European Journal of Operational Research 218 (2012) 7–20

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Continuous Optimization

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

ARTICLE INFO

Article history:
Received 9 May 2010
Accepted 27 August 2011
Available online 6 September 2011

Keywords:
Differential evolution
Evolutionary computation
Dynamic environments
Optimization

ABSTRACT

This paper proposes two adaptations to DynDE, a differential evolution-based algorithm for solving dynamic optimization problems. The first adapted algorithm, Competitive Population Evaluation (CPE), is a multi-population DE algorithm aimed at locating optima faster in the dynamic environment. This adaptation is based on allowing populations to compete for function evaluations based on their performance. The second adapted algorithm, Reinitialization Midpoint Check (RMC), is aimed at improving the technique used by DynDE to maintain populations on different peaks in the search space. A combination of the CPE and RMC adaptations is investigated. The new adaptations are empirically compared to DynDE using various problem sets. The empirical results show that the adaptations constitute an improvement over DynDE and compares favorably to other approaches in the literature. The general applicability of the adaptations is illustrated by incorporating the combination of CPE and RMC into another Differential Evolution-based algorithm, *jDE*, which is shown to yield improved results.

1. Introduction

Dynamic optimization problems are found in many real world domains, for example traffic control, polarization mode dispersion compensation in optical fibre, and target tracking in military applications (Gibbon et al., 2008; Li et al., 2006). 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 does 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 is increased.

In this paper, adaptations to the above algorithm are suggested. The first adaptation is a novel algorithm 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 the Reinitialization Midpoint Check (RMC) algorithm. RMC is an improvement on the technique used in DynDE to prevent populations from converging to the same peak. CPE and RMC are combined to form a third adapted algorithm, referred to as Competing Differential Evolution (CDE). The general applicability of the CPE and RMC adaptations is illustrated by incorporation into *jDE*, another optimization algorithm for dynamic environments (Brest et al., 2009).

The rest of the paper is structured as follows: Related work is discussed in Section 2. The DE algorithm is discussed in Section 3. Section 4 reviews the research of Mendes and Mohais (2005). CPE is presented in Section 5, RMC in Section 6 and the combination of CPE and RMC (CDE) is described in Section 7. The benchmark which is used in this work is described in Section 8. The results of a comparative investigation into the scalability of each of these adaptations is given in Section 9. The new adapted algorithms are compared with other research in Section 10. The general applicability of the adaptations proposed in this paper is demonstrated in Section 11 by incorporating the novel sub-components of CDE into another optimization algorithm. Conclusions are drawn in Section 12.

2. Related work

Several of the earlier investigations into optimization in dynamic environments involved approaches based on genetic algorithms (GA) (Cobb, 1990; Vavak et al., 1997; Mori et al., 1997).

* Corresponding author.
E-mail addresses: mc.duplessis@nmmu.ac.za (M.C. du Plessis), engel@cs.up.ac.za (A.P. Engelbrecht).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.08.031

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 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 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). The principal of using Quantum or Brownian individuals is similar to the DE adaptations suggested by Kaelo and Ali (2006) where the region around an offspring individual is explored before the individual is inserted into the population.

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.

Angira and Santosh (2007) used the trigonometric DE approach of Fan and Lampinen (2003) to optimize dynamic systems. The standard DE mutation operator is adapted to use as the target vector the average of three randomly selected vectors. The three weighted vector differentials are then added to the target vector. This approach was extended in (Angira and Santosh, 2008) so that offspring individuals are inserted into the population immediately, as opposed to at the end of the generation.

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, called speciation. The social component of PSO provides a simple method to divide the population into sub-populations. In this algorithm, a particle is classed into a particular sub-population if the Euclidean distance between the position of the particle and the best particle in the particular sub-population (referred to as the species seed) is within a certain threshold value. Species seeds are determined by processing a list of all particles sorted in descending order of fitness. The set of species seeds is constructed by adding each particle encountered in the list that is not within the threshold distance from any of the particles already in the set of species seeds (Parrott et al., 2006). Should a particle be located within the threshold distance of more than one sub-population best, it is assigned to the sub-population with the fittest best particle. The *global best* value of each particle within a sub-population is set to the *personal best* value of the species seed. Particles can thus migrate to another sub-population by moving too far away from the current sub-population's best particle and 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.

An approach that uses both DE and PSO populations was proposed by Lung and Dumitrescu (2007). This technique is called Collaborative Evolutionary-Swarm Optimization (CESO). A population optimized using PSO refines the global optimum, while a DE variant, called Crowding Differential Evolution, is used to maintain diversity and locate local optima. Crowding DE (Thomsen, 2004) is an algorithm aimed at locating multiple optima in static environments. Crowding DE changes the offspring operator of DE so that the most similar individual in the parent population is replaced, rather than the parent. This idea is similar to an approach used in niching, where the fitness of an individual is divided by a sharing function which is proportional to the number of other individuals within a threshold distance of the individual (Sareni and Krahenbuhl, 1998). In CESO, the individuals in the PSO population are replaced by the individuals in the DE population when a change occurs in the environment or when the distance between the best individuals within the two populations drops below a threshold value.

CESO was improved by incorporating a second DE population (Lung and Dumitrescu, 2010). The new algorithm is called Evolutionary Swarm Cooperative Algorithm (ESCA). The second DE population acts as a memory of previously found optima for the first DE population.

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 from converging to the same optimum. An aging 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 in 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. One of the sub-populations is generated by selecting random individuals from the archive and adding small random numbers to each of the individuals' components.

Other approaches to self-adapting or eliminating control parameters in DE include self-adaptive differential evolution (Salman et al., 2007) and Barebones DE (Omran et al., 2009).

The most successful non-population-based algorithm for dynamic environments 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.

