, and with a constant number of populations of 10.

Before DynPopDE results are given, results will be presented to show that the penalty factor described in Section 7.3 yields an increase in offline error when introduced into CDE. Table 3 lists the offline error of CDE with 10 populations without and with the penalty factor component in 5 dimensions. In all cases the offline error was increased by the incorporation of the penalty factor. Similar trends were found in 10 and 15 dimensions.

The same experiments that were done for DynDE and CDE were performed using DynPopDE, with and without, the penalty factor. The results are summarized in Table 4. In contrast with what was found for CDE, the penalty factor positively affects the results of DynPopDE. Figure 4 illustrates how the number of populations is adapted by DynPopDE for various numbers of peaks (results depicted are averaged over 30 repeats in each case). DynPopDE commences with a single population and then introduces more populations as is required. Although the number of populations had not stabilized in the graphs of the experiments with the larger number of peaks, the figure demonstrates that more populations are created in situations where more peaks are present.

DynPopDE performed statistically significantly better than DynDE in all experiments, except for 3 of the 10 peak experiments, where the differences were not statistically significant.

A similar trend indicating the superiority of DynPopDE was evident in the high-dimensional experiments. DynPopDE performed statistically significantly better than DynDE in all but 2 of the 10 dimensional experiments and in all 15 dimensional experiments.

DynPopDE outperforms CDE in most situations except those where the number of peaks is equal to 10. This is reasonable since CDE made use of 10 populations. However, the results of the Mann-Whitney U tests listed in Table 4 indicate that the improvements are not always statistically significant. In 5 dimensions the results indi-

Table 4 Offline error in experiments with various numbers of peaks (**#P**) and change periods (**CP**) of DynDE with population size 10, CDE with population size 10 (**CDE 10**) DynPopDE without penalty factor (**DynPopDE WP**), and DynPopDE (**DynPopDE**). The p-value outcomes of a Mann-Whitney U test comparing CDE with population size 10 to DynPopDE (**p-A**), and DynDE with population size 10 to DynPopDE (**p-B**) are given with values that are below a 95% confidence level printed in boldface.

#P	CP	DynDE 10	CDE 10	DynPopDE WP	DynPopDE	p-A	p-B
5	1000	4.36 ± 0.13	2.83 ± 0.10	2.41 ± 0.11	2.41 ± 0.08	0.00	0.00
	2000	2.49 ± 0.09	1.72 ± 0.06	1.69 ± 0.13	1.67 ± 0.10	**0.08**	0.00
	3000	1.96 ± 0.09	1.27 ± 0.06	1.23 ± 0.12	1.29 ± 0.07	**0.90**	0.00
	4000	1.56 ± 0.12	1.05 ± 0.07	1.10 ± 0.15	1.12 ± 0.09	**0.26**	0.00
	5000	1.33 ± 0.08	0.86 ± 0.05	1.09 ± 0.14	0.94 ± 0.07	**0.17**	0.00
10	1000	3.80 ± 0.09	2.75 ± 0.08	3.15 ± 0.10	3.08 ± 0.08	0.00	0.00
	2000	2.43 ± 0.09	1.75 ± 0.08	2.55 ± 0.13	2.14 ± 0.08	0.00	0.00
	3000	1.88 ± 0.11	1.36 ± 0.08	2.32 ± 0.20	1.74 ± 0.09	0.00	**0.06**
	4000	1.48 ± 0.08	1.15 ± 0.11	2.54 ± 0.24	1.54 ± 0.11	0.00	**0.54**
	5000	1.36 ± 0.10	1.04 ± 0.10	2.61 ± 0.29	1.31 ± 0.09	0.00	**0.43**
20	1000	4.92 ± 0.10	4.07 ± 0.08	4.52 ± 0.12	4.02 ± 0.08	**0.33**	0.00
	2000	3.72 ± 0.14	3.10 ± 0.08	3.72 ± 0.17	2.69 ± 0.07	0.00	0.00
	3000	3.05 ± 0.13	2.75 ± 0.13	3.78 ± 0.25	2.19 ± 0.07	0.00	0.00
	4000	2.82 ± 0.13	2.50 ± 0.13	3.78 ± 0.27	1.86 ± 0.07	0.00	0.00
	5000	2.53 ± 0.13	2.35 ± 0.13	3.56 ± 0.25	1.72 ± 0.07	0.00	0.00
40	1000	5.79 ± 0.10	4.95 ± 0.09	5.07 ± 0.12	4.76 ± 0.10	0.00	0.00
	2000	4.66 ± 0.16	4.08 ± 0.14	4.13 ± 0.16	3.33 ± 0.08	0.00	0.00
	3000	3.99 ± 0.14	3.55 ± 0.12	4.30 ± 0.21	2.74 ± 0.08	0.00	0.00
	4000	3.69 ± 0.18	3.25 ± 0.17	4.33 ± 0.25	2.36 ± 0.10	0.00	0.00
	5000	3.34 ± 0.18	3.20 ± 0.17	4.32 ± 0.33	1.97 ± 0.06	0.00	0.00
80	1000	6.18 ± 0.10	5.36 ± 0.10	5.44 ± 0.13	5.07 ± 0.11	0.00	0.00
	2000	4.86 ± 0.12	4.31 ± 0.14	4.68 ± 0.20	3.64 ± 0.08	0.00	0.00
	3000	4.38 ± 0.11	3.94 ± 0.14	4.22 ± 0.24	3.01 ± 0.06	0.00	0.00
	4000	4.04 ± 0.18	3.79 ± 0.19	4.10 ± 0.24	2.59 ± 0.06	0.00	0.00
	5000	3.69 ± 0.19	3.61 ± 0.19	4.25 ± 0.28	2.25 ± 0.06	0.00	0.00
160	1000	6.25 ± 0.11	5.37 ± 0.10	5.30 ± 0.11	5.12 ± 0.08	0.00	0.00
	2000	4.95 ± 0.11	4.45 ± 0.11	4.34 ± 0.16	3.78 ± 0.07	0.00	0.00
	3000	4.34 ± 0.14	3.92 ± 0.13	4.41 ± 0.21	3.00 ± 0.07	0.00	0.00
	4000	4.08 ± 0.15	3.87 ± 0.14	3.97 ± 0.23	2.68 ± 0.07	0.00	0.00
	5000	3.86 ± 0.21	3.61 ± 0.13	3.71 ± 0.23	2.40 ± 0.07	0.00	0.00

