

# CDEPSO: A Bi-population Hybrid Approach for Dynamic Optimization Problems

Javidan Kazemi Kordestani<sup>1</sup>, Alireza Rezvanian<sup>2,\*</sup>, Mohammad Reza Meybodi<sup>2</sup>

<sup>1</sup>*Department of Electrical, Computer and IT Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

<sup>2</sup>*Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran*

Javidan.kazemi@gmail.com, a.rezvanian@aut.ac.ir, mmeybodi@aut.ac.ir

## Abstract

Many real-world optimization problems are dynamic, in which the environment, i.e. the objective function and restrictions can change over time. In this case, the optimal solution(s) to the problem may change as well. These problems require optimization algorithms to continuously and accurately track the trajectory of the optima (optimum) through the search space. In this paper, we propose a bi-population hybrid collaborative model of Crowding-based Differential Evolution (CDE) and Particle Swarm Optimization (PSO) for Dynamic Optimization Problems (DOPs). In our approach, called CDEPSO, a population of genomes is responsible for locating several promising areas of the search space and keeping diversity throughout the run using CDE. Another population is used to exploit the area around the best found position using the PSO. Several mechanisms are used to increase the efficiency of CDEPSO when finding and tracking peaks in the solution space. A set of experiments was carried out to evaluate the performance of the proposed algorithm on dynamic test instances, generated using the Moving Peaks Benchmark (MPB). Experimental results show that the proposed approach is effective in dealing with DOPs.

**Keywords:** Dynamic Optimization Problems, Crowding-based Differential Evolution, Particle Swarm Optimization, Moving Peaks Benchmark, MPB.

## 1. Introduction

Optimization in dynamic environments is of great importance because in many real-world optimization problems the objective function, the problem instance, or other elements may change during the course of optimization. Unlike optimization in static environments [1-3], where the task of optimization is limited to finding the optimum solution(s) in the search space, the goal of optimization in dynamic optimization problems (DOPs) is to track the trajectory of the moving optima. DOPs have attracted a great deal of attention during the last two decades, because they can closely approximate a variety of real-world situations. The adaptive nature of Evolutionary Algorithm (EA) and Swarm Intelligence (SI) techniques make them good candidates

for addressing DOPs. To ensure that EA and SI methods obtain promising results for DOPs, we should address two major issues. The first is *outdated memory*, which appears due to changes in the environment. Once a change occurs in the environment, the fitness of the previously found positions may no longer be valid and can provide the population with misleading information. This problem can be solved by re-evaluating all individuals. The second issue is *diversity loss*, which arises when the population converges, and may prevent individuals from successfully tracking the moving optima. Between these two problems, diversity loss is much more serious and has a very detrimental impact on performance [4]. Several approaches have been proposed to deal with the diversity loss problem in dynamic environments that can be categorized into four groups [5] as follows:

- I. ***Generating diversity after a change:*** Various studies exist that try to deal with the diversity loss problem by injecting diversity in a partially converged population, after detecting a change in the environment. An example of this approach is the Re-randomization PSO (RPSO) proposed by *Hu and Eberhart* [6]. In RPSO, when a change in the environment is detected, the entire (or part of the) swarm is randomly relocated in the search space. Their results suggest that, when the step size of the environmental change is small, the re-randomization of a small portion of the swarm can improve the tracking ability of the PSO. Note that it is challenging to specifying the proper rate of randomization. Too much randomization will result in a loss of information and resemble solving the problem from scratch. On the other hand, too little randomization may be unable to introduce enough diversity to the population, which is necessary for tracking the optimum.
- II. ***Maintaining diversity throughout the run:*** A group of studies has attempted to preserve a sustainable level of diversity during the entire run by avoiding the convergence of individuals to an optimum. For example, *Yang* [7] proposed the Random Immigrants Genetic Algorithm (RIGA), where at each generation a part of the population is replaced by randomly generated individuals. This can reduce the tendency of a population to converge to a narrow region of the search space, thereby enhancing the diversity. It should be noted that a continuous focus on maintaining diversity can slow down the optimization process, because peaks in the promising areas are not exploited.
- III. ***Memory-based approaches:*** Many researchers have equipped EA and SI methods with a memory scheme that restores useful information about previously found positions in the search space. This technique is especially useful in periodical or recurrent environments, where the optima may repeatedly reappear in regions near to their previous positions [8, 9].
- IV. ***Multi-population approaches:*** Using a multi-population scheme, instead of single population, is one of the most successful techniques for enhancing diversity. In this approach, the main population is divided into several sub-populations and each of which is responsible for handling a separate area of the search space [10–13].

Recently, several novel multi-population schemes have emerged as effective methods for optimization in dynamic environments. Examples of such schemes involve the following:

- ***Space partitioning:*** One of the main reasons why traditional population-based methods fail when solving DOPs is that too many individuals search the same limited area of the search space. One possible solution is to divide the search space into several partitions

and keep the number of individuals in each partition less than a predefined threshold [14–19].

- **Population clustering:** Clustering methods can be used to divide the main population into several clusters (sub-populations) that are assigned to different promising regions of the search space. This is a relatively novel multi-population approach for DOPs [20, 21].
- **Hybrid methods:** Another approach is to combine desirable features from the different methods, taking advantage of their strengths and mitigating their weaknesses for DOPs [22, 23].

The hybrid methods are a growing area of intelligent systems research, which try to establish a good balance between the ability of algorithms to visit many different regions of the search space (*exploration*), and to obtain high-quality solutions within those regions (*exploitation*) [24]. Hybrid methods have been successfully applied to many problems [24].

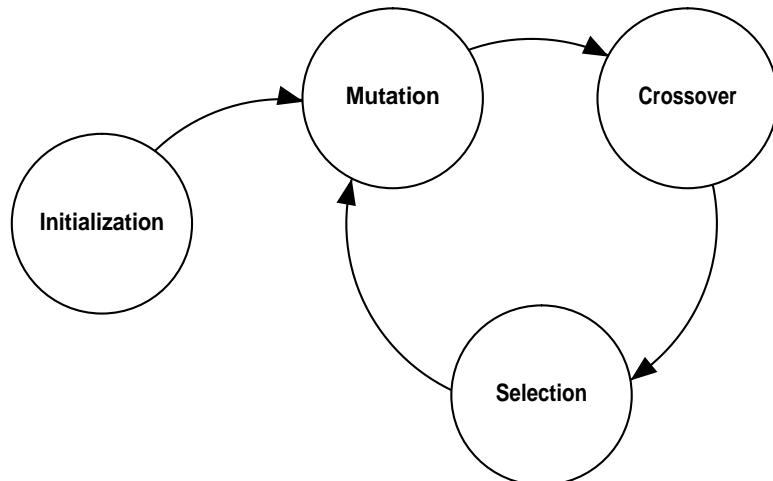
This paper presents a bi-population hybrid algorithm of CDE and PSO. Our method uses two separate populations. The first population is responsible for locating several promising areas of the search space and preserves the diversity throughout the run using CDE. The second population is used to exploit the best found area using PSO. We also introduce several mechanisms to further improve the performance of the proposed algorithm. Our approach is similar to that of CESO [23] in the sense that it hybridizes CDE and PSO, and it has two populations. However, the mechanisms of our proposed method are completely different.