Optimization in dynamic environments are related to the field of noisy optimization since in both cases the fitness function does not remain constant during the entire run of the algorithm. DE-based approaches have been suggested for noisy optimization problems, where parent individuals are probabilistically replaced with offspring individuals of inferior fitness. In Das et al. (2005) this was achieved by calculating the probability of offspring replacing parents as the ratio of the fitness of the parent over that of the offspring. Liu et al. (2008) utilized an exponentially decaying probability function that initially gives offspring a high probability of replacing the parent individual. As the run continues, this probability decreases until parents are replaced only if offspring have superior fitness.

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 base 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 (Price and Storn, 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 $\vec{u}_i$ by calculating each $u_{i,j}$ as:

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

where $u_{i,j}$ is the jth component of vector $\vec{u}_i$, $U(0,1)$ is a uniform random number between 0 and 1, $C_r \in (0,1)$ 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 an equal or better fitness value than $\vec{x}_i$ then replace $\vec{x}_i$ with $\vec{u}_i$.
(v) Repeat steps (iii) and (iv) 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 Eq. (1)), and how offspring are created (see Eq. (2)). By convention, schemes are labeled in the form DE/a/b/c, where a is the method used to select the base 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. When extending DynDE (see Sections 5 and 6) all settings were implemented as described in the following subsections.

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 desirable to keep track of the movements of all the peaks found in the problem space, since an entirely different peak may become the optimal peak once a change in the environment occurs. An effective method to track multiple peaks 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 sub-populations converge to different peaks. 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 threshold is calculated as follows:

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

where X is the range of the d dimensions (assuming equal ranges for all dimensions), and p is the number of peaks. Using Eq. (3), the exclusion threshold increases with an 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. Each of 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. Individuals are replaced even if the newly created individual has a higher error value. A Brownian individual, $\vec{x}_{\text{brown}}$, is thus created from the best individual within the sub-population, $\vec{x}_{\text{best}}$, using the formula:

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

where v is the standard deviation of the Gaussian distribution. The optimal value of v is problem dependant, since v controls the amount of diversity that is injected, however, Mendes and Mohais (2005) showed that an effective value of v is 0.2. Two Brownian individuals were used with a sub-population size of six and five Brownian individuals were used with a sub-population size of 10, as these values were experimentally found to work well.

4.4. DE scheme

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

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

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. Competitive population evaluation

This section presents a novel adaptation to improve DynDE and the motivations behind it. 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 convergence 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 quickly as possible after changes in the environment have occurred.

The Moving Peaks Benchmark (see Section 8) 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 environment, since it would yield a lower error 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. This argument is the motivation for a novel adapted algorithm named Competitive Population Evaluation (CPE). The primary goal of the new adaptation 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 adaptation 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. CPE thus differs from DynDE in that peaks are not discovered 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 Eq. (6)).
(ii) Calculate the performance value, $\mathcal{P}$ (see Eq. (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 (ii).
(vi) Return to step (i) 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

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

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:

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

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 Eq. (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 value of $\Delta f_k(t)$ is 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.

6. 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 the two populations reside on distinct peaks and that neither should be reinitialized (see Fig. 1, scenario A).

This adapted algorithm will be referred to as the Reinitialization Midpoint Check (RMC) algorithm. It is apparent that this strategy will not work in all cases. Scenarios B and C of Fig. 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.

RMC is similar, but simpler, than *hill-valley detection* suggested by Ursem (2000). The midpoint check adaptation 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. 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 Competing Differential Evolution (CDE). CDE 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. The pseudo-code for CDE is thus as follows:

Algorithm 1. CDE Pseudo-code

```
Create 10 sub-populations 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 P
    Create offspring using DE/best/2/bin for sub-population
   with highest value of P
    Insert better performing offspring into sub-population
  end
  Perform exclusion subject to RMC
  Create Brownian individuals
  count := count + 1
end
```

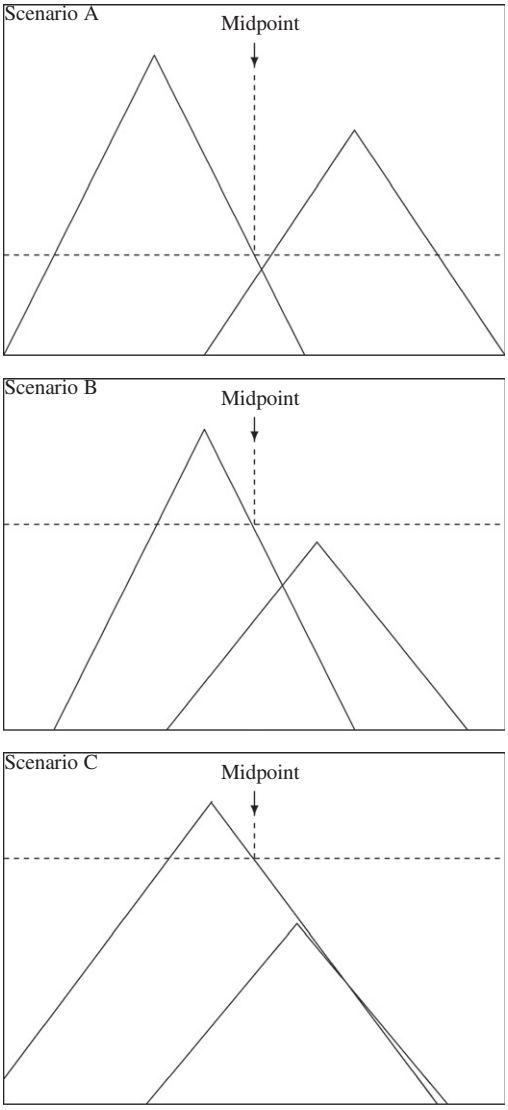

Fig. 1. Midpoint checking scenarios.

8. Moving peaks benchmark

Branke (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 moving peaks function is a maximization problem.

The performance measure suggested by Branke and Schmeck (2003) is the *offline error* which 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.