cate that CDE and DynPopDE are not consistently different in 5 peak experiments. CDE performed statistically significantly better that DynPopDE in the 10 peak experiments. However, DynPopDE was statistically significantly better than CDE in all but one of the 20 peak experiments and performed significantly better than CDE in all experiments with more than 20 peaks.

In 10 and especially 15 dimensions the differences between DynPopDE and CDE are less pronounced. Out of the 30 experiments in 10 dimensions, 12 results were not statistically significant, and 19 of the results in 15 dimensions were not statistically significantly different. However, DynPopDE performed better than CDE in all experiments that were statistically significant with more than 10 peaks in 10 dimensions and all but one of the experiments that were statistically significant in 15 dimensions.
In all these cases the number of peaks was fixed. The next section presents the results associated with a fluctuating number of peaks.

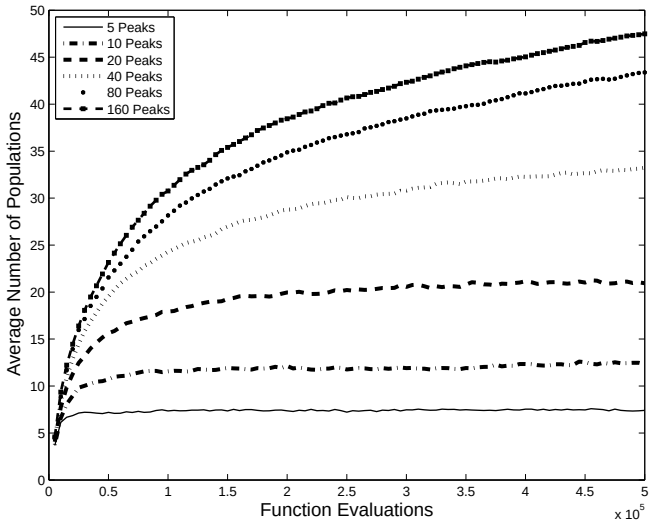

Fig. 4 Average number of populations versus function evaluations in the DynPopDE algorithm for various numbers of peaks.

8.3 Fluctuating Number of Peaks Experiments

The modified MPB described in Section 4 was used to model problems with a fluctuating number of peaks. For these experiments, the non-adaptive algorithms (DynDE and CDE) were given 10 populations each. DynDE was compared to DynPopDE for different values of maximum number of peaks and percentage change in number of peaks. Additionally, the effects of varying dimension and change periods were investigated. All combinations of the settings listed in Table 5 were investigated.

Table 5 MPB settings for fluctuating number of peaks experiments

Setting	**Values**
Number of Dimensions	5, 10, 15
Change period	1000, 2000, 3000, 4000, 5000
Maximum Number of Peaks	20, 40, 60, 80, 100, . . . , 180, 200
Percentage Change in Nr Peaks	10, 20, 30, 40, 50

The results for DynDE, CDE and DynPopDE along with the results of a Mann-Whitney U test to indicate statistical significant differences are summarized in Table 6. For the sake of brevity, the 10 and 15 dimensional results, and several of the other combinations investigated, are omitted, as similar trends were found.

CDE performed better than DynDE in all cases. The results of the Mann-Whitney U tests show that the difference between CDE and DynDE was statistically significant in 710 of the 750 experiments. On average, CDE yielded an improvement over DynDE of 23.5% in 5 dimensions, 32.14% in 10 dimensions and 34.17% in 15 dimensions.

Table 6 5 dimensional fluctuating numbers of peak experiments listing the maximum number of peaks (**#P**), percentage change in the number of peaks (**%C**), change period (**CP**), offline error of DynDE (**DynDE 10**), CDE (**CDE 10**) and DynPopDE, and Mann-Whitney U test p-values of DynDE compared to CDE (**p-A**), CDE compared to DynPopDE (**p-B**), and DynDE compared to DynPopDE (**p-C**) (Values that are below a 95% confidence level are in boldface).