The rest of this paper is organized as follows: Section 2 provides a brief review of DE and PSO, and their modifications for dynamic environments. Section 3 is devoted to the proposed algorithm. Experimental results and analysis are presented in Section 4. Finally, conclusions and future works are given in Section 5.

## 2. Related Work

### 2.1 Basics of DE

DE is a simple yet powerful stochastic real-parameter optimization algorithm, proposed by Storn and Price [25]. The main idea of DE is to use spatial difference among the population of vectors to guide the search process toward the optimum solution. The main steps of DE are illustrated in Figure 1.



**Figure 1. Schematic view of the main stages of DE algorithm**

Different variations of DE are specified with a general convection **DE/x/y/z**, where **DE** stands for “Differential Evolution”, **x** represents a string denoting the base vector to be perturbed, **y** is the number of difference vectors considered for perturbation of **x**, and **z** stands for the type of crossover being used (exponential or binomial) [25]. Algorithm 1 shows a typical pseudo-code for **DE/rand/1/exp** [25].

**Algorithm 1.** Pseudo-code for DE/rand/1/exp

1. Setting parameters
  2. Randomly generate the initial population of vectors in the D-dimensional search space
  3. **repeat**
  4.   **for each** genome  $i$  in the population **do**
  5.     select three mutually exclusive random genomes  $\vec{x}_1 \neq \vec{x}_2 \neq \vec{x}_3$
  6.     generate a donor vector according to Eq. (1)
  7.        $\vec{v}_i = \vec{x}_1 + \mathcal{F} \cdot (\vec{x}_2 - \vec{x}_3)$
  8.        $L=0$
  9.       **do**
  10.        |  $L=L+1$
  11.        | **while**(rand(0,1)<=Cr **AND**  $L \leq D$ )
  12.        |  $N =$  a random integer in the range of [1,D]
  13.        | generate a trial vector  $\vec{u}_i$  using exponential crossover by Eq. (2)
  14.        |  $u_{ij} = \begin{cases} v_{ij} & \text{for each dimension } j = (N \bmod D), (N + 1 \bmod D), \dots, (N + L - 1) \bmod D \\ x_{ij} & \text{for all other dimensions } j \in [1, D] \end{cases}$
  15.        | evaluate the candidate  $\vec{u}_i$
  16.        | replace  $\vec{x}_i$  with  $\vec{u}_i$ , if fitness of  $\vec{u}_i$  is better than fitness of  $\vec{x}_i$
  17.     **end-for**
  18. **until** a termination condition is met

**Figure 2. Pseudo-code for DE/rand/1/exp**

In algorithm 1,  $\mathcal{F}\epsilon[0,1]$  is a scale factor used to control the amplification of difference vector  $(\vec{x}_2 - \vec{x}_3)$ , and  $Cre(0,1)$  is the crossover rate for specifying the approximate number of components transferred from donor vector  $\vec{v}_i$  to trial vector  $\vec{u}_i$ .

The only difference between DE and CDE is in the replacement stage, where in the conventional DE, an offspring replaces its parents (if it is fitter) whereas in CDE, the produced offspring replaces the most similar genome of the population (if it is fitter). The similarity measure in real-valued encoding is Euclidean distance between two candidate solutions [26]. This simple scheme can provide the population of solutions with a good level of diversity which results in a suitable coverage of the search space with a better chance to locate multiple optima. CDE has several advantages that make it a powerful tool for multi-modal optimization problems. Specifically, (a) In comparison with other DE variants CDE has a fast convergence; (b) CDE has the ability to fine-tune the found solutions; (c) CDE is robust in the sense that it can regenerate the results over several runs, i.e. standard deviation of the results is equal or close to zero [26].

## 2.2 Basics of PSO

PSO is a population-based optimization method which was first proposed by *Kennedy and Eberhart* [27]. PSO begins with a population of randomly generated particles in a  $D$ -dimensional search space. Each particle  $i$  of the swarm has three features:  $\vec{x}_i$  that shows the current position of the particle  $i$  in the search space,  $\vec{v}_i$  which is the velocity of the particle  $i$  and  $\vec{p}_i$  which represents the best position found so far by the particle  $i$ . Each particle  $i$  updates its position in the search space, at every time step  $t$ , according to the following equations:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1 r_1 [\vec{p}_i(t) - \vec{x}_i(t)] + c_2 r_2 [\vec{p}_g(t) - \vec{x}_i(t)] \quad (3)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (4)$$

where  $w$  is the inertia weight parameter which determines the portion of velocity preserved during the last time step of the algorithm.  $c_1$  and  $c_2$  denote the cognitive and social learning factors that are used to adjust the degree of the movement of particles toward their personal best position and global best position of the swarm, respectively.  $r_1$  and  $r_2$  are two independent random variables drawn with the uniform probability from  $[0,1]$ . Finally,  $\vec{p}_g$  is the globally best position found so far by the swarm. The pseudo-code of the PSO algorithm is shown in Algorithm 2.

---

**Algorithm 2.** Pseudo-code for canonical PSO

---

1. Setting parameters
  2. Generate the initial swarm of the particles with random positions and velocities
  3. Evaluate the fitness of each particle of the swarm
  4. **repeat**
  5.   **for each** particle  $i$  in the swarm **do**
  6.     update particle  $i$  according to Eqs.(3),(4)
  7.     **if**(fitness( $\vec{p}_i$ )<fitness( $\vec{x}_i$ )) **then**
  8.        $\vec{p}_i = \vec{x}_i$
  9.       **if**(fitness( $\vec{p}_g$ )<fitness( $\vec{x}_i$ )) **then**
  10.           $\vec{p}_g = \vec{x}_i$
  11.       **end-if**
  12.     **end-if**
  13.   **end-for**
  14. **until** a termination condition is met
- 

Figure 3. Pseudo-code for canonical PSO

PSO has been widely studied and applied to a wide range of applications with promising results due to its ease of implementation and fast convergence [28–34].

## 2.3 PSO and DE in Dynamic Environments

So far, numerous methods for solving DOPs based on PSO and DE have been proposed. *Blackwell and Branke* [10, 35] proposed a multi-swarm algorithm based on an atomic metaphor, which they called mQSO. The main idea of mQSO is to establish a mutual repulsion between a

number of swarms, so that each swarm is positioned on a separate peak in the search space. In mQSO, each swarm is composed of a number of neutral particles and a number of quantum particles. The neutral particles in a swarm update their velocity and position according to the principles of standard PSO. However, the quantum particles do not move according to the PSO dynamics, they change their positions around the center of the best particle of the swarm according to a random uniform distribution with radius  $r_{cloud}$ . Consequently, they never converge, but provide the swarm with the sustainable level of diversity that is needed to capture the shifting optimum. Moreover, they also proposed operators to establish two forms of interactions between swarms. An operator named *exclusion*, operates when two swarms are searching in the same area of the search space. It prevents swarms from settling on the same peak by reinitializing the worst performing swarm in the search space. The second operator is *anti-convergence*, which is triggered upon the convergence of all swarms. It removes the worst swarm from its peak and reinitializes it in the search space.

