A note on the Exclusion operator in multi-swarm PSO algorithms for dynamic environments

Javidan Kazemi Kordestani[1], Mohammad Reza Meybodi[2,*], Amir Masoud Rahmani[1]

[1]*Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*

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

javidan.kazemi@gmail.com, mmeybodi@aut.ac.ir[*], rahmani@srbiau.ac.ir

Abstract

The exclusion operator is a key component in separating the search territory of each population in multi-population optimization algorithms for unconstraint continues dynamic optimization problems (DOPs) with the aim of maintaining the overall diversity of the population and avoiding redundant search. Although extensively used by the researchers, the role of exclusion has been barely studied in detail. Therefore, in this paper, we solely study the role of exclusion as a part of multi-population methods in DOPs. For this purpose, a comprehensive review of the various exclusion strategies reported in the literature is provided. Four strategies are also introduced to reduce the shortcomings of exclusion operator. Experimental results show that proposed strategies compared to other schemes such as reinitialized midpoint check, hill-valley detection with three checkpoints, and merging information of collided populations have the same or even higher ability to improve the performance of the multi-swarm PSO algorithms in moving peaks benchmark.

Keywords: exclusion, dynamic optimization problems, multi-swarm PSO, multi-population methods, moving peaks benchmark.

1. Introduction

The multi-population approach has shown to be very effective for handling dynamic optimization problems (DOPs), especially for multimodal fitness landscapes. The success of this approach can be contributed to three reasons [1]:

1. As long as different populations search in different sub-areas in the fitness landscape, the overall population diversity can be maintained at the global level.
2. It is possible to locate and track multiple changing optima simultaneously. This feature can facilitate tracking of the global optimum, given that one of the being-tracked local optima may become the new global optimum when changes occur in the environment.
3. It is easy to extend any single-population approach, e.g. diversity increasing/maintain schemes, memory schemes, adaptive schemes, etc., to multi-population version.

According to the above remarks, one of the key challenging task of addressing DOPs using multi-population approach is to keep each population on a distinct sub-area in the fitness landscape.

Most researchers have been using exclusion to differ the search areas covered by the populations and preventing them from redundant search [2]–[13]. Despite its advantage, the exclusion operator in its generic form suffers from various drawbacks that can be summarized as follows:

- The performance of multi-population methods is highly sensitive to the exclusion radius and identifying a proper exclusion radius for each population is a challenging task.
- exclusion operator is unable to properly keep populations on two distinct peaks that are located within their exclusion radius. As a result, the weaker population will be restarted in the fitness landscape and one peak remains unpopulated.
- the information obtained by the weaker population will be discarded due to the restart.
- the excluded population may frequently converge to the same peak or other populated peaks, triggering another exclusion which wastes the precious computing resources.

There have been several proposals to modify some aspects of exclusion to somehow rectify the above disadvantages. The aim of this paper is to provide a survey of these works, attempting to classify them based on the type of the challenges they resolved. To the best of our knowledge, this is the first survey on exclusion. While there are other factors that can affect the performance of the multi-population algorithms in DOPs, this paper only focuses on exclusion operator and it is not intended to cover all state-of-the-art multi-population algorithms such as CESO [14], ESCA [15] and CDEPSO [16]. New modifications related to exclusion are also proposed and compared with the current methods. Experimental results show that the proposed exclusion schemes can improve the performance of the multi-swarm PSO in DOPs.

The rest of this paper is structured as follows: Section 2 gives an in-depth overview of the exclusion and its variants. In Section 3, we explain the multi-swarm optimization for dynamic environments as a case study for implementing our proposals. Our proposed modifications are presented in Section 4. Experimental study regarding the parameters sensitivity analysis, effect of applying different exclusion related schemes, and comparison with other approaches are presented in Section 5. Finally, conclusions and future works are given in Section 6.

2. Exclusion

The term “exclusion” was originally used by Blackwell and Branke in [2] to show the local interaction between a number of swarms in a multi-swarm optimization method with an atomic structure for DOPs. Later, many researchers borrowed the concept of exclusion to develop other multi-population methods for DOPs.

Since the initial introduction of exclusion by Blackwell and Branke, several attempts have been made by researchers to improve this operator. Globally, exclusion related variations in the literature can be classified and discussed in four main categories: They are approaches for setting the exclusion radius, approaches for detecting situations where collided populations are standing on distinct optima, approaches for preventing the reinitialized population from redundant search and approaches for saving the search history of the weaker population. The rest of this section describes each variation in detail.

2.1. Setting the exclusion radius

The first category of studies on exclusion is focused on setting the parameter exclusion radius (r_{excl}) to determine the search area of each population in fitness landscape. These methods can be further categorized into two groups: (a) methods that tune the parameter r_{excl} and (b) methods that control the parameter r_{excl}.

2.1.1. Tuning the parameter r_{excl}

Most existing multi-population methods try to tune the parameter r_{excl} by finding a good value for it before the actual run of the algorithm and then running the algorithm using this value, which remains fixed during the run. This is done by experimenting with different values of r_{excl} and selecting the one that gives the best results on the tested DOPs at hand.

Although hand tuning the parameters based on experimentation is a common practice in evolutionary computation, it has some inherent drawbacks when applying on r_{excl} that can be summarized as follows:

- The process of parameter tuning for r_{excl} is time consuming, especially if the search domain is continuous.
- A good value for parameter r_{excl} depends on different criteria like the number of peaks, search space range, search space dimensionality, shape of the peaks, and distribution of the peaks in the fitness landscape. Moreover, the suitable value of r_{excl} can be varied in different methods. For example, the exclusion radius of all populations is set to 30 in mQSO [3] and HmSO [6], but 25 in FMSO [17].
- For a given DOP, the selected value for parameter r_{excl} is not necessarily optimal, even if the attempt made for setting the r_{excl} was significant.
- Since the value of r_{excl} is fixed during the run, tuning the value of r_{excl} is not an effective approach for DOPs with varying number of optima.

2.1.2. Controlling the parameter r_{excl}

To overcome the above shortcomings, a few studies have been conducted with aim of controlling the parameter r_{excl} during the run. In the preliminary work done by Blackwell and Branke, the linear diameter of the basin of attraction of any peak was used to predict the optimal value for r_{excl}, based on the assumption that all p peaks are evenly distributed in search space X^d, as follows [3]:

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

where X is the search space range of the d dimensions, and p is the number of existing peaks in the landscape. The disadvantage of this adaptation is that it requires the information about the environment, e.g. number of optima, that may not be available in real-world applications.