9. Experimental results

9.1. Experimental procedure

The following sections present the empirical results comparing the proposed algorithms to the base algorithm, DynDE. DynDE was implemented as described in Section 4. A value of 0.5 was used

for both $\mathcal{F}$ and C_r in accordance with best results found by Mendes and Mohais (2005). The MPB settings corresponding to Scenario 2 (Branke, 2007) are given in Table 1 (*default values*). In addition, the scalability of the algorithms in terms of change severity, change period, number of dimensions and different peak functions were investigated, by repeating the experiments for all combinations of the default settings with the *other tested* values listed in Table 1.

Furthermore, the effect of population size was investigated for sub-population size six (two Brownian individuals) and 10 (five Brownian individuals). Experiments were conducted on both the Spherical and Conical peak functions. The most apparent difference between the two functions 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 function 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 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 five 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 default MPB settings in Table 1, an offline error of 1.28 ± 0.09 was found for experiments with a population size of six and two Brownian individuals. Experiments with a population size of 10 and five 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 implementation 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.

The results of the scalability study of DynDE with respect to the various combinations of settings in Table 1 are summarized in Tables 2 and 3. For the sake of brevity, only the results for population size 6 in five 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 Figs. 2 and 4 for change severity values of 1.0 and 5.0, respectively. The same results for the spherical peak function can be seen in Figs. 3 and 5. 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 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.

Table 1
MPB settings.

Setting	Default value	Other tested
Number of dimensions	5	10, 15, 20, 25
Number of peaks	10	
Max and Min Peak height	[30,70]	
Max and Min Peak width	[1.0,12.0]	
Change period	5000	1000, 2000, 3000, 4000
Change severity	1.0	3.0, 5.0
Height severity	7.0	
Width severity	1.0	
Peak function	Cone	Sphere
Correlation	[0.0,1.0]	

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 2 and 3. 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 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. Figs. 6 and 7 graphicly depict the percentage improvement of CPE over DynDE when using the conical function with change frequencies of 1.0 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.

9.4. RMC results

The results of experiments conducted to investigate the effectiveness of the RMC adaptation are listed in Tables 2 and 3. The

Table 2
Offline error using conical peak function. Change Severity (**CS**), Change Period (**CP**), Percentage Improvement over DynDE (**% Imp**) and Mann–Whitney *U*-test result (*p*-value). Results in bold are not statistically significant.

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

Table 3
Offline error using spherical peak function. Change Severity (**CS**), Change Period (**CP**), Percentage Improvement over DynDE (**% Imp**) and Mann–Whitney *U*-test result (*p*-value). Results in bold are not statistically significant.

DynDE			RMC			CPE			CDE		
CS	CP	Offline error	Offline error	% Imp	*p*-Value	Offline error	% Imp	*p*-Value	Offline error	% Imp	*p*-Value
1	1000	1.66 ± 0.14	1.30 ± 0.07	21.46	0.00	1.31 ± 0.10	21.29	0.00	1.12 ± 0.08	32.37	0.00
1	2000	1.20 ± 0.10	1.00 ± 0.08	16.87	0.01	1.06 ± 0.10	11.66	**0.05**	0.85 ± 0.08	29.16	0.00
1	3000	1.03 ± 0.11	0.86 ± 0.07	16.3	0.02	0.90 ± 0.08	12.14	**0.15**	0.74 ± 0.07	28.24	0.00
1	4000	1.04 ± 0.09	0.76 ± 0.07	27.27	0.00	0.95 ± 0.10	9.13	**0.10**	0.75 ± 0.08	28.34	0.00
1	5000	0.91 ± 0.10	0.74 ± 0.07	18.75	0.01	0.94 ± 0.10	−3.24	**0.61**	0.66 ± 0.08	26.92	0.00
3	1000	9.66 ± 1.03	9.58 ± 1.00	0.84	**0.80**	4.16 ± 0.10	56.87	0.00	3.61 ± 0.08	62.63	0.00
3	2000	3.27 ± 0.10	2.89 ± 0.05	11.75	0.00	2.42 ± 0.09	25.95	0.00	2.1 ± 0.06	35.71	0.00
3	3000	2.44 ± 0.10	2.15 ± 0.06	11.83	0.00	1.85 ± 0.09	24.18	0.00	1.59 ± 0.07	34.68	0.00
3	4000	1.91 ± 0.07	1.76 ± 0.07	7.71	0.00	1.55 ± 0.09	18.58	0.00	1.29 ± 0.06	32.25	0.00
3	5000	1.70 ± 0.08	1.53 ± 0.07	10.12	0.00	1.34 ± 0.08	21.04	0.00	1.15 ± 0.07	32.33	0.00
5	1000	54.49 ± 11.02	65.70 ± 11.65	−20.58	**0.59**	15.14 ± 1.21	72.21	0.00	14.92 ± 1.15	72.62	0.00
5	2000	9.29 ± 0.16	8.93 ± 0.18	3.81	0.00	5.89 ± 0.11	36.61	0.00	5.54 ± 0.06	40.30	0.00
5	3000	6.13 ± 0.11	5.84 ± 0.06	4.69	0.00	4.12 ± 0.07	32.69	0.00	3.95 ± 0.06	35.49	0.00
5	4000	4.79 ± 0.10	4.56 ± 0.07	4.67	0.00	3.22 ± 0.07	32.83	0.00	3.08 ± 0.06	35.65	0.00
5	5000	3.95 ± 0.07	3.83 ± 0.09	3.24	0.00	2.80 ± 0.07	29.22	0.00	2.61 ± 0.06	34.03	0.00