Inspired by mQSO, Mendes and Mohais [36] proposed a multi-population differential evolution algorithm for dynamic environments, called DynDE. In DynDE, several populations of genomes are initialized in the search space to explore high quality regions. They incorporate the *exclusion* and *diversification* mechanisms. They use three components to introduce diversity to each population, namely *Quantum individuals*, *Brownian individuals* and *Entropic differential evolution*. Moreover, DynDE uses random values for the *F* and *Q* parameters, taken from the uniform distribution. This eliminates the need for the fine-tuning of parameters, which can be very time consuming.

Kamosi et al. [12] introduced a multi-swarm algorithm for dynamic environments, which consists of a parent swarm to explore the search space and a varying number of child swarms to exploit promising areas found by the parent swarm. In this method, called mPSO, the optimization process starts with a parent swarm exploring the entire search space for locating promising areas. Whenever the fitness value of the best particle in the parent swarm improves, a child swarm will be created as follows: (a) a number of particles located within the radius  $r$  from the best particle of the parent swarm are separated from the parent swarm and form a new child swarm (b) a number of particles are randomly initialized in a hyper-sphere with radius  $r/3$  centered at the best particle of the child swarm. Afterwards, a number of randomly initialized particles are added to the parent swarm in order to compensate for the migration of particles from parent swarm to the child swarm. After creating child swarms, each of them exploits their respective sub-regions in the search space. Upon detecting a collision between two child swarms, the worse child swarm is removed. In case that a particle of the parent swarm finds a better position in the search territory of a child swarm, it replaces the local best particle of the respective child swarm. Afterwards, the particle of the parent swarm is reinitialized in the search space. In another work, inspired by hibernation phenomenon in animals, they extended mPSO and proposed a hibernating multi-swarm algorithm (HmSO) [11]. In HmSO, unproductive search in child swarms is stopped by “sleeping” converged child swarms. This hibernation mechanism makes more function evaluations available for the parent swarm to explore the search space, and for the other child swarms to exploit their respective areas in the landscape.

Hashemi and Meybodi [14] incorporated PSO and Cellular Automata (CA) into an algorithm referred to as Cellular PSO. In Cellular PSO, the search space is partitioned into some equally

sized cells using CA. Then, particles of the swarm are allocated to different cells according to their positions in the search space. At any time, particles residing in each cell use their personal best experience and the best solution found in their neighborhood cells for searching an optimum. Moreover, whenever the number of particles within each cell exceeds a predefined threshold, randomly selected particles from the saturated cells are transferred to random cells within the search space. In addition, each cell has a memory that is used to keep track of the best position found within the boundary of the cell and the best position among its neighbors. In another work [15], they changed the role of some particles in each cell, from standard particles to quantum particles for a few iterations upon the detection of a change in the environment.

*Noroозi et al.* [16] adopted the concept of Cellular PSO and proposed an algorithm based on DE, called CellularDE. CellularDE employs DE/rand-to-best/1/bin scheme to provide the genomes of each cell with local exploration capability. Moreover, upon detection of a change in the environment, all genomes perform a random local search for several upcoming iterations. In another work [37], they equipped CellularDE with a mechanism to estimate the presence of a peak within each cell's boundary or its neighbors. In this approach, more function evaluations are allocated to the cells which are more likely to contain local optimum. In addition, the search process in cells that contains peaks is conducted through hill-climbing local search around the best position of the cell. Moreover, they have used a directed local search to improve the tracking ability of the algorithm after detecting a change in the environment.

*Du Plessis and Engelbrecht* [38] proposed Dynamic Population Differential Evolution (DynPopDE), in which the populations are adaptively spawned and removed, as required, based on the search progress. DynPopDE starts with a single population and gradually adapts to an appropriate number of populations. In this approach, whenever all of the current populations fail to improve their qualities after spending their last respective function evaluations, a new population of random individuals are generated in the search space. On the other hand, when the number of populations surpasses the number of peaks in the landscape, redundant populations are detected as those which are frequently reinitialized by exclusion operator.

*Lung and Dumitrescu* introduced a hybrid collaborative approach for dynamic environments called Collaborative Evolutionary Swarm Optimization (CESO) [23]. CESO has two equal-size populations: a main population to maintain a set of local and global optima during the search process using CDE, and a PSO population acting as a local search operator around solutions provided by the first population. During the search process, information is transmitted between both populations via collaboration mechanisms. In another work [22], a third population is incorporated to CESO which acts as a memory to recall some promising information from past generations of the algorithm.

*Nickabadi et al.* [39] proposed a Competitive Clustering Particle Swarm Optimizer (COPSO) for DOPs. They employed a multi-stage clustering procedure to split the particles of the main swarm over a varying number of sub-swarms. In addition to the sub-swarms, there is also a group of free particles that is used to explore the environment to locate new emerging optima or exploit the current optima which are not followed by any sub-swarm. Moreover, each sub-swarm adjusts its inertia weight according to the success percentage of the sub-swarm [40].

### 3. The Proposed Algorithm

#### 3.1. General procedure

The proposed algorithm consists of two separate populations called QUEST and TARGET. The QUEST population, which follows the principles of CDE, explores the search space to find promising areas. Moreover, it preserves diversity among the solutions by avoiding the convergence of the entire population to the same area of the search space. The TARGET population, which is evolved by PSO, also has two critical roles: (1) to search in the area around the best position found so far by QUEST, and refines the best found solution, and (2) to track of the best found position after a change occurs in the environment. We have incorporated different collaborative mechanisms between the two populations to enhance the ability of the algorithm in dynamic environments. The first mechanism is related to the PSO velocity update equation in the TARGET population. Each particle in the TARGET population adjusts its movement toward its personal best position and the best position found by the QUEST population, ( $QUEST_{best}$ ), using the following equations:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1 r_1 [\vec{p}_i(t) - \vec{x}_i(t)] + c_2 r_2 [QUEST_{best}(t) - \vec{x}_i(t)] \quad (5)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (6)$$

When the TARGET population finds a better position than the best position found by QUEST, it replaces the most similar individual in QUEST. Furthermore, ineffective searches in the TARGET population are stopped by making the TARGET inactive, which reduces the computational cost. As studied in [41], it is possible to accelerate DE by hybridizing it with a local search operation to improve the neighborhood exploitation. Therefore, in this work we use a simple hill-climbing search in some iterations to improve the performance of the QUEST population. We use a restart mechanism in the QUEST population to increase the diversity, which prevents genomes from searching in the same area. The outline of CDEPSO is shown in Algorithm 3.