To overcome the above disadvantage, a number of adaptations have been proposed for situations when the information on the number of optima is not available. For example, Blackwell [18] suggests to estimate the exclusion radius based on the number of current populations in the search space by modifying the Eq. 1 as follows:

$$r_{excl} = \frac{X}{2M^{1/d}} \tag{2}$$

where M is the current number of populations in the search space.

Rezazadeh et al. [19] used the fuzzy c-means (FCM) clustering method for adaptive determination of the exclusion radius with respect to position of the particles in the search space. In this approach, in each iteration, all particles are first clustered using FCM. Then the distance between centers of the clusters are used to adjust the exclusion radius as follows:

$$mindis_i = \text{minimum}\left(dist\left(center_i, center_j\right)\right),\ \ where\ 1 \leq i < j \leq n, i \neq j \tag{3}$$

$$r_{excl} = \frac{mean(\sum_{i=1}^{n} mindist_i)}{2^{-n}+2} \tag{4}$$

In above equations, $dist$ is a function which calculates the Euclidian distance between centers of the clusters i and j in the d-dimensional search space using Eq. 5, n is the total number of clusters determined by FCM, and $mindist_i$ denotes the distance from ith cluster to the closest cluster in the search space.

$$dist(i,j) = \sqrt{\sum_{d=1}^{D}\left(x_i^d - x_j^d\right)^2} \tag{5}$$

This adaptation also eliminated the need to have information about the search domain for calculating r_{excl}.

2.2. Detecting situations where collided populations are standing on distinct optima

One of the technical drawbacks of exclusion is that it ignores the situations when two populations stand on two distinct but extremely close peaks (i.e. within the exclusion radius of each other). In these cases, exclusion operator simply removes the worst population and leaves one of the peaks unpopulated. A group of studies has tried to address this issue by adding an extra routine to the exclusion which is executed when a collision between two populations is detected. For example, du Plessis & Engelbrecht [20] proposed re-initialization midpoint check (RMC) to detect whether different peaks are located within the exclusion radius of each other. In RMC, which is a simpler version of hill-valley detection [21], once a collision occurs between two populations, the fitness of the midpoint on the line between the best individuals in each population is evaluated. If the fitness of the midpoint has a lower value than the fitness value of the best individuals of both populations, it implies that the two populations reside on distinct peaks and that neither should be reinitialized. Otherwise, the worst performing population will be reinitialized in the search space. Although RMC is effective, as pointed by the authors, it is unable to correctly detect all extremely close peaks.

Zuo and Xiao [22] applied hill-valley detection with three checkpoints to determine whether two colliding populations are located on different peaks. In their approach, three points between the best solutions of the collided populations x and y are examined. Thereafter, if there exists a point $\mathrm{z} = \mathrm{c} \cdot \mathrm{x} + (1 - \mathrm{c}) \cdot \mathrm{y}$ for which $f(\mathrm{z}) < min\{f(x), f(y)\}$, where $\mathrm{c} \epsilon \{0.05, 0.5, 0.95\}$, then two populations are on different peaks and they remain unchanged. Otherwise, they are resided on the same peak.

2.3. Preventing the reinitialized population from redundant search

Exclusion in its native form simply restarts the worst-performing population when an overlapping between two populations is detected. However, there is a great chance that the reinitialized population is again moving toward the other populations or populated peaks, triggering another exclusion. This can waste the precious computing resources that could be otherwise used to explore the non-visited areas of the search space or to further exploit the previously found peaks. Therefore, a number of studies devised some strategies to overcome the mentioned limitation. For instance, in [22], for each population marked to be re-initialized by the exclusion, a part of its individuals are generated by an opposition-based initialization operator [23]. This operator can improve the capability of the algorithm in searching unexplored areas.

In [24], du Plessis & Engelbrecht used a mechanism to remove populations that are frequently reinitialized by the exclusion operator to stop unproductive search.

2.4. Saving the information obtained the weaker population

Another disadvantage of basic exclusion is that the information obtained by the weaker population will be lost due to restart. This can waste the precious computing resources and reduce the performance of the multi-population algorithms. Some authors proposed modifications to alleviate the information loss caused by re-initializing the worst-performing population upon collision. For instance, in [25], when the overlapping among two populations is detected, before restarting the worst population, the best individual of the worst population replaces the worst individual of the best population, if it is fitter. In [7], if the degree of overlapping among collided populations becomes greater than a pre-defined threshold, two populations will be merged. This way, it is possible that the information obtained by the weaker population is maintained in the next generation.

2.5. Other ways to classify exclusion variations

Besides the above classification, which was aimed at highlighting the weaknesses of the exclusion, we can also use the following classification criteria:

- Application of exclusion
    - Exclusion is applied between populations: Populations compete with each other and no more than one population is allowed to search in the same area [2]–[13].
    - Exclusion is not applied between populations: Populations are allowed to search in the same area [14]–[16].
- The way the algorithm reacts to the weaker population
    - Random restart with uniform distribution: The weaker population will be reinitialized into the search space according to the uniform distribution [2]–[12].
    - Restart using heuristic methods: The whole or a part of weaker population will be reinitialized using a heuristic method to detect unexplored areas [22], [25], [26].
    - Removing the weaker population: The weaker population will be removed [13], [24].
    - Inactivating the weaker population: The weaker population will be temporarily inactivated [11].
- Population level vs. individual level

- Population: The exclusion is applied at population level [2]–[13].
- Individuals: The exclusion is applied at individual level [26].

In the next section we first review multi-swarm optimization as a case study for solving DOPs, and then we discuss various options related to exclusion.