#P	%C	CP	DynDE 10	CDE 10	DynPopDE	p-A	p-B	p-C
40	10	1000	16.70 ± 1.44	12.92 ± 1.06	10.68 ± 0.99	0.00	0.00	0.00
		2000	7.95 ± 0.83	6.45 ± 0.69	5.30 ± 0.51	0.01	0.01	0.00
		3000	5.09 ± 0.48	4.23 ± 0.45	3.94 ± 0.47	0.00	**0.16**	0.00
		4000	3.69 ± 0.29	3.58 ± 0.38	2.97 ± 0.32	**0.49**	0.00	0.00
		5000	3.07 ± 0.20	2.87 ± 0.27	2.38 ± 0.20	**0.09**	0.01	0.00
	50	1000	38.19 ± 1.22	30.86 ± 1.10	28.00 ± 0.88	0.00	0.00	0.00
		2000	22.50 ± 1.05	15.77 ± 0.79	14.87 ± 0.77	0.00	**0.09**	0.00
		3000	15.00 ± 1.18	10.89 ± 0.70	10.53 ± 0.69	0.00	**0.30**	0.00
		4000	10.21 ± 0.95	7.81 ± 0.57	7.70 ± 0.64	0.00	**0.71**	0.00
		5000	8.52 ± 0.89	6.90 ± 0.65	5.41 ± 0.58	0.01	0.00	0.00
80	10	1000	16.30 ± 1.70	12.20 ± 1.25	11.82 ± 1.19	0.00	**0.69**	0.00
		2000	8.87 ± 1.02	7.11 ± 0.79	6.56 ± 0.62	0.01	**0.42**	0.00
		3000	6.64 ± 0.80	5.08 ± 0.55	3.95 ± 0.34	0.00	0.00	0.00
		4000	4.56 ± 0.43	4.08 ± 0.30	3.26 ± 0.40	**0.13**	0.00	0.00
		5000	3.62 ± 0.22	3.34 ± 0.25	2.76 ± 0.29	0.01	0.00	0.00
	50	1000	37.94 ± 1.17	30.22 ± 0.96	28.09 ± 0.78	0.00	0.00	0.00
		2000	24.71 ± 1.58	16.69 ± 1.00	16.00 ± 0.83	0.00	**0.30**	0.00
		3000	17.12 ± 1.12	13.39 ± 1.02	11.75 ± 0.93	0.00	0.01	0.00
		4000	13.35 ± 0.96	9.51 ± 0.64	8.51 ± 0.72	0.00	0.03	0.00
		5000	9.83 ± 0.87	7.67 ± 0.63	7.60 ± 0.72	0.00	**0.98**	0.00
120	10	1000	16.62 ± 1.68	12.33 ± 1.22	12.37 ± 1.14	0.00	**0.81**	0.00
		2000	8.98 ± 1.15	6.85 ± 0.81	6.60 ± 0.73	0.00	**0.65**	0.00
		3000	6.19 ± 0.84	5.04 ± 0.53	4.39 ± 0.55	0.01	0.01	0.00
		4000	5.10 ± 0.51	4.30 ± 0.39	3.28 ± 0.31	0.01	0.00	0.00
		5000	4.26 ± 0.35	3.57 ± 0.22	3.00 ± 0.50	0.00	0.00	0.00
	50	1000	38.35 ± 1.27	30.54 ± 1.16	28.43 ± 0.99	0.00	0.01	0.00
		2000	23.39 ± 0.99	17.53 ± 0.92	17.10 ± 0.81	0.00	**0.97**	0.00
		3000	17.36 ± 0.97	12.86 ± 0.74	13.07 ± 0.89	0.00	**0.98**	0.00
		4000	13.55 ± 1.00	9.71 ± 0.69	9.51 ± 0.82	0.00	**0.56**	0.00
		5000	11.68 ± 0.98	8.12 ± 0.58	8.02 ± 0.65	0.00	**0.74**	0.00
160	10	1000	16.60 ± 1.55	12.48 ± 1.43	12.03 ± 1.22	0.00	**0.82**	0.00
		2000	8.83 ± 1.04	6.97 ± 0.88	6.89 ± 0.90	0.00	**0.70**	0.01
		3000	7.94 ± 1.29	5.52 ± 0.66	5.03 ± 0.57	0.01	**0.15**	0.00
		4000	5.04 ± 0.66	4.40 ± 0.44	3.72 ± 0.46	0.01	0.00	0.00
		5000	4.53 ± 0.46	4.34 ± 0.48	3.37 ± 0.61	**0.28**	0.00	0.00
	50	1000	36.32 ± 1.18	30.01 ± 1.08	29.31 ± 0.93	0.00	**0.14**	0.00
		2000	24.75 ± 0.95	18.13 ± 0.79	17.49 ± 0.93	0.00	**0.56**	0.00
		3000	18.71 ± 1.20	12.54 ± 0.81	12.14 ± 0.84	0.00	**0.71**	0.00
		4000	14.64 ± 1.29	11.08 ± 0.83	10.10 ± 0.74	0.00	**0.11**	0.00
		5000	12.58 ± 1.10	8.07 ± 0.80	8.15 ± 0.72	0.00	**0.56**	0.00
200	10	1000	16.34 ± 1.87	12.10 ± 1.36	11.81 ± 1.12	0.00	**0.98**	0.00
		2000	8.60 ± 1.05	6.78 ± 0.80	7.13 ± 0.82	0.00	**0.62**	**0.06**
		3000	7.54 ± 1.01	5.98 ± 0.68	5.47 ± 0.67	0.01	**0.21**	0.00
		4000	5.56 ± 0.66	4.31 ± 0.58	4.04 ± 0.64	0.00	0.03	0.00
		5000	4.33 ± 0.44	4.27 ± 0.44	3.43 ± 0.46	**0.66**	0.00	0.00
	50	1000	35.72 ± 0.98	29.28 ± 1.08	27.82 ± 1.18	0.00	**0.11**	0.00
		2000	22.83 ± 1.13	17.59 ± 0.74	18.27 ± 0.94	0.00	**0.22**	0.00
		3000	19.47 ± 1.22	12.88 ± 0.83	13.23 ± 1.02	0.00	**0.63**	0.00
		4000	15.15 ± 1.30	9.53 ± 0.65	10.33 ± 0.89	0.00	**0.19**	0.00
		5000	12.00 ± 0.96	8.36 ± 0.58	8.77 ± 0.72	0.00	**0.60**	0.00