#### 3.2. Detection and response to changes

In CDEPSO, best genome of the QUEST population ( $QUEST_{best}$ ) is re-evaluated, at the beginning of each iteration, to see if a change occurs in the environment. A change is detected when an alteration in the fitness of the  $QUEST_{best}$  is observed. Upon detecting a change in the environment, all genomes of the QUEST are re-evaluated and the TARGET is restarted in a hyper-sphere with radius of  $r_s$  centered at the  $QUEST_{best}$ . Afterwards, the QUEST population is evolved by CDE principles.

#### 3.3. Saving function evaluations

After updating the position of all genomes, we check to see if the TARGET population has stagnated. Stagnation occurs due to one of the following reasons: (1) the TARGET is moving in a plateau, (2) the TARGET is oscillating around a false attractor, or (3) individuals are oscillating along a line perpendicular to the true optimum. In all three cases, it is rare that the TARGET can escape the situation and many function evaluations are wasted without reaching a better

position. Therefore, if the best position found by the TARGET population has not improved over a certain number of iterations, we make the TARGET population inactive. This makes more function evaluations available for the QUEST population to explore the search space. In this case, each iteration evolves the QUEST population in the normal manner and performs a hill-climbing local search on each genome of the QUEST population. If the QUEST population finds a better position, we give the TARGET population another chance to reach to a better position. If the TARGET population is not stagnant, it either comes closer to  $\text{QUEST}_{\text{best}}$  by restarting in a hyper-sphere with radius  $R_{\text{local}}$ , or updates its particles using Eqs. (5) and (6), respectively. The TARGET population is brought closer to  $\text{QUEST}_{\text{best}}$ , because the position of the best genome in the search space may change and causing the TARGET population to wander and waste function evaluations without contributing to the search.

---

**Algorithm 3.** CDEPSO Collaborative Optimization Algorithm

---

```

1. setting parameters
2. randomly initialize populations
3. repeat
4.   if a change is detected in the environment then
5.     for each genome  $i$  in the QUEST population do
6.       | re-evaluate  $\text{QUEST}_i$ ;
7.     end-for
8.     restart TARGET within a hyper-sphere with radius  $r_s$  centered at  $\text{QUEST}_{\text{best}}$  genome
9.     stagnationCounter = 0
10.    end-if
11.    for each genome  $i$  in the QUEST population do
12.      evolve  $\text{QUEST}_i$ 
13.    end-for
14.    if stagnationCounter <  $\theta$  then
15.      if distance( $\text{TARGET}_{\text{best}}$  ,  $\text{QUEST}_{\text{best}}$ ) >  $\varepsilon$  then
16.        | restart TARGET within a hyper-sphere with radius of  $R_{\text{local}}$  centered at  $\text{QUEST}_{\text{best}}$  genome
17.      end-if
18.      for each particle  $i$  in the TARGET do
19.        | update particle's velocity and position according to Eqs. (5) and (6), respectively
20.        | update  $p_i$ ;
21.      end-for
22.    else
23.      for each genome  $i$  in QUEST do
24.        | perform a hill-climbing search around  $\text{QUEST}_i$ ;
25.        | if new position of  $\text{QUEST}_i$  is better than its previous one do
26.          | | transfer  $\text{QUEST}_i$  to the new position
27.        | end-if
28.      end-for
29.      if position  $\text{QUEST}_{\text{best}}$  is better than position  $\text{TARGET}_{\text{best}}$ 
30.        | stagnationCounter = stagnationCounter - 1
31.      end-if
32.    end-if
33.    if position of  $\text{TARGET}_{\text{best}}$  is better than  $\text{QUEST}_{\text{best}}$  then
34.      | replace the nearest genome of QUEST with  $\text{TARGET}_{\text{best}}$ 
35.    end-if
36.    if  $\text{TARGET}_{\text{best}}$  hasn't improved then
37.      | stagnationCounter = stagnationCounter + 1

```

---

---

```

38. else
39.     stagnationCounter = 0
40. end-if
41. find nearest pair of genomes  $(i,j)$  in QUEST
42. if  $\text{distance}(\text{QUEST}_i, \text{QUEST}_j) < \xi$  then
43.     | reinitialize the worst one in the search space, randomly
44. end-if
45. until a termination condition is met

```

---

Figure 4. Pseudo-code for CDEPSO

### 3.4. Complexity Analysis

The most computationally expensive stage of CDEPSO is the search for the nearest pair of genomes in the QUEST population. Therefore, the complexity of the algorithm can be estimated using the number of Euclidean distances between genomes. Assuming that there are  $N$  genomes in the QUEST population, the number of Euclidean distance calculations between all genomes required for finding the nearest pair,  $T(N)$ , can be calculated as follows:

$$T(N) = \sum_{i=1}^N (i - 1) = \frac{N(N-1)}{2} \quad (7)$$

Thus, the worst time complexity of this operation using a naive strategy is  $O(N^2)$ . This is similar to the time complexity of methods that apply the *exclusion* operator between pairs of populations, e.g. mQSO [35] and mPSO [12]. However, CDEPSO performs better than those methods. Moreover, this time complexity can be reduced to  $O(N \log N)$  by using a more sophisticated method such as the recursive divide-and-conquer approach [42]. Besides, there are only a small number of genomes in the QUEST population (i.e. 10 genomes). The complexity of cellular approaches (e.g. CellularPSO [14, 15] and CellularDE [16]) creates a bottleneck, and CDEPSO performs much better in both complexity and performance. The CESO [23] which hybridizes CDE with PSO, has a complexity of  $O(N)$ . However, the control parameters of CDEPSO allow it to be fine-tuned for different DOPs.

## 4. Experimental study

### 4.1. Dynamic test function

MPB is a dynamic environment with a D-dimensional landscape consisting of  $N$  peaks, where the height, the width and the position of each peak are changed slightly every time a change occurs in the environment [9]. The fitness landscape is defined as the maximum over all peak functions as below:

$$f(\vec{x}, t) = \max_{i=1, \dots, N} H_t(i) - W_t(i) \sqrt{\sum_{j=1}^D (x_t(j) - X_t(i, j))^2} \quad (8)$$

where  $H_t(i)$  and  $W_t(i)$  are the height and the width of peak  $i$  at time  $t$ , respectively. The coordinates related to the location of peak  $i$  at time  $t$  are expressed by  $X_t(i, j)$ . The location of each peak is changed by a vector  $\vec{v}_t(i)$  which is formulated as follows:

$$\vec{v}_{t+1}(i) = \frac{s}{|\vec{r} + \vec{v}_t(i)|} ((1 - \lambda)\vec{r} + \lambda\vec{v}_t(i)) \quad (9)$$

where the shift vector  $\vec{v}_t(i)$  is a linear combination of a random vector  $\vec{r}$ , which is created by drawing random numbers in [-0.5, 0.5] for each dimension, and the current shift vector  $\vec{v}_t(i)$ , and normalized to the length  $s$ . The parameter  $\lambda$  determines the trajectory of changes, where  $\lambda=0$  means that the peaks are shifted in completely random directions and  $\lambda=1$  means that the peaks always follow the same direction, until they hit the boundaries where they bounce off. In order to investigate the performance of the proposed algorithm and compare it with alternative methods, various experiments are performed based on the MPB Scenario 2 [43]. Unless stated otherwise, environmental parameters are set according to the values listed in Table 1.