3. multi-swarm optimization in dynamic environments

PSO is a versatile population-based stochastic optimization method which was first proposed by Kennedy and Eberhart [18] in 1995. 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 denotes 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:

$$\vec{v}_i(t+1) = \omega\vec{v}_i(t) + c_1r_1[\vec{p}_i(t) - \vec{x}_i(t)] + c_2r_2[\vec{p}_g(t) - \vec{x}_i(t)] \tag{6}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{7}$$

where ω is an inertia weight that governs the amount of speed preserved from the previous iteration. 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 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 is shown in Figure 1.

```
01. Setting parameters;
02. Generate the initial swarm of the particles with random positions and velocities;
03. Evaluate the fitness of each particle of the swarm;
04. repeat
05.     for each particle $i$ in the swarm do
06.         update particle $i$ according to Eqs. (6), (7);
07.         if(fitness($\vec{p}_i$)<fitness($\vec{x}_i$)) then
08.             $\vec{p}_i$:= $\vec{x}_i$;
09.             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 1. Pseudo-code for canonical PSO

Although PSO has been successfully applied into many stationary problems [27], it should undergo a certain adjustment to work well in dynamic environments. The reason is that the dynamic behavior of DOPs poses additional challenges to the PSO, among which *diversity loss* is by far more serious. Diversity loss arises due to the tendency of the swarm to converge to a single optimum. Consequently, when a change occurs in environment, the number of function evaluations required for a partially converged swarm to re-diversify and re-converge to the shifted optimum is quite deleterious to the performance of PSO [28].

To overcome the above mentioned issue, Blackwell and Branke [3] proposed an effective multi-swarm PSO (mPSO). The main idea of mPSO is to establish a mutual repulsion among several fixed-size swarms to place them over different promising areas of the search space. Figure 2 summarizes the main steps of the mPSO for dynamic environments.

In order to maintain the global diversity of the mPSO, the authors introduced two operators to establish two forms of interactions between swarms. First operator named *exclusion*, which is triggered when two swarms are collided (i.e. searching in the same area of the search space), and prevent them from settling on the same peak by reinitializing the worst performing swarm in the search space. Second operator referred to as *anti-convergence* which is triggered whenever all swarms converged, and removes the worst swarm from its peak and reinitialize it in the search space. The convergence of a population is occurred if its size (i.e. the maximum distance between any two individual in the population in all dimensions) becomes smaller than a specified convergence radius r_{conv}. With this operator, there is at least one swarm monitoring the search space with the aim to locate new peaks.

Two versions of the mPSO have been implemented by the authors: mCPSO and mQSO. mCPSO and mQSO differ in a way they introduce diversity to the swarms.

In mCPSO, each swarm is composed of a number of *neutral* particles and a number of *charged classical* particles. Neutral particles in any swarm update their velocity and position according to the principles of pure PSO. On the other hand, charged classical particles in each swarm move in the same way as neutral particles, but are also mutually repelled from other charged particles residing in their own swarm as follows:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) + \sum_{j=1, j\neq i}^{N^+} \frac{Q_i Q_j}{\left|\vec{x}_i(t) - \vec{x}_j(t)\right|^3} \left(\vec{x}_i(t) - \vec{x}_j(t)\right) \tag{8}$$

where $Q_k > 0$ is an extra, constant parameter that determines the charge of the particle k, and N^+ is the number of the charged particles. Therefore, charged particles can maintain the diversity among the particles in a swarm.

In mQSO, instead of having charged particles, each swarm contains a number of *quantum* particles. Quantum particles do not move according to the PSO dynamics but change their positions in a hypersphere with radius r_{cloud} around the center of the best particle of the swarm ($\vec{p}_g$) according to a random distribution (*RD*) as follows:

$$\vec{x}_i(t+1) = RD(\vec{p}_g(t), r_{cloud}) \tag{9}$$

Consequently, they never converge, but provide the swarm with a sustainable level of diversity needed for capturing the shifting optimum.

Among mCPSO and mQSO, mQSO has attracted quite a deal of attention due to its effectiveness. In addition, as pointed by the authors, "CPSO suffers from quadratic complexity, arising from Coulomb repulsions which are calculated between all pairs of particles in the charged sub-swarm" [3]. The general framework for mPSO is illustrated in Figure 2. The complete detailed description and analysis of the mPSO can be found in [3].