The same experiments that were conducted with DynDE and CDE were repeated using DynPopDE. The relationship between the number of peaks and the number of populations in a typical run of the DynPopDE algorithm is illustrated in Figure 5. It is clear that the number of populations is increased when the number of peaks increases, but remains less than the number of peaks when the number of peaks is large. This

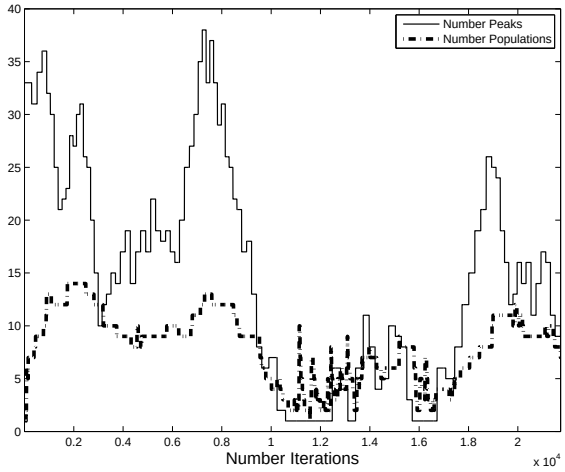

Fig. 5 Comparison between the number of peaks and the number of populations in a typical run of the DynPopDE algorithm

is consistent with results found in the unknown number of peaks experiments, which indicated that the number of populations should be kept relatively small when many peaks are present.

The results of the DynPopDE experiments are summarized in Table 6. DynPopDE yielded a clear improvement over DynDE, with 746 of the 750 experiments resulting in a statistically significantly better result for DynPopDE. On average, DynPopDE yielded an improvement over DynDE of 29.22% in 5 dimensions, 37.02% in 10 dimensions and 40.55% in 15 dimensions.

The difference between CDE and DynPopDE is less pronounced than between DynDE and CDE. Out of the 750 experiments only 296 cases showed a statistically significant difference between CDE and DynPopDE. However, of those 296 cases, 294 showed an improvement of DynPopDE over CDE. In many cases the improvement was over 20%. Thus, while it cannot be said that DynPopDE performs better that CDE in all cases, and no trend between improvement of DynPopDE over CDE and the parameters of the experiments is evident, there seems to be definite benefits to using DynPopDE rather than CDE.

9 Comparison with other approaches

This section validates DynPopDE by comparing it with. Test cases with unknown numbers of peaks are investigated in Section 9.1 and fluctuating numbers of peaks scenarios are investigated in Section 9.2.

9.1 Unknown number of peaks

The first algorithm to which DynPopDE is compared is that of Li et al. [19], who adapted the PSO with speciation algorithm of Parrot and Li [27] to create an algo-

rithm designed specifically for situations where the number of peaks is unknown. This algorithm is called the Speciation-based PSO (SPSO). The second algorithm used is the self-adapting multi-swarm algorithm (referred to here as SAMS) of Blackwell [2]. The third algorithm used for comparison is the CellularDE algorithm of Noroozi et al. [25].

The creators of the above algorithms used Scenario 2 of the MPB with various settings of the number of peaks to validate their results. The same experiments were repeated in this section using DynPopDE. The best results reported by Li et al. [19], Blackwell [2] and Noroozi et al. [25] are given in Table 7 with the corresponding results of DynPopDE.