**Table 1. Parameter settings for the Moving Peaks problem (Scenario 2)**

Parameter	Value
Number of peaks (m)	10
Height severity	7.0
Width severity	1.0
Peak shape	Cone
Number of dimensions (D)	5
Height range (H)	$\in [30.0, 70.0]$
Width range (W)	$\in [1.0, 12.0]$
Standard height (l)	50.0
Search space range (A)	$[0.0, 100.0]^D$
Frequency of change ( $f$ )	5000
Shift severity (s)	1.0
Correlation coefficient ( $\lambda$ )	0.0

## 4.2 Performance measure

In order to evaluate the efficiency of different algorithms, offline error is used as the performance measure which is defined as follows [9, 43]:

$$\text{offline error} = \frac{1}{T} \sum_{t=1}^T (f(\text{best\_found})_t - f(\text{global\_optimum})_t) \quad (10)$$

where  $T$  is the maximum number of function evaluations, and  $f(\text{best\_found})_t$  and  $f(\text{global\_optimum})_t$  are the fitness value of the best position found by the algorithm and the global best position in the fitness landscape at  $t^{\text{th}}$  function evaluation.

## 4.3 Experimental settings

In ODEPSO, the population size for the QUEST and the TARGET population is set to 10. Regarding QUEST population, the scale factor  $F$  and crossover probability  $Cr$  are set to 0.5 and 0.9, respectively. For the TARGET population, the acceleration coefficient  $c_1$  and  $c_2$  are set to

1.496180 and 1.0, respectively. The reason for  $c_2=1.0$  is to increase the capability of the TARGET in searching around QUEST<sub>best</sub>. Inertia weight  $w$  is set to 0.729844. The velocity of particles is clamped within the range [-50,50]. The parameters  $r_s$  and  $R_{local}$  are set to 0.5 and 5, respectively. The thresholds  $\varepsilon$  and  $\theta$  are set to 20 and 10, respectively. The value of  $\xi$  is chosen from the interval [0.0,30.0] empirically. Finally, the hill-climbing local search radius is set to 0.4. The parameters setting of the proposed method is shown in Table 2.

Table 2. Default parameters settings for CDEPSO

Parameter	Value
DE scheme	DE/rand/1/exp [26]
$F$	0.5 [26]
$Cr$	0.9 [26]
$c_1$	1.496180 [12]
$c_2$	1.0
$w$	0.729844 [12]
$v_{clamp}$	[-50,50]
$r_s$	0.5
$R_{local}$	5.0
$\varepsilon$	20.0
$\theta$	10
$\xi$	[0.0,30.0]
Hill-climbing local search radius	0.4
QUEST and TARGET population size	10 [23]

For each experiment of the proposed algorithm on a specific DOP, 100 independent runs are executed with different random seeds and the results are reported in terms of average offline error and standard error. For each run of the algorithm, 100 environmental changes are considered as the termination condition, which results in  $f \times 100$  function evaluations.

#### 4.4 Experimental study on comparing CDEPSO with other state-of-the-art algorithms

In this section, we compare the performance of CDEPSO with various state-of-the-art algorithms using different DOPs generated by the MPB. We investigated the effect of important environmental parameters (i.e. the number of peaks, shift severities and intervals) on the performance of the proposed method. The results of the alternative algorithms were taken from the corresponding papers, except for the AmQSO where the results were taken from [44]. To verify statistical differences among the algorithms, we performed a two-tailed  $t$ -test to compare each pair of results.

##### 4.4.1 Experiment on DOPs with different number of peaks

In this experiment, the performance of CDEPSO and ten other state-of-the-art algorithms was investigated using MPB with a different number of peaks,  $m \in \{1, 5, 10, 20, 30, 50, 100, 200\}$ . Table 3 presents the numerical results of the eleven algorithms in terms of the offline error and standard error. From Table 3, we can observe that the performance of CDEPSO was much better than those of mCPSO [35], mQSO [35], SPSO [45], AmQSO [13], CellularPSO [14, 15], mQSO Rand-Rule [46], mPSO [12], CellularDE [16], DHPSO [47] and DynPopDE [38]. Among the ten peer algorithms, only CESO [23] can outperform CDEPSO on three DOPs.

The results also indicate that the number of peaks affects the performance of CDEPSO. When the number of peaks in the environment is not very large (i.e. 1, 5, 10 and 20), the QUEST population can efficiently locate the promising areas of the search space, and the TARGET population can further exploit the best found peak. For DOPs with a large number of peaks, it is not possible for the QUEST population to locate all peaks in the fitness landscape at the same time, because of its small population size. However, as shown in Table 3, CDEPSO can still achieve a better performance on most of DOPs when compared to the other nine peer algorithms.

**Table 3. Offline error ± standard error of algorithms on MPB with  $f=5000$ ,  $D=5$ ,  $s=1$ ,  $\lambda=0.0$ , and different number of peaks. The t-test results of comparing CDEPSO with contestant algorithms are shown in brackets. The best result for each DOP among the compared algorithms is highlighted in boldface.**

<i>m</i>	mCPSO (2006), [35]	mQSO (2006), [35]	SPSO (2006), [45]	CESO (2007), [23]	AmQSO (2008), [13]	CellularPSO (2009), [14]	mPSO (2010), [12]	CellularDE (2011), [16]	DHPSO (2012), [47]	DynPopDE (2013), [38]	CDEPSO
1	4.93±0.17(+)	5.07±0.17(+)	2.64±0.10(+)	1.04±0.00(+)	0.51±0.04(+)	2.79±0.19(+)	0.90±0.05(+)	1.53±0.07(+)	1.64±0.01(+)	-	<b>0.41±0.00</b>
5	2.07±0.08(+)	1.81±0.07(+)	2.15±0.07(+)	-	1.01±0.09(+)	1.94±0.18(+)	1.21±0.12(+)	1.50±0.04(+)	1.62±0.01(+)	1.03±0.13(+)	<b>0.97±0.01</b>
10	2.08±0.07(+)	1.80±0.06(+)	2.51±0.09(+)	1.38±0.02(+)	1.51±0.10(+)	1.93±0.08(+)	1.66±0.08(+)	1.64±0.03(+)	2.73±0.02(+)	1.39±0.07(+)	<b>1.22±0.01</b>
20	2.64±0.07(+)	2.42±0.07(+)	3.21±0.07(+)	1.72±0.02(+)	2.00±0.15(+)	2.73±0.12(+)	2.05±0.08(+)	2.46±0.05(+)	2.82±0.02(+)	-	<b>1.54±0.01</b>
30	2.63±0.08(~)	2.48±0.07(-)	3.64±0.07(+)	<b>1.24±0.01(-)</b>	2.19±0.17(-)	3.08±0.11(+)	2.18±0.06(-)	2.62±0.05(~)	3.97±0.03(+)	-	2.62±0.01
50	2.65±0.06(+)	2.50±0.06(+)	3.86±0.08(+)	<b>1.45±0.01(-)</b>	2.43±0.13(+)	3.34±0.07(+)	2.30±0.04(+)	2.75±0.05(+)	5.13±0.03(+)	2.10±0.06(-)	2.20±0.01
100	2.49±0.04(+)	2.36±0.04(+)	4.01±0.07(+)	<b>1.28±0.02(-)</b>	2.68±0.12(+)	3.48±0.11(+)	2.32±0.04(+)	2.73±0.03(+)	-	2.34±0.05(+)	1.54±0.01
200	2.44±0.04(+)	2.26±0.03(+)	3.82±0.05(+)	-	2.62±0.10(+)	3.37±0.08(+)	2.34±0.03(+)	2.61±0.02(+)	-	2.44±0.05(+)	<b>2.11±0.01</b>

"+" and "−" indicate a 0.05 level of significance with 49 degree of freedom by two-tailed *t*-test. "+" and "−" denote that the performance of CDEPSO is better than, worse than, and similar to that of the corresponding algorithm, respectively.