```
01. Setting the parameters $r_{excl}$, $r_{conv}$, $M$, $pop\_size$; /*Set the exclusion radius, convergence radius, multi-
    population cardinality and the size of each population*/
02. Randomly initialize $M$ populations of individuals in the search space; /*Initialize populations into the search
    space*/
03. while termination condition is not met do
04.     Apply exclusion operator; /*if ($r_{excl}=0$) then exclusion is off */
05.     Apply anti-convergence operator; /*if ($r_{conv}=0$) then anti-convergence is off */
06.     Detect change of the environment and react to it;
07.     for each swarm $m$ do
08.         Evolve particles based on their type (neutral, quantum or charged);
            Evaluate the fitness of each particle;
09.         Update $\vec{p}_i$ and $\vec{p}_g$;
10.     end
11. end
```

Figure 2. Template of the multi-swarm PSO for DOPs

4. Proposed methods

As stated in Section 2, one major disadvantage of exclusion is that it leads to wasting computing resources (e.g. function evaluations). Keeping this in mind, all proposed modifications were designed with the aim of improving the efficiency of the mPSO by reducing the wastage of computing resources due to exclusion. To reach to this objective, the following approaches are proposed:

(1) avoiding populations from moving toward positions where they were excluded lately;

(2) preventing populations from moving toward populated positions;

(3) introducing schemes to alleviate the effect of applying exclusion on two collided populations when they stand on distinct peaks;

(4) using re-initialization of the excluded population as an opportunity to further explore non-visited areas of the search space;

In the rest of this section we describe our methods based on the above approaches.

4.1. Algorithm 1

In the native form of exclusion, when a collision is detected between two populations, the weaker population is re-initialized into the search space according to the uniform distribution. This way, there is a chance that the re-initialized population is converging to the same positions. Regarding the mPSO algorithm, we can assume that the restarted swarm i may re-converge to the position where it was excluded lately. The first modification aims at avoiding populations from moving to the position where they were excluded lately. Following this idea, we modify the velocity update equation for neutral particles as follows:

$$\vec{v}_i(t+1) = \omega\vec{v}_i(t) + c_1 r_1[\vec{p}_i(t) - \vec{x}_i(t)] + c_2 r_2[\vec{p}_g(t) - \vec{x}_i(t)] - c_3 r_3[\vec{p}_{\psi_i}(t) - \vec{x}_i(t)] \tag{10}$$

where c_3 is used to adjust the degree of reduction in the tendency of neutral particles to move toward the position where they were excluded lately. r_3 is a random variable drawn with uniform probability from [0,1]. Finally, $\vec{p}_{\psi_i}$ is the position where the swarm i were excluded lately. Therefore, Algorithm 1 differs from mQSO in that neutral particles are updated according to Eq. 10. In short, the Algorithm 1 performs as follows:

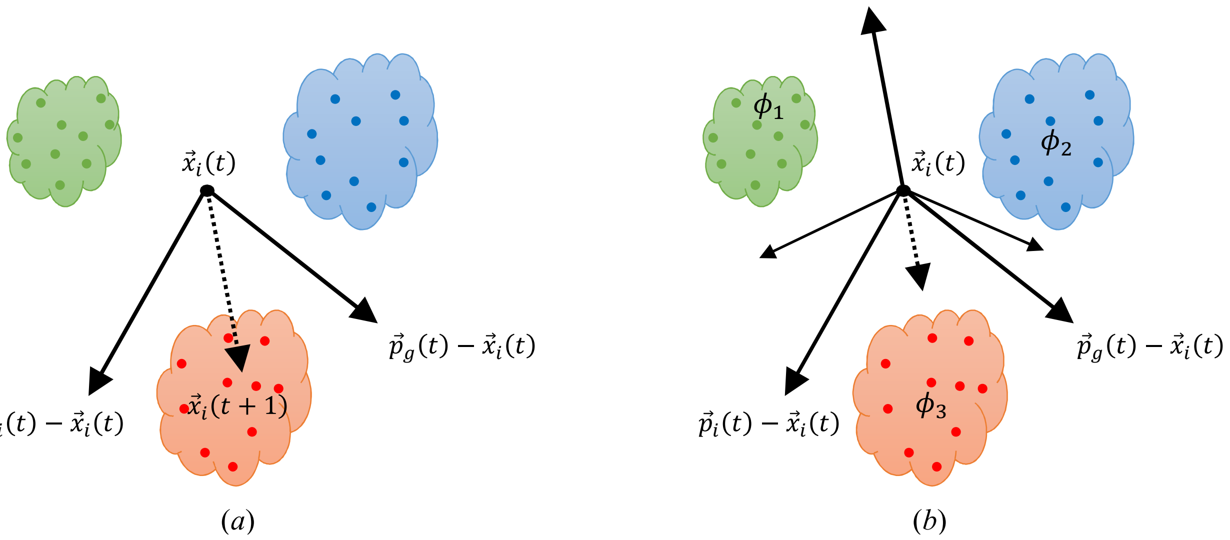
i. At the beginning of the run, randomly initialize sub-populations into the search space.
ii. Apply exclusion and anti-convergence.
iii. If a change in the environment is detected, re-evaluate all sub-populations.
iv. Evolve the sub-populations using the principles of mQSO except that neutral particles are updated according to Eq. (10).
v. Return to step (ii).

4.2. Algorithm 2

Another reason for wastage of computing resources is if a population is converging toward a currently populated peak. To address this issue, the second strategy, which is an extension of algorithm 1, is to prevent populations from moving toward converged populations. This can be obtained by modifying the Eq. 4 as follows:

$$\vec{v}_i(t+1) = \omega\vec{v}_i(t) + c_1r_1[\vec{p}_i(t) - \vec{x}_i(t)] + c_2r_2[\vec{p}_g(t) - \vec{x}_i(t)] - c_3r_3\sum_{j=1}^{n}[\vec{p}_{\phi_j}(t) - \vec{x}_i(t)] \tag{11}$$

where c_3 denote the *prevention factor* that is used to adjust the degree of reduction in the tendency of neutral particles to move toward the best position captured by the other swarms. r_3 is a random variable drawn with uniform probability from [0,1]. Finally, $\vec{p}_{\phi_j}$ is the best position found so far by the swarm ϕ_j which its diameter is less than the r_{excl}. Figure 3 shows the movement of neutral particles in the mQSO and Algorithm 2.



$\vec{x}_i(t)$
$\vec{p}_g(t) - \vec{x}_i(t)$
$\vec{p}_i(t) - \vec{x}_i(t)$
$\vec{x}_i(t+1)$
(a)
ϕ_1
ϕ_2
$\vec{x}_i(t)$
$\vec{p}_g(t) - \vec{x}_i(t)$
$\vec{p}_i(t) - \vec{x}_i(t)$
ϕ_3
(b)

Figure 3. Comparison between the movement of neutral particles in (*a*) mQSO and (*b*) Algorithm 2.

As can be observed in Figure 3, the proposed method prevents the populations from moving toward captured positions by other populations. Similar to Algorithm 1, the only difference between Algorithm 2 and mQSO is in the evolution of neutral particles. The Algorithm 2 is summarized as follows:

i. At the beginning of the run, randomly initialize sub-populations into the search space.
ii. Apply exclusion and anti-convergence.

iii. If a change in the environment is detected, re-evaluate all sub-populations.
iv. Evolve the sub-populations using the principles of mQSO except that neutral particles are updated according to Eq. (11).
v. Return to step (ii).

4.3. Algorithm 3

Once two populations are located in the exclusion radius of each other, the native exclusion strictly react to the weaker population by reinitializing it into the search space. This reaction is a bit harsh, since two populations may search in the same area haphazardly or they may stand on close but distinct peaks. Therefore, it is reasonable to consider that as populations come closer to each other it is more probable that they are standing on the same peak. Accordingly, here we propose a probabilistic model of the exclusion operator as Algorithm 3. we first define an exclusion probability around the best individual of each population as follows:

$$p_{excl} = \left(\frac{r_{excl} - dist(m_1, m_2)}{r_{excl}}\right)^{\alpha} \qquad (12)$$

where r_{excl} is the exclusion radius, $dist(m_1, m_2)$ represents the Euclidean distance between the best position found by any collided populations m_1 and m_2. Finally, α is a regularization parameter that determines the distribution of the p_{excl} around the best individual of population. Figure 4 shows the difference between native exclusion and the proposed probabilistic exclusion.

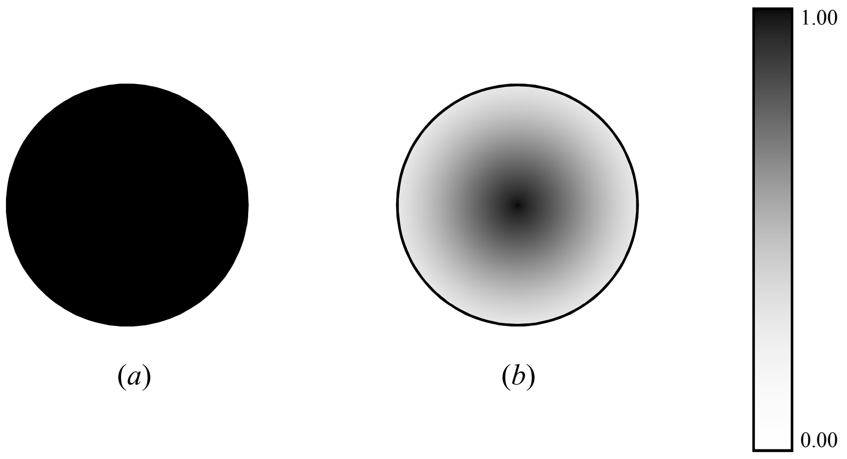


Figure 4. Schematic comparison between the probability of exclusion in (*a*) native exclusion and (*b*) the proposed stochastic exclusion with $\alpha = 1$. The colorbar scale maps tonalities with the probability of exclusion.

Regarding the probabilistic model in Figure 4, as two populations come closer to the center of each other, the probability of exclusion is also increases. Therefore, when two populations violate the search territory of each other, the occurrence of the exclusion is determined with regard to p_{excl}. The main steps of probabilistic exclusion are shown in Figure 5.