Table 7 Offline error of SPSO, SAMS, CellularDE and DynPopDE

# Peaks	SPSO	SAMS	CellularDE	DynPopDE
3	2.14 ± 0.02			0.86 ± 0.06
5	1.98 ± 0.01		1.50 ± 0.04	1.03 ± 0.13
8	1.89 ± 0.02			1.24 ± 0.07
10	1.98 ± 0.01	1.77 ± 0.01	1.64 ± 0.03	1.39 ± 0.07
12	2.39 ± 0.02			1.41 ± 0.10
15	2.78 ± 0.02			1.54 ± 0.09
18	2.94 ± 0.02			1.65 ± 0.06
50	3.47 ± 0.02		2.75 ± 0.05	2.10 ± 0.06
100	3.6 ± 0.02		2.73 ± 0.03	2.34 ± 0.05
200	3.47 ± 0.02	2.37 ± 0.01	2.61 ± 0.02	2.44 ± 0.05
300	3.12 ± 0.01			2.32 ± 0.04

It is clear that DynPopDE performed considerably better than both SPSO and CellularDE on all the test cases investigated, and performed better than SAMS on 10 peaks. SAMS gave a slightly lower offline error than DynPopDE when 200 peaks were present in the environment. It should be pointed out, however, that SAMS has a parameter (n_{excess}) that is problem dependant and must be manually tuned. When the default setting of $n_{excess} = 1$ was used, SAMS yielded an offline error of 2.54, which is higher than the result found when using DynPopDE. In practice it may be more feasible to use DynPopDE rather than SAMS in order to avoid manually tuning parameters.

Despite the previously mentioned disadvantage associated with the SAMS algorithm, the reported results suggest that potentially the swarm spawning and removing approach followed by Blackwell [2] could be more effective than the technique used in DynPopDE. This possibility was investigated by incorporating the approach followed in SAMS to spawn and remove populations into CDE. A population is thus created when there are no available free populations, i.e. all the current populations had converged to within a diameter of $2r_{conv}$, were r_{conv} is calculated using the same equation as r_{excl} (equation 5). The worst performing free population is removed when the number of free populations exceed n_{excess}. The new algorithm is referred to as SAMSCDE. Experiments were conducted on the MPB using the Scenario 2 settings for various numbers of peaks. Nine values of the n_{excess} parameter were tested to determine the influence of this parameter. Results are given in Table 8, along with outcomes of Mann-Whitney U tests comparing the results of SAMSCDE to DynPopDE.

Table 8 Offline error of SAMSCDE for various settings of n_{excess} and number of peaks (**#P**).

#P	Error	p-val	Error	p-val	Error	p-val
	$n_{excess} = 1$		$n_{excess} = 2$		$n_{excess} = 4$	
5	47.82 ± 2.16	0.00	45.25 ± 2.12	0.00	39.43 ± 2.24	0.00
10	39.26 ± 1.51	0.00	35.43 ± 1.14	0.00	26.75 ± 0.97	0.00
20	32.87 ± 0.74	0.00	29.23 ± 0.63	0.00	19.52 ± 0.52	0.00
40	27.27 ± 0.46	0.00	22.86 ± 0.36	0.00	11.70 ± 0.42	0.00
80	22.35 ± 0.29	0.00	16.37 ± 0.32	0.00	5.22 ± 0.32	0.00
160	17.50 ± 0.26	0.00	8.92 ± 0.31	0.00	3.42 ± 0.15	0.00
	$n_{excess} = 8$		$n_{excess} = 16$		$n_{excess} = 32$	
5	18.35 ± 1.25	0.00	4.98 ± 0.28	0.00	4.37 ± 0.22	0.00
10	11.48 ± 0.82	0.00	4.09 ± 0.13	0.00	4.08 ± 0.16	0.00
20	5.68 ± 0.35	0.00	3.76 ± 0.13	0.00	3.70 ± 0.10	0.00
40	3.62 ± 0.10	0.00	3.29 ± 0.10	0.00	3.45 ± 0.11	0.00
80	3.32 ± 0.11	0.00	3.26 ± 0.11	0.00	3.17 ± 0.12	0.00
160	3.11 ± 0.16	0.00	3.28 ± 0.15	0.00	3.18 ± 0.14	0.00
	$n_{excess} = 64$		$n_{excess} = 128$		$n_{excess} = 256$	
5	4.51 ± 0.18	0.00	4.62 ± 0.20	0.00	4.61 ± 0.23	0.00
10	4.18 ± 0.16	0.00	4.12 ± 0.12	0.00	4.14 ± 0.16	0.00
20	3.68 ± 0.10	0.00	3.73 ± 0.10	0.00	3.76 ± 0.11	0.00
40	3.33 ± 0.09	0.00	3.37 ± 0.09	0.00	3.37 ± 0.11	0.00
80	3.20 ± 0.11	0.00	3.11 ± 0.11	0.00	3.16 ± 0.11	0.00
160	3.25 ± 0.14	0.00	3.19 ± 0.16	0.00	3.20 ± 0.13	0.00

The experimental results indicate that SAMSCDE works best when a relatively large value of n_{excess} is used (between 32 and 64) as opposed to Blackwell's results on SAMS where values of 3 and 4 gave the best results. This is a consequence of CDE evaluating populations in sequence rather than in parallel. It thus appears as if many free populations are present for a large number of iterations, simply because some of the populations have not been allowed to evolve yet. A larger n_{excess} threshold is consequently required to prevent the unnecessary removal of populations.

The SAMSCDE experiments in which larger values of n_{excess} was used yielded reasonable results. However, DynPopDE performed statistically significantly better in all experiments. It can thus be concluded that, while Blackwell's swarm spawning and removing approach worked well in SAMS, it is not an effective technique to use in conjunction with CDE on problems where the number of peaks is unknown.