#### 4.4.2 Experiment on DOPs with different shift lengths

We investigate the performance of CDEPSO compared to nine other algorithms using MPB with different shift lengths,  $s \in \{0, 1, 2, 5\}$ . The offline error and standard error of the numerical results of the ten algorithms are presented in Table 4. We can see that, it became more difficult for the algorithms to track the global optima as the severity of the environmental changes increased. Our results indicate that the proposed algorithm is good at adapting to the environmental dynamics in terms of the shift severity.

**Table 4. Offline error ± standard error of algorithms on MPB with  $f=5000$ ,  $m=10$ ,  $D=5$ ,  $\lambda=0.0$ , and different shift lengths. The t-test results of comparing CDEPSO with contestant algorithms are shown in brackets. The best result for each DOP among the compared algorithms is highlighted in boldface.**

<i>s</i>	mCPSO (2006), [35]	mQSO (2006), [35]	SPSO (2006), [45]	CESO (2007), [23]	AmQSO (2008), [13]	CellularPSO (2009), [14]	mPSO (2010), [12]	CellularDE (2011), [16]	DHPSO (2012), [47]	CDEPSO
0	1.18±0.07(+)	1.18±0.07(+)	0.95±0.08(+)	0.85±0.02(+)	1.00±0.11(+)	0.93±0.09(+)	1.05±0.09(+)	0.94±0.09(+)	-	<b>0.83±0.00</b>
1	2.05±0.07(+)	1.75±0.06(+)	2.51±0.09(+)	1.38±0.02(+)	1.51±0.10(+)	1.93±0.08(+)	1.66±0.08(+)	1.64±0.03(+)	2.73±0.03(+)	<b>1.22±0.01</b>
2	2.80±0.07(+)	2.40±0.06(+)	3.78±0.09(+)	1.78±0.02(+)	2.09±0.08(+)	2.72±0.10(+)	2.54±0.11(+)	2.35±0.10(+)	3.29±0.01(+)	<b>1.74±0.01</b>
5	4.89±0.11(+)	4.24±0.10(+)	4.59±0.19(+)	<b>2.52±0.06(-)</b>	3.71±0.11(+)	5.68±0.18(+)	5.32±0.09(+)	4.20±0.13(+)	5.75±0.03(+)	3.56±0.01

“+” and “-” indicate a 0.05 level of significance with 49 degree of freedom by two-tailed  $t$ -test. “+”, “-” and “~” denote that the performance of CDEPSO is better than, worse than, and similar to that of the corresponding algorithm, respectively.

#### 4.4.3 Experiment on DOPs with different change intervals

Table 5 shows the performance comparison between CDEPSO and six other peer algorithms on DOPs with different change intervals,  $f \in \{500, 1000, 2500, 5000\}$ . In this experiment, the results of mQSO are reported from [44]. We can see from Table 5 that the performances of all algorithms improved with an increase in the change interval. We can also see that the results from CDEPSO were much better than the other algorithms. These results indicate that the proposed algorithm can quickly adapt to the environmental dynamics in terms of change frequency.

**Table 5. Offline error ± standard error of algorithms on MPB with  $m=10$ ,  $D=5$ ,  $s=1$ ,  $\lambda=0.0$ , and different change intervals. The  $t$ -test results of comparing CDEPSO with contestant algorithms are shown in brackets. The best result for each DOP among the compared algorithms is highlighted in boldface.**

$f$	mQSO (2006), [35]	AmQSO (2008), [13]	CellularPSO (2009), [14]	mQSO Rand-Rule (2010), [46]	mPSO (2010), [12]	CellularDE (2011), [16]	CDEPSO
500	9.62±0.34(+)	<b>5.37±0.42(-)</b>	9.42±0.21(+)	10.72±0.21(+)	7.19±0.23(+)	5.93±0.04(-)	6.57±0.02
1000	5.71±0.22(+)	4.56±0.40(+)	5.15±0.13(+)	6.35±0.10(+)	4.57±0.18(+)	4.59±0.03(+)	<b>4.00±0.01</b>
2500	3.12±0.14(+)	2.49±0.10(+)	2.80±0.10(+)	4.17±0.06(+)	2.66±0.16(+)	2.48±0.04(+)	<b>1.92±0.01</b>
5000	1.85±0.08(+)	1.51±0.10(+)	1.93±0.08(+)	3.25±0.04(+)	1.66±0.08(+)	1.64±0.03(+)	<b>1.22±0.01</b>

“+” and “-” indicate a 0.05 level of significance with 49 degree of freedom by two-tailed  $t$ -test. “+”, “-” and “~” denote that the performance of CDEPSO is better than, worse than, and similar to that of the corresponding algorithm, respectively.

Figure 6 shows the convergence behavior of the CDEPSO (in terms of offline and current error) for Scenario 2 of MPB. When the QUEST population is able to effectively locate the best peak in the fitness landscape, the current error is significantly reduced because of the exploitation of the TARGET population around the best found position.

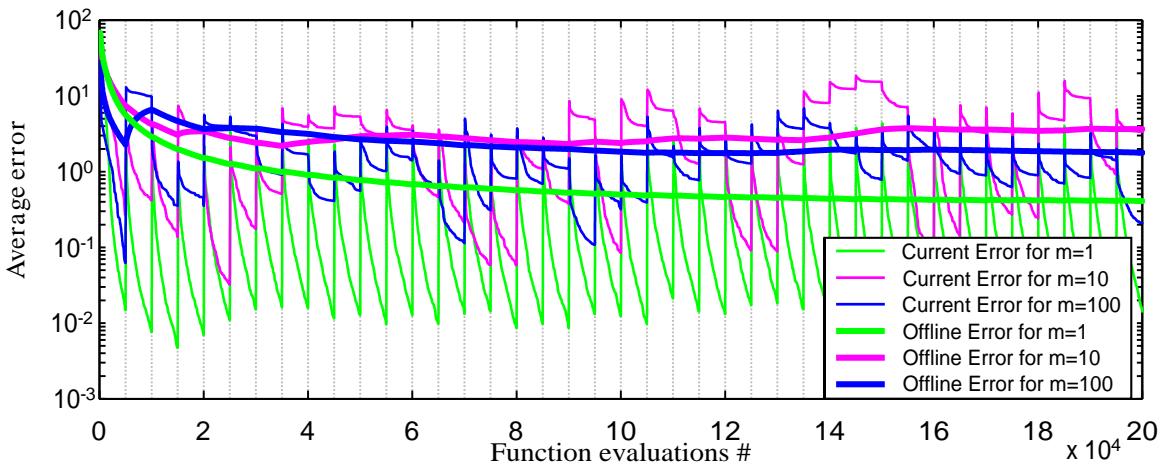


Figure 6. Offline error and current error for different DOPs with  $f = 5000$ ,  $D=5$ ,  $s=1$ , and  $\lambda=0.0$ . The vertical dotted lines in the figure indicate the start of a new period.