```
for each pair of swarm $m \neq n$ do
    if swarm attractor $p_{ng}$ is within $r_{excl}$ of $p_{mg}$ then
        Calculate $p_{excl}$ according to Eq. 5;
        Set $r = uniformRand(0,1)$;
        if $(r < p_{excl})$ then
            if $\left(fitness(p_{ng}) < fitness(p_{mg})\right)$ then
                Mark swarm $n$ for exclusion;
            else
                Mark swarm $m$ for exclusion;
            end-if
        end-if
    end-if
end-for
```
Figure 5. Pseudo-code of probabilistic exclusion

Bearing in mind the inner working of probabilistic exclusion, the Algorithm 3 performs as follows:

i. At the beginning of the run, randomly initialize sub-populations into the search space.
ii. Apply probabilistic exclusion according to pseudo-code in Figure 5.
iii. Apply anti-convergence.
iv. If a change in the environment is detected, re-evaluate all sub-populations.
v. Evolve the sub-populations using the principles of mQSO.
vi. Return to step (ii).

4.4. Algorithm 4

In native exclusion operator, when a collision is detected between two populations, the weaker population will be restarted according to a uniform distribution. The fourth modification aims at scattering reinitialized population far from the existing ones to fulfill a two-fold objective: (*a*) exploring the non-visited areas of the search space, (*b*) decreasing the probability of converging to the previously found positions. To this end, we sequentially initialize the individuals of the weaker population into the search space one by one according to Figure 6. At each step, a new individual is randomly generated in the search space. Afterwards, the distances between the generated individual and all the other individuals are calculated. If all distances are greater than r_{excl}, the generated individual will be accepted. This process will be continued until all the individuals of the weaker population reinitialized into the search space. This way we can be sure that all the individuals of newly created population is at least r_{excl} far from the other individuals.