9.2 Fluctuating numbers of peaks

The SAMSCDE algorithm that was discussed in the previous section was evaluated using the modified MPB to simulate problems where the number of peaks fluctuate. Several settings for n_{excess} were tested. Table 9 lists results when using the Scenario 2 MPB settings for various values for maximum number of peaks and percentage change in the number of peaks, along with outcomes of Mann-Whitney U tests comparing the results of SAMSCDE to DynPopDE.

Once again, it was found that a relatively large value of n_{excess} works best. A comparison of the results of Tables 6 and 9 show that DynPopDE performed considerably better than SAMSCDE on problems where the number of peaks fluctuate. In all cases the results were statistically significantly different. It can thus be concluded that the population spawning an removing technique employed by SAMS is not appropriate for incorporation into CDE.

Table 9 Offline error of SAMSCDE for various settings of n_{excess}, maximum number of peaks (**#P**) and percentage change in the number of peaks (**%C**).

#P	% C	Error	p-val	Error	p-val	Error	p-val
		$n_{excess} = 1$		$n_{excess} = 2$		$n_{excess} = 4$	
40	10	30.81 ± 1.41	0.00	26.79 ± 1.61	0.00	20.09 ± 2.07	0.00
	50	32.32 ± 0.96	0.00	31.73 ± 1.18	0.00	27.45 ± 1.28	0.00
80	10	27.18 ± 1.51	0.00	22.80 ± 1.73	0.00	16.17 ± 2.01	0.00
	50	31.12 ± 1.16	0.00	29.25 ± 1.12	0.00	25.77 ± 1.36	0.00
120	10	23.49 ± 1.49	0.00	18.95 ± 1.75	0.00	11.94 ± 1.77	0.00
	50	28.78 ± 1.12	0.00	27.28 ± 1.62	0.00	23.23 ± 1.28	0.00
160	10	23.80 ± 1.37	0.00	17.88 ± 1.91	0.00	11.72 ± 2.23	0.00
	50	28.89 ± 1.51	0.00	25.43 ± 1.32	0.00	22.68 ± 1.54	0.00
200	10	22.05 ± 1.63	0.00	16.77 ± 1.98	0.00	10.70 ± 1.68	0.00
	50	26.47 ± 1.04	0.00	25.19 ± 1.18	0.00	23.14 ± 1.40	0.00
		$n_{excess} = 8$		$n_{excess} = 16$		$n_{excess} = 32$	
40	10	12.16 ± 1.67	0.00	11.27 ± 1.89	0.00	10.06 ± 1.64	0.00
	50	23.16 ± 1.45	0.00	22.83 ± 1.61	0.00	22.69 ± 1.42	0.00
80	10	10.52 ± 1.83	0.00	9.42 ± 1.50	0.00	12.26 ± 2.10	0.00
	50	22.81 ± 1.45	0.00	22.47 ± 1.32	0.00	21.71 ± 1.28	0.00
120	10	9.06 ± 1.79	0.00	8.30 ± 1.47	0.00	8.46 ± 1.49	0.00
	50	21.38 ± 1.43	0.00	20.93 ± 1.28	0.00	21.10 ± 1.43	0.00
160	10	9.78 ± 1.92	0.00	8.56 ± 1.78	0.00	8.43 ± 1.74	0.00
	50	21.03 ± 1.17	0.00	22.28 ± 1.24	0.00	21.46 ± 1.50	0.00
200	10	8.82 ± 1.61	0.00	7.97 ± 1.63	0.00	8.26 ± 1.64	0.00
	50	21.34 ± 1.51	0.00	21.16 ± 1.28	0.00	21.02 ± 1.21	0.00
		$n_{excess} = 64$		$n_{excess} = 128$		$n_{excess} = 256$	
40	10	11.12 ± 1.84	0.00	9.55 ± 1.68	0.00	9.35 ± 1.81	0.00
	50	21.22 ± 1.57	0.00	21.28 ± 0.94	0.00	23.43 ± 1.38	0.00
80	10	9.32 ± 1.93	0.00	10.85 ± 1.98	0.00	9.52 ± 1.52	0.00
	50	22.83 ± 1.67	0.00	22.62 ± 1.35	0.00	22.31 ± 1.47	0.00
120	10	7.77 ± 1.25	0.00	10.66 ± 1.87	0.00	9.21 ± 1.60	0.00
	50	21.58 ± 1.07	0.00	22.11 ± 1.47	0.00	22.40 ± 1.38	0.00
160	10	9.71 ± 2.07	0.00	10.03 ± 2.09	0.00	10.16 ± 2.00	0.00
	50	21.69 ± 1.53	0.00	22.31 ± 1.57	0.00	21.21 ± 1.37	0.00
200	10	8.65 ± 1.76	0.00	9.15 ± 1.70	0.00	11.08 ± 2.08	0.00
	50	21.39 ± 1.22	0.00	21.20 ± 1.40	0.00	21.92 ± 1.12	0.00

The 2009 Congress on Evolutionary Computation (CEC2009) ran a dynamic optimization competition. This competition used the Generalized Benchmark Generator (GBG) of Li et al. [18], [17] to compare the results of the competing algorithms. The winning algorithm of this competition was *jDE*, developed by Brest et al. [8]. *jDE* was implemented by the authors of this article in order to compare its performance with the algorithms discussed in this paper.