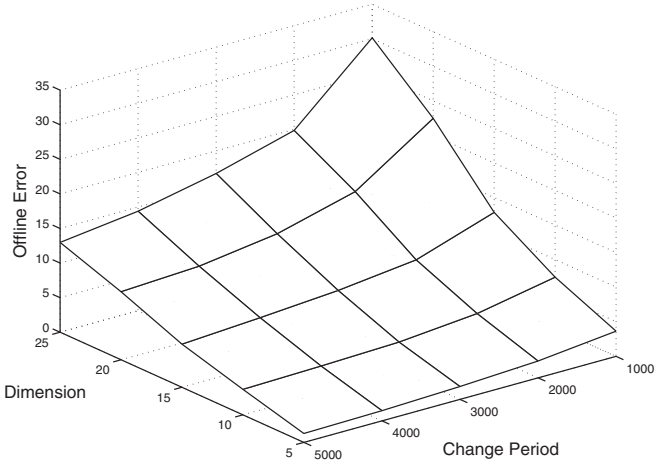

Fig. 2. Offline error of DynDE using the conical peak function with population size of 6 with change severity 1.0 for various settings of dimension and change period.

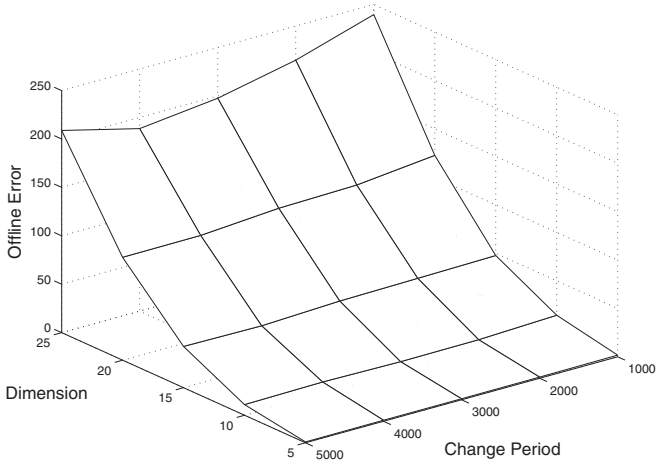

Fig. 3. Offline error of DynDE using the spherical peak function with population size of 6 with change severity 1.0 for various settings of dimension and change period.

outcomes of the Mann–Whitney *U*-tests comparing the RMC results with those of DynDE indicate that, on the whole, only experiments in five dimensions yield a statistically significantly different result from DynDE. Furthermore, only some of the five-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 five-dimensional experiments

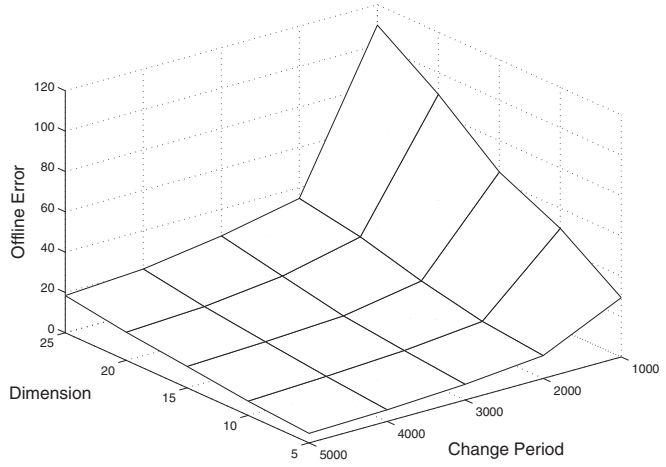

Fig. 4. Offline error of DynDE using the conical peak function with population size of 6 with change severity 5.0 for various settings of dimension and change period.

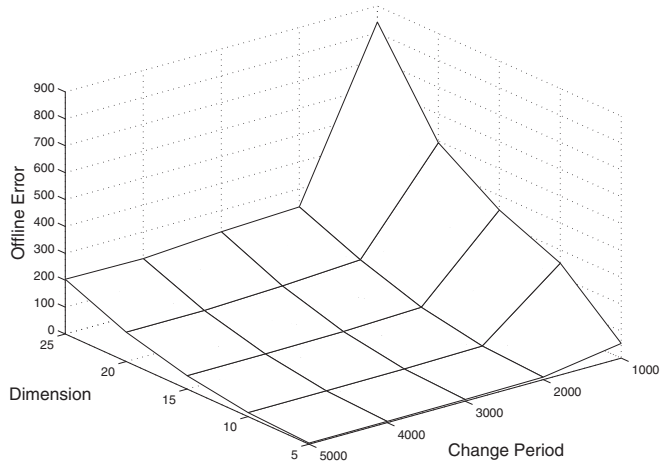

Fig. 5. Offline error of DynDE using the spherical peak function with population size of 6 with change severity 5.0 for various settings of dimension and change period.

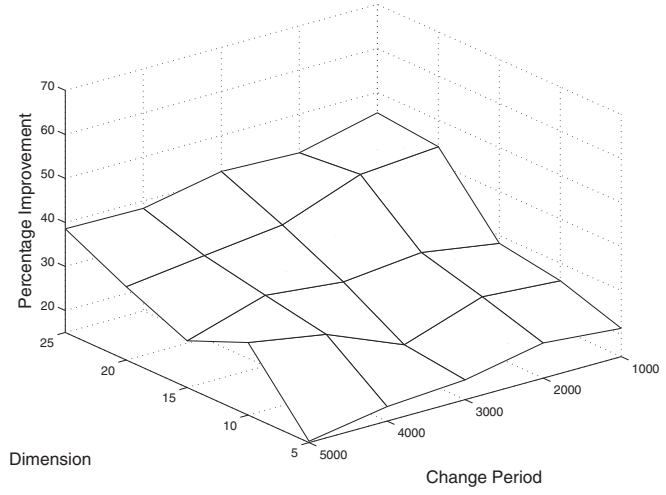

Fig. 6. Percentage improvement of CPE over DynDE using the conical peak function with population size of 6 with change severity 1.0 for various settings of dimension and change period.