Note that we carried out all experiments in this paper using the default parameter settings of CDEPSO. Therefore, it is possible that more appropriate parameter values can improve the tracking ability of the proposed algorithm. Figure 7, shows the effect of varying the parameters  $F$  and  $C_r$  (in the range [0, 1]) for Scenario 2 of MPB with 10 peaks ( $m=10$ ). We can observe that the values of  $F$  and  $C_r$  have a significant effect on the behavior of CDEPSO. By using  $F=0.12$  and  $C_r=0.92$ , we have improved the average offline error of the CDEPSO to 0.91.

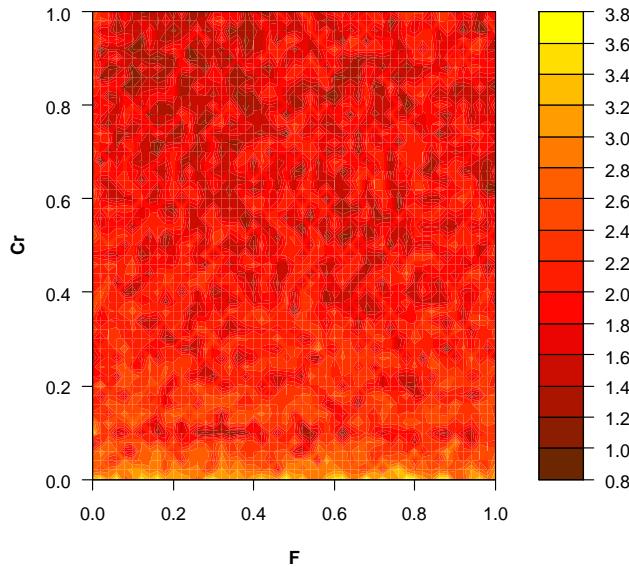


Figure 7. Average offline error of CDEPSO for varying  $F$  and  $C_r$  on Scenario 2 of MPB with  $m=10$

## 5. Conclusion and future works

This paper presents a bi-population hybrid approach for DOPs. The proposed algorithm consists of two equal-size populations. The first population (QUEST) follows ODE principles. It is responsible for locating multiple promising areas of the search space and preserving a certain level of diversity throughout the run. The second population (TARGET) uses PSO to exploit useful information in the vicinity of the best position found by the QUEST. Different mechanisms are used to enhance the performance of the algorithm for DOPs. Experimental results on the MPB demonstrate that the proposed approach is effective in dealing with DOPs. Different future works can be considered to promote the performance of the proposed algorithm. First, the values of the parameters of CDEPSO are based on other studies and/or some simple preliminary experiments. As we discussed at the end of Section 4, choosing appropriate values for  $F$  and  $C_r$  could improve the performance of the proposed algorithm. Therefore, a sensitivity analysis on the effect of parameters could be a direction for future research. Second, adaptively choosing the parameters according to the search progress of

CDEPSO would also be interesting. Third, it is possible to incorporate other mechanisms into CDEPSO to further improve its performance for more complex DOPs.

**Acknowledgement** The authors are grateful to V. Noroozi for providing the results of his algorithm. The authors also like to thank the anonymous associate editor and reviewers for their constructive comments to improve the quality and the clarity of the paper.

## 6. References

1. Vafashoar R, Meybodi M, Momeni Azandaryani A (2012) CLA-DE: a hybrid model based on cellular learning automata for numerical optimization. *Appl Intell* 36:735–748. doi: 10.1007/s10489-011-0292-1
2. Hasanzadeh M, Meybodi MR, Ebadzadeh MM (2013) Adaptive cooperative particle swarm optimizer. *Appl Intell* 39:397–420. doi: 10.1007/s10489-012-0420-6
3. Norouzzadeh MS, Ahmadzadeh MR, Palhang M (2012) LADPSO: using fuzzy logic to conduct PSO algorithm. *Appl Intell* 37:290–304. doi: 10.1007/s10489-011-0328-6
4. Blackwell TM (2005) Particle swarms and population diversity. *Soft Comput* 9:793–802. doi: 10.1007/s00500-004-0420-5
5. Jin Y, Branke J (2005) Evolutionary optimization in uncertain environments-a survey. *IEEE Trans Evol Comput* 9:303–317. doi: 10.1109/TEVC.2005.846356
6. Hu X, Eberhart RC (2002) Adaptive particle swarm optimization: detection and response to dynamic systems. *Proceedings of the 2002 Congress on Evolutionary Computation*. pp 1666–1670
7. Yang S (2008) Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. *Evol Comput* 16:385–416. doi: 10.1162/evco.2008.16.3.385
8. Yang S, Yao X (2008) Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans Evol Comput* 12:542–561. doi: 10.1109/TEVC.2007.913070
9. Branke J (1999) Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems. *Proceedings of the 1999 Congress on Evolutionary Computation*. Washington, DC, USA, pp 1875–1882
10. Blackwell T, Branke J (2004) Multi-swarm Optimization in Dynamic Environments. In: Raidl GR (ed) *Applications of Evolutionary Computing*. pp 489–500
11. Kamosi M, Hashemi AB, Meybodi MR (2010) A hibernating multi-swarm optimization algorithm for dynamic environments. *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*. pp 363–369
12. Kamosi M, Hashemi AB, Meybodi MR (2010) A new particle swarm optimization algorithm for dynamic environments. In: Panigrahi BK (ed) *Swarm, Evolutionary, and Memetic Computing*. pp 129–138
13. Blackwell T, Branke J, Li X (2008) Particle swarms for dynamic optimization problems. In: Blum, C (ed) *Swarm Intelligence*. Springer Berlin Heidelberg, pp 193–217
14. Hashemi A, Meybodi M (2009) Cellular PSO: A PSO for dynamic environments. In: Cai Z (ed) *Advances in Computation and Intelligence*. Springer Berlin Heidelberg, pp 422–433