Du Plessis and Engelbrecht [10] showed that, while *jDE* delivers superior results in low-change frequency problems, it is outperformed by CDE in high-change frequency problems. *jDE* was not specifically developed for problems where the number of optima fluctuate, but the ageing mechanism that is employed (see Section 2) allows further exploration once a population has converged to an optimum. The results of *jDE* on the fluctuating number of peaks experiments are given in Table 10.

A comparison of Tables 6 and 10 shows that DynPopDE performed better in all cases.

10 Discussion and Conclusions

This paper evaluated the performance of DynDE and CDE on dynamic environments where the number of optima is unknown or is fluctuating. A new algorithm, DynPopDE,

Table 10 jDE results for 5 dimensional fluctuating numberd of peak experiments listing the maximum number of peaks (**Max # Peaks**), percentage change in the number of peaks (**% Change**), change period (**Change Period**) and offline error of jDE (**jDE**)

Max # Peaks	% Change	Change Period	jDE
40	10	1000	19.59 ± 1.44
		2000	12.90 ± 0.83
		3000	10.40 ± 0.48
		4000	8.10 ± 0.29
		5000	7.62 ± 0.20
	50	1000	32.87 ± 1.22
		2000	23.21 ± 1.05
		3000	16.75 ± 1.18
		4000	13.86 ± 0.95
		5000	10.55 ± 0.89
80	10	1000	17.64 ± 1.70
		2000	12.79 ± 1.02
		3000	9.91 ± 0.80
		4000	8.83 ± 0.43
		5000	7.36 ± 0.22
	50	1000	31.45 ± 1.17
		2000	22.13 ± 1.58
		3000	17.09 ± 1.12
		4000	13.75 ± 0.96
		5000	10.87 ± 0.87
120	10	1000	17.45 ± 1.68
		2000	14.18 ± 1.15
		3000	9.90 ± 0.84
		4000	8.83 ± 0.51
		5000	7.52 ± 0.35
	50	1000	30.76 ± 1.27
		2000	22.52 ± 0.99
		3000	17.18 ± 0.97
		4000	13.92 ± 1.00
		5000	11.59 ± 0.98
160	10	1000	17.15 ± 1.55
		2000	12.42 ± 1.04
		3000	11.02 ± 1.29
		4000	9.54 ± 0.66
		5000	7.99 ± 0.46
	50	1000	29.68 ± 1.18
		2000	22.47 ± 0.95
		3000	16.28 ± 1.20
		4000	13.37 ± 1.29
		5000	11.67 ± 1.10
200	10	1000	16.63 ± 1.87
		2000	12.91 ± 1.05
		3000	11.31 ± 1.01
		4000	8.65 ± 0.66
		5000	7.39 ± 0.44
	50	1000	29.68 ± 0.98
		2000	22.85 ± 1.13
		3000	16.98 ± 1.22
		4000	14.21 ± 1.30
		5000	11.05 ± 0.96

specifically aimed at this type of dynamic optimization problem was proposed and evaluated.

The first result of this investigation is the fact that maintaining an equal number of populations as the number of optima is not an effective strategy. Too many populations reduce the number of generations that any specific population can complete before a change in the environment occurs. Consequently, the effectiveness of the optimization algorithm is reduced. When a large number of optima are present, a relatively small number of populations should be used.

The results indicated that the competitive population evaluation approach of CDE resulted in an algorithm that is more robust than DynDE when faced with an unknown number of peaks. The population spawning and removing process of DynPopDE yielded a further improvement in results. It was shown that both DynPopDE and CDE performed better than DynDE in situations where the number of optima is unknown on almost all of the cases investigated. Experiments to investigate the introduction of a penalty factor into DynPopDE's base algorithm, CDE, showed that the penalty factor reduced the effectiveness of CDE. However, when introduced into DynPopDE, it was shown that the penalty factor aided the population-spawning process and consequently improved DynPopDE.

DynPopDE performed better than CDE in the 5 dimensional unknown number of peaks experiments where the number of peaks were more than 20. This trend continued to the higher dimensions, although the improvement of DynPopDE over CDE was not always statistically significant.

For problems where the number of optima fluctuates during the course of the optimization process, results showed that CDE and DynPopDE clearly outperforms DynDE. In experiments comparing DynPopDE to CDE it was found that the majority of the results were not statistically significantly different. DynPopDE did outperform CDE in virtually all cases where a statistical significant difference was found. It can thus be concluded that DynPopDE is the superior algorithm.

Comparisons with SPSO and *jDE*, two state-of-the-art optimization algorithms for dynamic environments, illustrated the superiority of DynPopDE on all unknown and fluctuating optima-problems investigated. Although DynPopDE did not perform better than SAMS when a large number of peaks were used in an unknown number of peaks problem, it was argued that DynPopDE may be the preferred algorithm since it has fewer parameters to tune. Experimental evidence showed that the population spawning and removing technique used in SAMS were not effective when used in conjunction with CDE.

DynPopDE does not depend on any intrinsic DE behaviour, since its main components involve the creation of independent populations which competitively optimize different peaks. Future work could include a study the effects of applying aspects of DynPopDE to other multi-population optimization algorithms for dynamic environments.