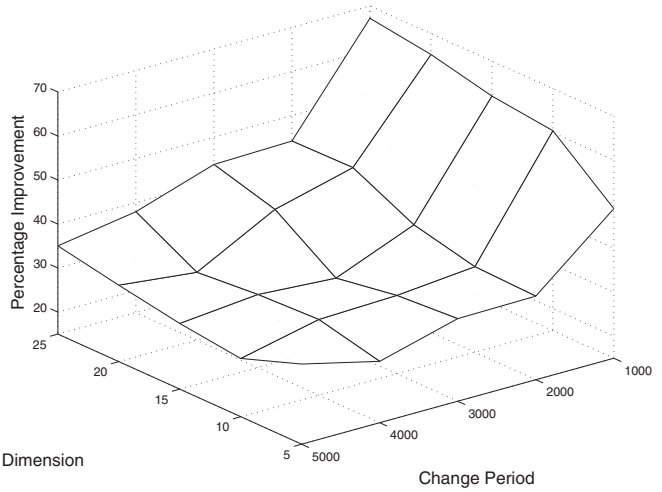

Fig. 7. Percentage improvement of CPE over DynDE using the conical peak function with population size of 6 with change severity 5.0 for various settings of dimension and change period.

with change severity of 1 and population size of six. For all high dimensional experiments almost none of the results are statistically significantly different.

An experiment was conducted to investigate why RMC only delivers improvements in low numbers of dimensions, and why RMC is more effective when using the spherical peak function. 100 000 random moving peak function search spaces were created for each of the numbers of dimensions from 2 to 40. For each case the number of pairs of peaks located within an Euclidean distance below the exclusion threshold of each other were counted. For each pair of these peaks a midpoint check was performed to determine whether the RMC algorithm would have detected two peaks correctly. The results of these experiments are depicted in Figs. 8 and 9 for simulations on the conical and spherical peak functions respectively. Observe that the number of peaks that are located within the exclusion threshold of each other drops sharply from over 300 000 to zero between two and 16 dimensions. This explains why the RMC approach is only affective in low dimensional cases: in high dimensional problems the situations where RMC would have been useful do not occur. The midpoint check approach was more effective at identifying pairs of peaks on the spherical function than on the conical function in low dimensions (i.e. more cases as depicted in Fig. 1, scenarios B and C, occur with the conical peak function). This explains why RMC yielded larger improvements in offline error on the spherical peak function than on the conical peak function.

As with standard DynDE, the population size 6 experiments outperformed the population size 10 experiments.

9.5. CDE results

The results for the combined CDE adaptation are summarized in Tables 2 and 3. 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, CDE performed better than DynDE. Once again, the population size 6 experiments outperformed the population size 10 experiments. 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, 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.

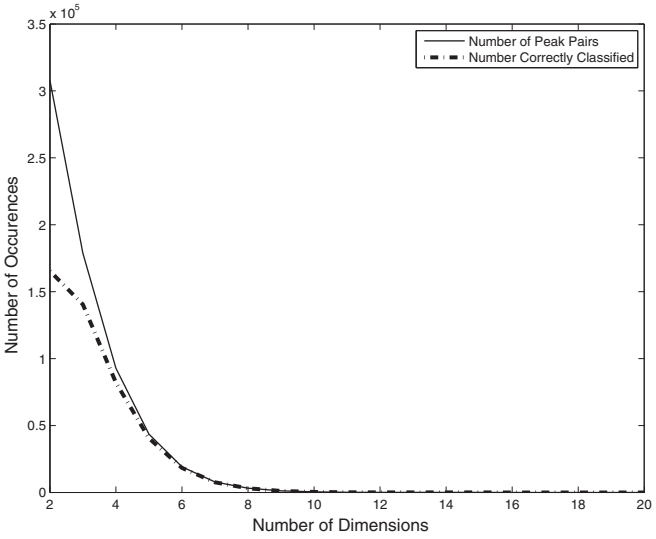

Fig. 8. Number of peak pairs that fall within the exclusion threshold and number of correct classifications by RMC per dimension on the conical peak function.

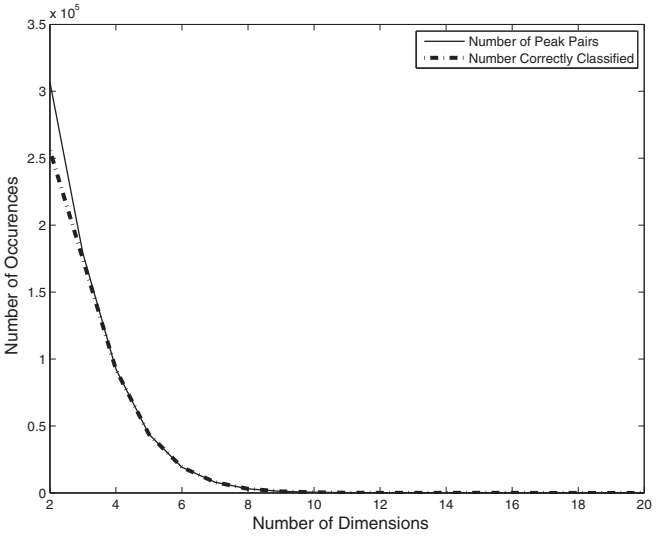

Fig. 9. Number of peak pairs that fall within the exclusion threshold and number of correct classifications by RMC per dimension on the spherical peak function.

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

In the comparison between CPE and CDE it was found that statistically significant improvements of CDE over CPE were isolated mainly to five-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 CDE yielded a statistically significant improvement over CPE in 41 of the cases, while CPE was only statistically significantly better than CDE in five of the cases.