```
01. for each particle $i$ in excluded population $m$ do
02.     $success = FALSE$;
03.     while $success = FALSE$ do
04.         $\vec{x}_i = lb + uniformRand(0,1) \times (ub - lb)$;//$ub$ and $lb$ are upper and lower bounds of search space
05.         for each swarm $n \neq m$ do
06.             for each particle $j$ of swarm $n$ do
07.                 $distance = EuclideanDistance(\vec{x}_i, \vec{x}_j)$;
08.                 if $(distance < r_{excl})$ then
09.                     Go to line 4;
10.                 end-if
11.             end-for
12.         end-for
13.         $success = TRUE$;
14.     end while
15. end-for
```

Figure 6. Template for the proposed re-initialization method

The Algorithm 6 is summarized as follows:

i. At the beginning of the run, randomly initialize sub-populations into the search space.
ii. Apply exclusion and reinitialize the weaker populations according to pseudo-code in Figure 6.
iii. Apply anti-convergence.
iv. If a change in the environment is detected, re-evaluate all sub-populations.
v. Evolve the sub-populations using the principles of mQSO.
vi. Return to step (ii).

5. Experimental study

In order to suitable analyze the impact of our proposal, in this section we described the results obtained from several computational experiments. First, we introduce the technical aspects of the artificial DOP used for testing the algorithms. The performance measures employed for assessing the algorithms are described later. Finally, we present and discuss the results we obtained in the experiments.

5.1. Dynamic test function

In what follows we describe the properties of the dynamic test function used in this study. Although several options exist, we have considered *Moving Peaks Benchmark* (MPB) [29]. MPB is indeed one of the most widely used dynamic benchmarks in the literature, which is highly regarded due to its configurability. Here, our goal is not proposing new optimization algorithms for DOPs to beat any state-of-art algorithm or to show the power of our optimizers for complex DOPs; instead, we are more interested on studying how exclusion operator impact the performance of multi-population algorithms when applying for dynamic environments.

MPB is a real-valued dynamic environment with a *D*-dimensional landscape consisting of *m* peaks, where the height, width and position of each peak are changed slightly every time a change occurs in the environment [30]. Different landscapes can be defined by specifying the shape of the peaks and some other parameters. A typical peak shape is conical which is defined as follows:

$$f(\vec{x},t) = \max_{i=1,\dots,m} H_t(i) - W_t(i)\sqrt{\sum_{j=1}^{D}(x_t(j) - X_t(i,j))^2} \tag{13}$$

where $H_t(i)$ and $W_t(i)$ are the height and width of peak i at time t, respectively. The coordinates of each dimension $j \in [1, D]$ of peak i at time t are expressed by $X_t(i,j)$, while D is the problem dimensionality. A typical change of a single peak can be modeled as follows:

$$H_{t+1}(i) = H_t(i) + height_{severity}.\sigma_h \tag{14}$$

$$W_{t+1}(i) = W_t(i) + width_{severity}.\sigma_w \tag{15}$$

$$\vec{X}_{t+1}(i) = \vec{X}_t(i) + \vec{v}_{t+1}(i) \tag{16}$$

$$\vec{v}_{t+1}(i) = \frac{s}{|\vec{r}+\vec{v}_t(i)|}\left((1-\lambda)\vec{r} + \lambda\vec{v}_t(i)\right) \tag{17}$$

where σ_h and σ_w are two random Gaussian numbers with zero mean and standard deviation one. Moreover, the shift vector $\vec{v}_{t+1}(i)$ is a 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. Parameter $\lambda \in [0.0, 1.0]$ specifies the correlation of each peak's changes to the previous one. This parameter determines the trajectory of changes, where λ=0 means that the peaks are shifted in completely random directions and λ=1 means that the peaks always follow the same direction, until they hit the boundaries where they bounce off.

In this paper we consider the so-called *Scenario 2* which is the most widely used configuration of MPB. Unless stated otherwise, the MPB's parameters are set according to the values listed in Table 1 (*default values*). In addition, to investigate the effect of the environmental parameters (i.e. the change severity, change period, number of peaks, number of dimensions, number of peaks, and peak shape) on the performance of the proposed approach, various experiments were carried out with different combinations of other tested values listed in Table 1.

Table 1. Parameter settings for the moving peaks benchmark

Parameter	Default values	Other tested values
Number of peaks (m)	100	1,5, 10, 20, 30, 40, 50, 100, 200
Height severity	7.0	
Width severity	1.0	
Peak function	cone	sphere
Number of dimensions (D)	5	
Height range (H)	$\in$ [30, 70]	
Width range (W)	$\in$ [1, 12]	
Standard height (I)	50.0	
Search space range (A)	$[0, 100]^D$	
Frequency of change (f)	5000	500, 1000, 2500, 10000
Shift severity (s)	1.0	3.0, 5.0
Correlation coefficient (λ)	0.0	
Basic function	No	

5.2. Performance measures

For measuring the efficiency of the tested algorithms, we considered two performance measures in our study: offline error [31] and best-error-before-changes [32].

5.2.1. Offline error

The first used measure is the performance measure suggested by Branke and Schmeck [31] which is defined as the average of the smallest error found by the algorithm in every time step:

$$E_{off} = \frac{1}{T}\sum_{t=1}^{T} e_t^* \tag{18}$$

where T is the maximum number of FEs so far and e_t^* is the minimum error obtained by the algorithm at the time step t. We assume that a function evaluation made by the algorithm corresponds to a single time step.

5.2.2. Best error before change

The other measure, which was first proposed in [32] as *accuracy* and later named as *best error before the change* in [33], is calculated as the average of the minimum fitness error achieved by the algorithm at the end of each environment, that is, just before a new change:

$$E_{bbc} = \frac{1}{K}\sum_{k=1}^{K}(h_k - f_k) \tag{19}$$

where f_k is the fitness value of the best solution obtained by the algorithm before the kth change occurs, h_k is the optimum value of the kth environment and K is the total number of environments. In this study, we use both measures to evaluate the performance of the proposed algorithms.

5.3 Experiment results

The experiments were organized in order to study:

1. different parameter settings for the proposed exclusion schemes;
    a. Effect of c_3 on the performance of Algorithm1 and Algorithm 2.
    b. Effect of parameter α on the behavior of Algorithm 3.
2. the effect of varying the environmental parameters, i.e. change frequency, shift severity, number of peaks and peak function;
3. the performance of the proposed modifications against other existing variants. The involved methods include:

   a. Transferring the information of the best individual of the weaker population to the worst individual of the stronger population, if it is fitter (B/W) [25].
   b. Re-initialization midpoint check (RMC) [20].
   c. Hill-valley detection with three checkpoints (HVD3P) [22].
4. and the benefits of combining different exclusion scheme.

In all experiments we performed 100 independent runs with different random seeds for the problem and algorithm. Specifically, each run finished when the algorithm reaches 500000 FEs. Unless otherwise stated, the results are reported in terms of average offline error and standard error.

In order to show the significant statistical difference between best-performing modification(s) and mQSO, the Wilcoxon rank sum test (Derrac et al. 2011; Wilcoxon 1945) is used for independent samples at the 0.05 significance level. The outcomes of the Wilcoxon rank-sum test revealing that no significant differences exist between the algorithms, are highlighted with an asterisk symbol.

The parameters setting of the mQSO are as shown in Table 2.

Table 2. Default parameter settings for the proposed mQSO variants

Parameter	Default value
Number of swarms	10
Number of neutral particles	5
Number of quantum particles	5
ω	0.729
c_1	1.496
c_2	1.496
c_3	0.748
α	1.00
r_{excl}	31.5
r_{conv}	0.00

5.3.1. Parameter sensitivity of the proposed exclusion schemes

As was noted before, the proposed mechanisms involve two additional parameters, i.e. c_3 and α. So, in this experiment, we investigate the effect of these parameters on the performance of mQSO.

A. Effect of C_3 on the performance of Algorithm 1 and Algorithm 2

As stated in Section 4, the value of c_3, controls the tendency of each population to avoid moving toward the position where either it was excluded lately (Algorithm 1) or currently captured by other populations (Algorithm 2).

Table 3 and Table 4 illustrate the effect of different values for c_3 parameter on the performance of Algorithm 1 and Algorithm 2, respectively. As can be observed in Table 3 and Table 4, c_3 has an important effect on the performance of Algorithm 1 and Algorithm 2. See for example that the worst result is obtained when $c_3 = 1.496$. The reason can be attributed to the fact that when $c_3 = 1.496$ the populations are severely repelled by the positions where either they were excluded lately (Algorithm 1) or currently captured by other populations (Algorithm 2). Similarly, it is also possible to note that best result is obtained with $c_3 = 0.748$.

Table 3. Average offline error obtained by Algorithm 1 when (*a*) $c_3 = 1.496$, (*b*) and $c_3 = 0.748$ on different instances of MPB with *m*=100, respectively.

	f=500	*f*=1000	*f*=2500	*f*=5000	*f*=10000	*f*=500	*f*=1000	*f*=2500	*f*=5000	*f*=10000
s=1	8.15	6.20	4.67	3.93	3.52	**7.93**	**6.01**	**4.47**	**3.79**	**3.37**
s=2	10.10	7.44	5.39	4.29	3.77	**9.74**	**7.20**	**5.22**	**4.28**	**3.64**
s=3	11.39	8.21	5.89	4.67	4.04	**10.94**	**7.94**	**5.71**	**4.58**	**3.93**
s=4	12.19	8.79	6.26	5.02	4.16	**11.79**	**8.51**	**6.12**	**4.88**	**3.99**
s=5	12.88	9.27	6.60	5.24	4.29	**12.36**	**8.95**	**6.44**	**5.13**	**4.23**

Table 4. Average offline error obtained by Algorithm 2 when (*a*) $c_3 = 1.496$ and (*b*) $c_3 = 0.748$ on different instances of MPB with *m*=100, respectively.

	f=500	*f*=1000	*f*=2500	*f*=5000	*f*=10000	*f*=500	*f*=1000	*f*=2500	*f*=5000	*f*=10000
s=1	8.56	6.13	4.18	3.29	2.91	**8.24**	**5.74**	**3.91**	**3.15**	**2.76**
s=2	10.14	7.02	4.71	3.79	3.30	**9.5**	**6.51**	**4.48**	**3.6**	**3.05**
s=3	11.15	7.68	5.21	4.21	3.67	**10.46**	**7.10**	**5.01**	**4.04**	**3.41**
s=4	11.94	8.22	5.65	4.56	3.91	**11.16**	**7.71**	**5.43**	**4.41**	**3.69**
s=5	12.53	8.71	6.01	4.94	4.19	**11.85**	**8.23**	**5.87**	**4.76**	**4.01**

B. Effect of α on the performance of Algorithm 3

The value of α parameter in Eq. 12, determines the probability distribution of triggering exclusion around the best individual of each population.

To see if parameter α can also affect the performance of the Algorithm 3, in this experiment we examine the effect of α on the performance of Algorithm 3 in DOPs with various change frequencies and m=100. Figure 7 draws the average offline error of Algorithm 3 with varying α on different dynamic environments.

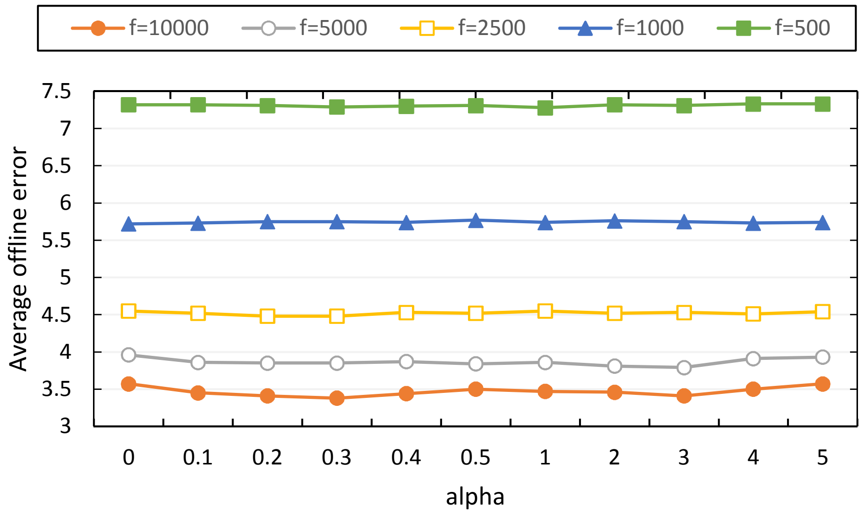


Figure 7. Performance of Algorithm 3 with different values of α on different instances of MPB with m=100

From Figure 7, it can be seen that the parameter α can slightly affect the performance of the mPSO. The effect of parameter α is more observable when the frequency of changes is low, i.e. f=5000 and f=10000. For example, on the DOP with f=10000, when $\alpha = 0$ the average offline error is 3.57. This value is decreased to 3.38 and 3.50 when the parameter α is set to 0.3 and 1, respectively. A similar observation can be made for the DOP with f=5000. In the rest of the experiments, for the sake of simplicity and to remove an extra parameter from our method, $\alpha = 1$ is applied.

5.3.2. Effect of varying the change frequency and shift severity

In this experiment, the effect of the change frequency and shift severity on the performance of mQSO with different exclusion variants is examined. The various combinations of (f, s) resulted

in 15 different problem instances. The other environmental parameters were set according to Table 1. The numerical results of different methods are reported in Table 5 and Table 6.
From these comparisons, it is possible to observe that in general, Algorithm 1 is the worst of eight in terms of optimality, even compared with mQSO. The reason can be contributed to the fact that the exclusion may happen between two populations before one of them is converged to an optimum. Therefore, the exclusion scheme used in Algorithm 1 misdirects the populations and prevent them from moving toward promising sub-areas of the search space.