15. Hashemi AB, Meybodi MR (2009) A multi-role cellular PSO for dynamic environments. Proceedings of 14th International CSI Computer Conference. Tehran, Iran, pp 412–417
16. Noroozi V, Hashemi A, Meybodi MR (2011) CellularDE: a cellular based differential evolution for dynamic optimization problems. In: Dobnikar A (ed) Adaptive and Natural Computing Algorithms. Springer Berlin Heidelberg, pp 340–349
17. Kianfar S, Meybodi MR (2012) Cellular Ant Colony Algorithm. Proceedings of 17th Annual CSI Computer Conference of Iran. Tehran, Iran, pp 45–50
18. Nabizadeh S, Rezvanian A, Meybodi MR (2012) Tracking Extrema in Dynamic Environment using Multi-Swarm Cellular PSO with Local Search. *Int J Electron Inform* 1:29–37.
19. Nabizadeh S, Rezvanian A, Meybodi MR (2012) A multi-swarm cellular PSO based on clonal selection algorithm in dynamic environments. 2012 International Conference on Informatics, Electronics & Vision (ICIEV). Dhaka, Bangladesh, pp 482–486
20. Yang S, Li C (2010) A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Trans Evol Comput* 14:959–974. doi: 10.1109/TEVC.2010.2046667
21. Li C, Yang S (2012) A general framework of multipopulation methods with clustering in undetectable dynamic environments. *IEEE Trans Evol Comput* 16:556–577. doi: 10.1109/TEVC.2011.2169966
22. Lung RI, Dumitrescu D (2010) Evolutionary swarm cooperative optimization in dynamic environments. *Nat Comput* 9:83–94.
23. Lung RI, Dumitrescu D (2007) A collaborative model for tracking optima in dynamic environments. IEEE Congress on Evolutionary Computation. pp 564–567
24. Thangaraj R, Pant M, Abraham A, Bouvry P (2011) Particle swarm optimization: Hybridization perspectives and experimental illustrations. *Appl Math Comput* 217:5208–5226. doi: 10.1016/j.amc.2010.12.053
25. Das S, Suganthan PN (2011) Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15:4–31. doi: 10.1109/TEVC.2010.2059031
26. Thomsen R (2004) Multimodal optimization using crowding-based differential evolution. Congress on Evolutionary Computation. pp 1382–1389
27. Kennedy J, Eberhart R (1995) Particle swarm optimization. IEEE International Conference on Neural Networks. pp 1942–1948
28. Nabizadeh S, Faez K, Tavassoli S, Rezvanian A (2010) A novel method for multi-level image thresholding using particle swarm Optimization algorithms. 2010 2nd International Conference on Computer Engineering and Technology (ICCET). pp V4–271–V4–275
29. Zheng Y-J, Chen S-Y (2013) Cooperative particle swarm optimization for multiobjective transportation planning. *Appl Intell* 39:202–216. doi: 10.1007/s10489-012-0405-5
30. Wang K, Zheng YJ (2012) A new particle swarm optimization algorithm for fuzzy optimization of armored vehicle scheme design. *Appl Intell* 37:520–526. doi: 10.1007/s10489-012-0345-0
31. Khan SA, Engelbrecht AP (2012) A fuzzy particle swarm optimization algorithm for computer communication network topology design. *Appl Intell* 36:161–177. doi: 10.1007/s10489-010-0251-2

32. Ali YMB (2012) Psychological model of particle swarm optimization based multiple emotions. *Appl Intell* 36:649–663. doi: 10.1007/s10489-011-0282-3
33. Masoud H, Jalili S, Hasheminejad SMH (2013) Dynamic clustering using combinatorial particle swarm optimization. *Appl Intell* 38:289–314. doi: 10.1007/s10489-012-0373-9
34. Soleimani-Pour M, Rezvanian A, Meybodi MR (2012) Finding a Maximum Clique Using Ant Colony Optimization and Particle Swarm Optimization in Social Networks. Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012). IEEE Computer Society, Washington, DC, USA, pp 58–61
35. Blackwell T, Branke J (2006) Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Trans Evol Comput* 10:459–472. doi: 10.1109/TEVC.2005.857074
36. Mendes R, Mohais AS (2005) DynDE: a differential evolution for dynamic optimization problems. The 2005 IEEE Congress on Evolutionary Computation. pp 2808–2815
37. Noroozi V, Hashemi AB, Meybodi MR (2012) Alpinist CellularDE: a cellular based optimization algorithm for dynamic environments. Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion (GECCO 2012). ACM, pp 1519–1520
38. Du Plessis MC, Engelbrecht AP (2013) Differential evolution for dynamic environments with unknown numbers of optima. *J Glob Optim* 55:73–99. doi: 10.1007/s10898-012-9864-9
39. Nickabadi A, Ebadzadeh M, Safabakhsh R (2012) A competitive clustering particle swarm optimizer for dynamic optimization problems. *Swarm Intell* 6:177–206. doi: 10.1007/s11721-012-0069-0
40. Nickabadi A, Ebadzadeh MM, Safabakhsh R (2011) A novel particle swarm optimization algorithm with adaptive inertia weight. *Appl Soft Comput* 11:3658–3670. doi: 10.1016/j.asoc.2011.01.037
41. Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. *IEEE Trans Evol Comput* 12:107–125. doi: 10.1109/TEVC.2007.895272
42. Shamos MI, Hoey D (1975) Closest-point problems. 16th Annual Symposium on Foundations of Computer Science, 1975. pp 151–162
43. (2008) The Moving Peaks Benchmark. <http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/>.
44. Yazdani D, Nasiri B, Sepas-Moghaddam A, Meybodi MR (2013) A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Appl Soft Comput* 13:2144–2158. doi: 10.1016/j.asoc.2012.12.020
45. Parrott D, Li X (2006) Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Trans Evol Comput* 10:440–458. doi: 10.1109/TEVC.2005.859468
46. Del Amo IG, Pelta DA, González JR (2010) Using heuristic rules to enhance a multiswarm PSO for dynamic environments. 2010 IEEE Congress on Evolutionary Computation (CEC). pp 1–8
47. Karimi J, Nobahari H, Pourtakdoust SH (2012) A new hybrid approach for dynamic continuous optimization problems. *Appl Soft Comput* 12:1158–1167. doi: 10.1016/j.asoc.2011.11.005



**Javidan Kazemi Kordestani** received the B.S. in Computer Engineering (Software Engineering) from Islamic Azad University of Karaj, Iran in 2008 and his M.S. in Computer Engineering (Artificial Intelligence) from Islamic Azad University of Qazvin, Iran in 2012. His current research interests include evolutionary computation, dynamic optimization problems and learning systems.



**Alireza Rezvanian** received the B.S. degree in Computer Engineering from Bu-Ali Sina University of Hamedan, Iran, in 2007, and the M.S. degree in Computer Engineering from Islamic Azad University of Qazvin in 2010. He is currently pursuing Ph.D. degree in Computer Engineering at the Computer Engineering & Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Iran. His research activities include soft computing, evolutionary algorithms, complex social networks, and learning systems.



**Mohammad Reza Meybodi** received the B.S. and M.S. degrees in Economics from the Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include, channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.

This document was created with Win2PDF available at <http://www.daneprairie.com>.  
The unregistered version of Win2PDF is for evaluation or non-commercial use only.