9.6. Effect of CDE on minimum error

The components used to form CDE has been shown to produce a lower offline error than normal DynDE. However, these components could potentially negatively affect the lowest error found before changes occur in the environment. For example, the CPE component could allocate too many function evaluations to a sub-population that is positioned on a sub-optimal peak, to the detriment of the sub-population that is positioned on the global optimum. A reduction in offline error does not necessarily imply that the lowest error found, before a change in the environment occurs, is also reduced.

To investigate the affect of CDE on the lowest errors before changes in the environment, a performance measure, similar to the average mean measure of the Generalized Benchmark Generator (GBG) (refer to Section 10), was incorporated into the MPB. The average current error before changes (ACEBC) is calculated by averaging the MPB current errors (see Section 8) found immediately before changes occur in the environment. The ACEBC of DynDE and CDE can be compared to determine whether CDE has a negative or a positive effect on the lowest error that is found before changes occur in the environment.

Experiments were conducted to investigate the effect of extreme values of change severity, change period and number of dimensions on the MPB. The offline error and the ACEBC error measure for DynDE and CDE were recorded for each experiment. Results are listed in Tables 4–6 for change severity, change period and number of dimensions respectively. Averages over 50 independent runs of the experiments are reported. Outcomes of Mann–Whitney *U*-tests on the differences in performance between DynDE and CDE (**p-val**) are printed in bold when not statistically significant at a 95% confidence level.

For change severities less than 20 it was found that DynDE yielded a lower ACEBC value than CDE. However, differences are small and only statistically significant on one case (change severity of 3). On change severities of 20 and larger, CDE yielded significantly lower ACEBC values than DynDE. From these results it can be concluded that the lowest errors before changes in the environment is not meaningfully affected when using CDE on problems with low and moderate change severities. When the change severity is large, CDE performed better than DynDE in terms of the lowest errors found before changes occur in the environment.

For experiments with a low change period, it was found that CDE performs statistically significantly better than DynDE in terms of ACEBC value. Change periods between 3 000 and 10 000 did not yield statistically significantly different results, while DynDE performed statistically significantly better than CDE when high change periods were used. The slightly inferior performance of CDE in comparison to DynDE in situations where the change period is greater than 20 000 may be attributed to CDE unnecessarily allocating too many function evaluations to sub-populations not located on the global optimum. Note that, in terms of offline error, there were no statistically significant differences between DynDE and CDE results for high change periods. The reason for this result is that CDE adapts DynDE to locate optima faster after a change in the environment occurs, which becomes less beneficial as the period between changes is increased.

CDE performed statistically significantly better than DynDE in terms of ACEBC value with the exception of the five dimensional experiment in which the differences were not statistically significant. The results listed in Table 6 indicate that the magnitude of the improvement of CDE over DynDE is large, especially in the very high dimensional experiments.

The investigation into the lowest errors found before changes in the environment shows that the difference between DynDE and CDE is minimal for low and moderate values of change severity, change period and number of dimensions. However, on extreme settings of these parameters meaningful differences are observed. CDE improves the lowest error before changes in the environment for high change severity and high dimensional problems.

Table 4
Change severity errors.

CS	Offline error			Average current error before changes		
	DynDE	CDE	*p*-Value	DynDE	CDE	*p*-Value
1	1.28 ± 0.09	0.92 ± 0.07	0.00	0.39 ± 0.08	0.35 ± 0.06	**0.73**
3	2.68 ± 0.12	1.98 ± 0.10	0.00	0.43 ± 0.08	0.57 ± 0.09	0.00
5	4.76 ± 0.22	3.26 ± 0.18	0.00	0.64 ± 0.09	0.69 ± 0.09	***0.39***
10	10.46 ± 0.43	6.97 ± 0.32	0.00	1.39 ± 0.20	1.53 ± 0.16	***0.09***
20	24.21 ± 0.86	16.33 ± 0.75	0.00	6.82 ± 0.42	3.18 ± 0.27	0.00
40	29.95 ± 0.89	27.01 ± 0.95	0.00	11.76 ± 0.46	8.84 ± 0.49	0.00
80	37.73 ± 1.10	32.79 ± 1.01	0.00	16.34 ± 0.54	10.10 ± 0.57	0.00

Table 5
Change period errors.

CP	Offline error			Average current error before changes		
	DynDE	CDE	*p*-Value	DynDE	CDE	*p*-Value
1000	3.63 ± 0.10	2.75 ± 0.07	0.00	1.39 ± 0.06	1.03 ± 0.07	0.00
2000	2.43 ± 0.10	1.75 ± 0.08	0.00	0.77 ± 0.08	0.56 ± 0.06	0.00
3000	1.85 ± 0.09	1.31 ± 0.08	0.00	0.60 ± 0.08	0.55 ± 0.09	**0.09**
4000	1.47 ± 0.09	1.16 ± 0.07	0.00	0.47 ± 0.07	0.43 ± 0.08	**0.36**
5000	1.28 ± 0.09	0.92 ± 0.07	0.00	0.34 ± 0.05	0.36 ± 0.06	***0.59***
10000	0.77 ± 0.11	0.70 ± 0.11	**0.09**	0.32 ± 0.11	0.37 ± 0.10	***0.07***
20000	0.48 ± 0.16	0.41 ± 0.11	**0.25**	0.20 ± 0.16	0.20 ± 0.10	0.01
40000	0.30 ± 0.11	0.33 ± 0.10	***0.59***	0.12 ± 0.11	0.17 ± 0.09	0.00
80000	0.23 ± 0.08	0.45 ± 0.26	***0.52***	0.06 ± 0.07	0.23 ± 0.21	0.00
100000	0.25 ± 0.16	0.44 ± 0.28	***0.84***	0.07 ± 0.13	0.24 ± 0.21	0.00

Table 6
Dimension errors.