References

1. T. M. Blackwell and P. J. Bentley. Dynamic search with charged swarms. In W. B. Langdon et al., editor, *Genetic and Evolutionary Computation Conference*, pages 19–26. Morgan Kaufmann, 2002.
2. T.M. Blackwell. Particle swarm optimization in dynamic environments. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 29–49. Springer, 2007.
3. T.M. Blackwell and J. Branke. Multiswarm optimization in dynamic environments. *Applications of Evolutionary Computing*, 3005:489–500, 2004.
4. T.M. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.
5. J. Branke. *The moving peaks benchmark.* http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/, June 2007.
6. J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. In S. Tsutsui and A. Ghosh, editors, *Advances in evolutionary computing: theory and applications*, pages 239–262, Springer-Verlag New York, 2003.
7. J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, Dec. 2006.
8. J. Brest, A. Zamuda, B. Boškovic, M. S. Maučec, and V. Žumer. Dynamic optimization using self-adaptive differential evolution. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 415–422, Piscataway, NJ, USA, 2009. IEEE Press.
9. H.G. Cobb. *An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments.* Navy Center for Applied Research in Artificial Intelligence, Technical Report AIC-90-001, 1990.
10. M.C. du Plessis and A.P. Engelbrecht. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *Submitted to European Journal of Operational Research*, 2010.
11. J. J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Congress on Evolutionary Computation*, volume 3, pages 2031–2038. IEEE, 1999.
12. X. Hu and R.C. Eberhart. Adaptive particle swarm optimisation: detection and response to dynamic systems. In *Congress on Evolutionary Computation*, pages 1666–1670, 2002.
13. S. Janson and M. Middendorf. A hierachical particle swarm optimizer. In *Congress on Evolutionary Computation*, pages 770–776. IEEE, 2003.
14. S. Janson and M. Middendorf. A hierachical particle swarm optimizer for dynamc optimization problems. In G. R. Raidl, editor, *Applications of evolutionary computing*, volume 3005 of *LNCS*, pages 513–524. Springer, 2004.
15. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.
16. J. Lampinen K. Price, R. Storn. *Differential evolution - A practical approach to global optimization.* Springer, 2005.
17. C. Li and S. Yang. A generalized approach to construct benchmark problems for dynamic optimization. In *SEAL '08: Proceedings of the 7th International Conference on Simulated Evolution and Learning*, pages 391–400, Berlin, Heidelberg, 2008. Springer-Verlag.
18. C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H. g. Beyer, and P. N. Suganthan. *Benchmark Generator for CEC2009 Competition on Dynamic Optimization.* University of Leicester, University of Birmingham, Nanyang Technological University, Technical Report, 2008.
19. X. Li, J. Branke, and T.M. Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 51–58, New York, NY, USA, 2006. ACM.
20. X. Li and K. H. Dam. Comparing particle swarms for tracking extrema in dynamic environments. In *Congress on Evolutionary Computation*, volume 3, pages 1772– 1779. IEEE, 2003.
21. R. Mendes and A. Mohais. DynDE: a differential evolution for dynamic optimization problems. In *Congress on Evolutionary Computation*, pages 2808–2815. IEEE, 2005.
22. N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, number 1498 in LNCS, pages 149–158. Springer, 1998.

23. R.W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Springer, 2004.
24. I. Moser and T. Hendtlass. A simple and efficient multi-component algorithm for solving dynamic function optimisation problems. In *Congress on Evolutionary Computation*, pages 252–259. IEEE, 2007.
25. V. Noroozi, A.B. Ali and M.R. Meybodi. CellularDE: a cellular based differential evolution for dynamic optimization problems. In *Proceedings of the 10th international conference on Adaptive and natural computing algorithms - Volume Part I*, pages 340–349. Springer-Verlag, 2011.
26. F. Oppacher and M. Wineberg. The shifting balance genetic algorithm: Improving the GA in a dynamic environment. In Wolfgang Banzhaf et al., editor, *Genetic and Evolutionary Computation Conference (GECCO)*, volume 1, pages 504 – 510. Morgan Kaufmann, San Francisco, 1999.
27. D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In *Congress on Evolutionary Computation*, pages 98–103. IEEE, 2004.
28. R. Storn. On the usage of differential evolution for function optimization. In *Biennial Conference of the North American Fuzzy Information Processing Society*, pages 519–523. IEEE, 1996.
29. R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.
30. K. Trojanowski. 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*, pages 143–148. CISIM '07, 2007.
31. R. K. Ursem. Multinational GA optimization techniques in dynamic environments. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Genetic and Evolutionary Computation Conference*, pages 19–26. Morgan Kaufmann, 2000.
32. F. Vavak, K. Jukes, and T. C. Fogarty. Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search. In T. Bäck, editor, *International Conference on Genetic Algorithms*, pages 719–726. Morgan Kaufmann, 1997.
33. S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1115–1122, New York, NY, USA, 2005. ACM.
34. D. Zaharie and F. Zamfirache. Diversity enhancing mechanisms for evolutionary optimization in static and dynamic environments. In *3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence*, pages 460–471, 2006.