Dim	Offline error			Average current error before changes		
	DynDE	CDE	*p*-Value	DynDE	CDE	*p*-Value
5	1.28 ± 0.09	0.92 ± 0.07	0.00	0.34 ± 0.06	0.40 ± 0.05	***0.10***
15	6.24 ± 0.32	4.55 ± 0.29	0.00	3.71 ± 0.26	2.92 ± 0.22	0.00
25	12.99 ± 0.66	8.47 ± 0.39	0.00	9.85 ± 0.66	5.99 ± 0.41	0.00
50	37.20 ± 2.94	17.23 ± 0.83	0.00	30.84 ± 2.62	14.10 ± 0.58	0.00
100	93.86 ± 7.39	38.40 ± 2.46	0.00	87.05 ± 7.92	32.87 ± 1.83	0.00
200	261.32 ± 26.93	85.18 ± 4.93	0.00	232.93 ± 16.38	76.68 ± 4.70	0.00

On the other hand, for high change period problems it was found that DynDE performed better than CDE in terms of ACEBC value. This result is not entirely unexpected since the CPE component of CDE aims to locate optima earlier in the optimization process. When changes in the environment are infrequent, the benefit of discovering optima slightly earlier diminishes. It should also be noted that increases in the change period make the environment less dynamic. CDE thus performed better than DynDE in terms of ACEBC value on the harder, low change period problems.

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 CDE algorithm is 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 Optimization (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 CDE 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, CDE resulted in an offline error of 2.70 ± 0.23 and 4.55 ± 0.29, respectively. CDE is thus on a par with MMEO in 10 dimensions and outperforms MMEO in 15 dimensions.

After MMEO, the best reported results are from the CESO and ESCA algorithms (Lung and Dumitrescu, 2007; Lung and Dumitrescu, 2010). Results of 1.38 ± 0.02 for CESO and 1.53 ± 0.01 for ESCA are reported when using the MPB settings in Table 1. Both these results are inferior to that found using CDE. However, in 10 dimensions CESO performs better than CDE with an offline error of 2.51 ± 0.04. In terms of change severity, these algorithms appear to be more scalable than CDE, since, when a change severity of 3 is used, an offline error of 1.67 ± 0.01 is reported for ESCA as opposed to the CDE result of 1.98 ± 0.10. CDE did perform slightly better on this setting than CESO which resulted in an offline error of 2.03 ± 0.03. A change severity of 5 yielded a CESO offline error of 2.52 ± 0.06 and an ESCA offline error of 1.78 ± 0.06. Both these results are superior to the CDE result of 3.26 ± 0.18.

One of the earlier most successful algorithms is that 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. CDE 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 CDE 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 CDE result of 3.26 ± 0.18. In 10 dimensional experiments Blackwell and Branke found an offline error of 4.17 ± 0.15 while using CDE 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.

CDE 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). CDE 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 CDE with respect to change period, was investigated. Table 7 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 CDE in the small change period experiments.

Full results of *jDE* and CDE on the GBG with a change period of 5000 can be seen in Tables 8 and 9 respectively. *jDE* performed better than CDE on only one of the test cases, function 3 with change type 5.

jDE was run on the MPB for the various combinations of settings listed in Table 1 with the cone peak function. The results of the 5 dimensional experiments are given in Table 10.

Comparing these results to those found using CDE in Table 2 shows that CDE performed better than *jDE* on the MPB.

It can thus be seen that the CDE adaptation presented here compares favorably with some of the state of the art approaches in the literature.

Table 7
jDE compared to DynDE and CDE for different values of change period (**CP**) on GBG.

Change period	Measure	*jDE*	DynDE	CDE
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

11. General applicability

In this paper, DynDE (Mendes and Mohais, 2005) was used as base algorithm for the CPE and RMC adaptations. However, while DynDE is well suited for adaptation, the novel adaptations presented here can, in theory, be incorporated into any multi-population optimization algorithm. In order to determine whether the underlying algorithms of CDE yield improved results in another algorithm, *jDE* of Brest et al. (2009) was changed to evolve only one sub-population at a time based on performance (as in CDE), and to reinitialize sub-populations that are located on the same optimum using CDE's exclusion midpoint check algorithm.

The results of the changed *jDE* algorithm on the MPB for various settings of change period and change severity (**Cha. Sev.**) are given in Table 11. The outcomes of Mann–Whitney *U*-tests (**p-val**) comparing differences between *jDE* results and results of *jDE* with CDE components, are listed for each experiment. In this table, *p*-values that were not statistically significant within a 95% confidence interval are printed in boldface and situations where the results were inferior to those of normal *jDE* are printed in italics.

jDE with CDE components performed statistically significantly better than normal *jDE* in 11 of the 15 cases investigated. In two cases, the results were not statistically significantly different, and normal *jDE* performed better than the *jDE* version that utilized CDE components in only two cases. Improvements over *jDE* were especially pronounced in experiments with a low change period and a high change severity, which confirms the usefulness of the CPE adaptation which locates optima earlier and is consequently beneficial in low change period environments.

The experimental results indicate that the ideas behind CDE can be beneficial to other optimization algorithms aimed at dynamic environments. The results are especially encouraging considering that no other changes were made to the rest of *jDE* algorithm. For example, although *jDE* has several parameters that may influence the interaction between normal *jDE* components and the CDE components, improved offline errors were found without even fine-tuning any of the *jDE* parameters.

12. Discussion and conclusions

Competitive Population Evaluation (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.

Reinitialization Midpoint Check (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 deteriorates the performance of standard DynDE algorithm.

On average, the combination of RMC and CPE is in several cases more successful than each of the adaptations on its own. Experiments comparing the performance of Competing Differential Evolution (CDE) to normal DynDE indicated that large improvements were in problems with high change frequency and severity, making CDE a more viable option. Experimental evidence shows that, on extreme values of dimension and change severity, CDE in not only better than DynDE in terms of offline error, but also in terms of the lowest errors found before changes occur in the environment.

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

Table 8
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_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_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_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_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_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_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_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 9
CDE 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_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_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_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_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_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_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_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 10
jDE offline error using conical peak function for different values of change severity and change period.

Cha. Sev.	Change period	Offline error	Cha. Sev.	Change period	Offline error	Cha. 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

Table 11
jDE offline error when using the CDE components.

Cha. Sev.	Change period	Offline error	*p*-Value	Cha. Sev.	Change period	Offline error	*p*-Value	Cha. Sev.	Change period	Offline error	*p*-Value
1	1000	9.63 ± 0.57	0.00	3	1000	16.87 ± 0.82	0.00	5	1000	22.13 ± 0.58	0.00
1	2000	8.30 ± 0.49	0.00	3	2000	14.08 ± 0.55	0.00	5	2000	17.59 ± 0.56	0.00
1	3000	7.33 ± 0.40	***0.09***	3	3000	12.06 ± 0.48	0.00	5	3000	15.17 ± 0.61	0.00
1	4000	7.22 ± 0.55	0.00	3	4000	10.19 ± 0.50	0.00	5	4000	12.83 ± 0.60	0.00
1	5000	6.34 ± 0.49	0.00	3	5000	9.54 ± 0.60	**0.29**	5	5000	11.72 ± 0.65	0.01

Incorporating the sub-components of CDE into *jDE* resulted in improved results on several test cases. This suggests that the algorithms presented in this paper could be beneficial when applied to other optimization algorithms. CDE does not depend on any intrinsic DE behavior. Future work could include studying the effect of applying CDE to other multi-population optimization algorithms for dynamic environments.

The success of *jDE* makes the incorporation self-adaptive control parameters into CDE a logical avenue for future research. Apart from self-adapting the scaling and crossover factors, it may also be

useful to self-adapt the standard deviation of the Gaussian random number used to create Brownian individuals in Eq. (4), and the number of sub-populations.

References

Angira, R., Santosh, A., 2007. Optimization of dynamic systems: A trigonometric differential evolution approach. Computers and Chemical Engineering 31 (9), 1055–1063.

Angira, R., Santosh, A., 2008. A modified trigonometric differential evolution algorithm for optimization of dynamic systems. In: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 1463–1468.

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

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.

Boettcher, S., Percus, A.G., 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., 2007. The Moving Peaks Benchmark. <http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/>.

Branke, J., Schmeck, H., 2003. 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., 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.

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

Das, S., Konar, A., Chakraborty, U., 2005. An improved differential evolution scheme for noisy optimization problems. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1691–1698.

Fan, H.-Y., Lampinen, J., 2003. A trigonometric mutation operation to differential evolution. Journal of Global Optimization 27, 105–129.

Gibbon, T.B., Wu, L., Waswa, D.W., Conibear, A.B., Leitch, A.W.R., 2008. Polarization mode dispersion compensation for the south african optical-fibre telecommunication network. South African Journal of Science, 104–119.

Grefenstette, J.J., 1999. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. 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 hierarchical particle swarm optimizer. In: Congress on Evolutionary Computation. IEEE, pp. 770–776.

Janson, S., Middendorf, M., 2004. A hierarchical particle swarm optimizer for dynamic optimization problems. In: Raidl, G.R. (Ed.), Applications of Evolutionary Computing, LNCS, vol. 3005. 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.

Kaelo, P., Ali, M., 2006. A numerical study of some modified differential evolution algorithms. European Journal of Operational Research 169 (3), 1176–1184.

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

Li, X., Dam, K.H., 2003. Comparing particle swarms for tracking extrema in dynamic environments. Congress on Evolutionary Computation, vol. 3. IEEE, pp. 1772–1779.

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

Liu, B., Zhang, X., Ma, H., 2008. Hybrid differential evolution for noisy optimization. In: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 587–592.

Lung, R.I., Dumitrescu, D., 2007. A new collaborative evolutionary-swarm optimization technique. In: GECCO'07: Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation. ACM, New York, NY, USA, pp. 2817–2820.

Lung, R.I., Dumitrescu, D., 2010. Evolutionary swarm cooperative optimization in dynamic environments. Natural Computing: An International Journal 9 (1), 83–94.

Mendes, R., Mohais, A., 2005. DynDE: A differential evolution for dynamic optimization problems. In: Congress on Evolutionary Computation. IEEE, pp. 2808–2815.

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, LNCS, vol. 1141. Springer-Verlag, Berlin, pp. 513–522.

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., 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, LNCS, vol. 1498. Springer, pp. 149–158.

Morrison, R., 2004. Designing Evolutionary Algorithms for Dynamic Environments. Springer.

Moser, I., 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.

Omran, M.G., Engelbrecht, A.P., Salman, A., 2009. Bare bones differential evolution. European Journal of Operational Research 196 (1), 128–139.

Oppacher, F., Wineberg, M., 1999. The shifting balance genetic algorithm: Improving the GA in a dynamic environment. In: Banzhaf, W. et al. (Eds.), Genetic and Evolutionary Computation Conference (GECCO), vol. 2. 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.

Parrott, D., Li, X., 2006. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Transactions on Evolutionary Computation, 440–458.

Price, K., Storn, R.M., Lampinen, J.A., 2005. Differential Evolution – A Practical Approach to Global Optimization. Springer.

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.

Salman, A., Engelbrecht, A.P., Omran, M.G., 2007. Empirical analysis of self-adaptive differential evolution. European Journal of Operational Research 183 (2), 785–804.

Sareni, B., Krahenbuhl, L., 1998. Fitness sharing and niching methods revisited. IEEE Transactions on Evolutionary Computation 2 (3), 97–106.

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.

Thomsen, R., 2004. Multimodal optimization using crowding-based differential evolution. In: CEC'04: Proceedings of the 2004 Congress on Evolutionary Computation. IEEE Press, pp. 382–1389.

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.