Table 5. Average offline error ± standard error of different algorithms in DOPs with various combinations of shift severity and change frequency.

s	*f*	Method							
		mQSO	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	B/W	RMC	HVD3P
1	500	7.32±0.02	7.98±0.03	8.18±0.03	**7.28±0.02**	**7.28±0.02**	7.32±0.02	7.32±0.02	7.31±0.02
	1000	5.72±0.03	6.00±0.03	**5.70±0.02***	5.74±0.03	5.74±0.03	5.72±0.03	5.79±0.03	5.71±0.03
	2500	4.55±0.05	4.47±0.04	**3.93±0.03**	4.55±0.04	4.51±0.05	4.50±0.04	4.47±0.04	4.56±0.05
	5000	3.96±0.06	3.82±0.05	**3.13±0.04**	3.86±0.06	3.87±0.06	3.88±0.06	3.80±0.05	3.85±0.05
	10000	3.57±0.08	3.33±0.06	**2.64±0.04**	3.47±0.08	3.49±0.08	3.56±0.09	3.40±0.08	3.42±0.09
3	500	10.20±0.03	10.92±0.03	10.46±0.03	**10.19±0.03***	10.20±0.03	10.23±0.03	10.26±0.03	10.26±0.03
	1000	7.77±0.04	7.88±0.03	**7.13±0.02**	7.78±0.04	7.79±0.04	7.75±0.04	7.85±0.04	7.78±0.03
	2500	5.72±0.05	5.69±0.04	**5.01±0.03**	5.63±0.04	5.67±0.04	5.68±0.05	5.72±0.05	5.63±0.05
	5000	4.49±0.06	4.63±0.05	**4.01±0.03**	4.50±0.06	4.43±0.06	4.53±0.06	4.50±0.06	4.41±0.06
	10000	3.78±0.09	3.83±0.06	**3.42±0.04**	3.73±0.08	3.69±0.09	3.82±0.09	3.67±0.07	3.65±0.07
5	500	11.56±0.03	12.31±0.03	11.80±0.03	11.63±0.03	11.59±0.02	**11.54±0.03***	11.58±0.02	11.57±0.02
	1000	8.79±0.03	8.94±0.03	**8.17±0.02**	8.77±0.03	8.78±0.03	8.73±0.03	8.80±0.03	8.75±0.03
	2500	6.25±0.05	6.37±0.04	**5.79±0.03**	6.14±0.04	6.20±0.04	6.15±0.04	6.27±0.05	6.19±0.04
	5000	**4.66±0.05**	5.14±0.05	4.73±0.03	**4.66±0.05***	4.71±0.05	4.73±0.05	4.80±0.06	4.70±0.05
	10000	3.75±0.07	4.16±0.06	3.98±0.04	3.74±0.06	3.73±0.07	3.73±0.07	3.79±0.07	**3.72±0.07***

Best values are shown using *boldface*.

Regarding Table 5, it is clear that the Algorithm 2 is considerably superior to mQSO. Besides, it is possible to observe that Algorithm 2 shows an overall superiority over the other methods. It is the best-performing method in 10 out of 15 problem instances. Only in cases where the environment is fast changing (*f*=500), algorithm 2 fails to reach an acceptable error. However, a different pattern arises when the frequency of changes is decreased (500<*f*). Moreover, it is easy to observe that the algorithm 2 can produce the best results when shift severity *s*=1 and *s*=3.

On problems with frequent changes, i.e. *f*=500, Algorithm 3 and Algorithm 4 are the best-performing algorithms. It is also important to note that RMC is not among the best-performing algorithms in any of the tested DOPs.

Table 6. Average best-error-before-change ± standard error of different algorithms in DOPs with various combinations of shift severity and change frequency.

s	*f*	Method							
		mQSO	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	B/W	RMC	HVD3P
1	500	4.86±0.02	5.22±0.02	5.33±0.02	**4.84±0.02***	**4.84±0.02***	4.87±0.02	4.87±0.02	4.86±0.02
	1000	4.12±0.03	4.33±0.03	**4.08±0.02**	4.13±0.03	4.13±0.02	4.13±0.03	4.18±0.03	4.12±0.03
	2500	3.51±0.04	3.56±0.04	**3.14±0.03**	3.52±0.04	3.47±0.04	3.47±0.04	3.44±0.04	3.51±0.04
	5000	3.20±0.05	3.17±0.05	**2.57±0.03**	3.13±0.05	3.14±0.06	3.15±0.05	3.06±0.05	3.11±0.05
	10000	3.10±0.08	2.88±0.06	**2.17±0.03**	3.02±0.08	3.05±0.09	3.11±0.09	2.95±0.08	2.98±0.09
3	500	**6.57±0.02**	7.11±0.02	6.78±0.02	**6.57±0.02***	6.58±0.02	6.60±0.02	6.63±0.02	6.63±0.02
	1000	5.15±0.03	5.43±0.02	**4.90±0.02**	5.17±0.03	5.17±0.03	5.15±0.03	5.22±0.03	5.18±0.02
	2500	3.83±0.04	4.19±0.03	3.79±0.02	3.76±0.03	3.79±0.03	3.79±0.04	3.81±0.04	**3.75±0.04**
	5000	3.09±0.05	3.54±0.04	3.17±0.03	3.11±0.06	3.05±0.05	3.13±0.05	3.08±0.05	**3.03±0.05**
	10000	2.93±0.09	3.12±0.06	**2.81±0.04**	2.88±0.07	2.84±0.08	2.98±0.09	2.82±0.07	**2.81±0.07**
5	500	7.35±0.02	7.98±0.02	7.64±0.02	7.40±0.02	7.35±0.02	**7.33±0.02**	7.39±0.02	7.37±0.02
	1000	5.66±0.02	6.07±0.02	**5.53±0.02**	5.64±0.02	5.64±0.02	5.62±0.02	5.68±0.02	5.64±0.02
	2500	3.90±0.04	4.59±0.03	4.29±0.02	**3.82±0.03**	3.85±0.03	**3.82±0.03**	3.93±0.04	3.86±0.03
	5000	2.87±0.05	3.79±0.05	3.68±0.03	**2.86±0.04***	2.88±0.04	2.91±0.04	3.00±0.05	2.92±0.04
	10000	2.59±0.06	3.23±0.06	3.22±0.04	2.58±0.06	2.57±0.06	**2.56±0.06***	2.65±0.06	2.57±0.06

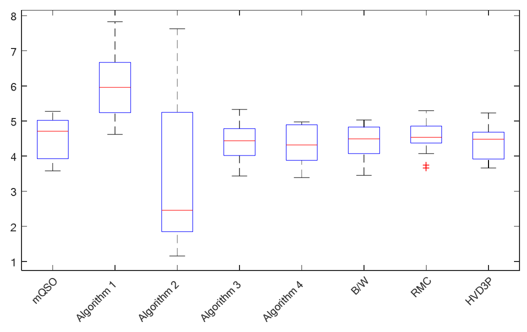
Best values are shown using *boldface*.

Different conclusions can be drawn from the comparison of Table 6. The reported results are more or less consistent with those of Table 5. Algorithm 2 is the best-performing method among the others in terms of best-error-before-change. Although Algorithm 2 is still the superior method when *s*=1, for DOPs with *s*=3 and *s*=5, HVD3P and B/W are performing better.

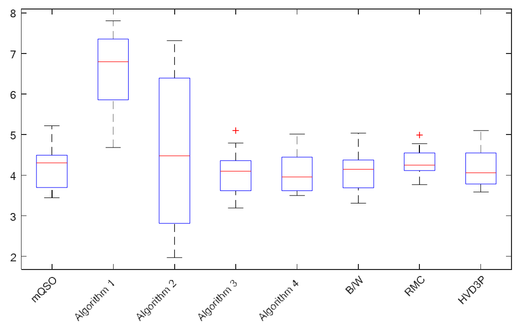
The results reported in Table 5 and Table 6 confirmed that using a suitable exclusion operator gives benefits to multi-swarm PSO when applying to DOP.

In order to have a better judgement about the optimality of the tested algorithms, we have performed non-parametric statistical tests using all reported mean values in Table 3 and Table 4. The analysis is divided in in two groups based on (*a*) offline error, and (*b*) best-error-before-change. The results are depicted in Figure 8 using boxplot. It is important to note that the lower the rank the better is the algorithm performance. Each boxplot in the figure corresponds to an exclusion variation with lines at the lower quartile, median, and upper quartile data values, where the whiskers are the lines extending from each end of the box to show the extent of the rest of the data.

Based on the Figure 8, it is observed that Algorithm 2 outperforms the other counterparts in terms of offline error. Moreover, although the difference of the mean ranks is lower than the other graphic, the algorithm 4 is ranked in better position than the others in terms of best-error-before-changes.



(*a*)



(*b*)

Figure 8. Ranking of the methods using Friedman test in the MPB's instances. The graphics summarized the rankings according the performance measure: (a) offline error and (b) best error before change

Despite the results reported in Tables 5 and Table 6, it is also important to analyze the algorithm behavior over time. In Figure 9 we plotted the evolution of the best-error-before-changes in terms of problem changes.

From Figure 9, the first thing that stands out is the superior performance of Algorithm 2. From the corresponding curve, it is possible to notice that the main difference between algorithms occurs during change 5 to change 100.

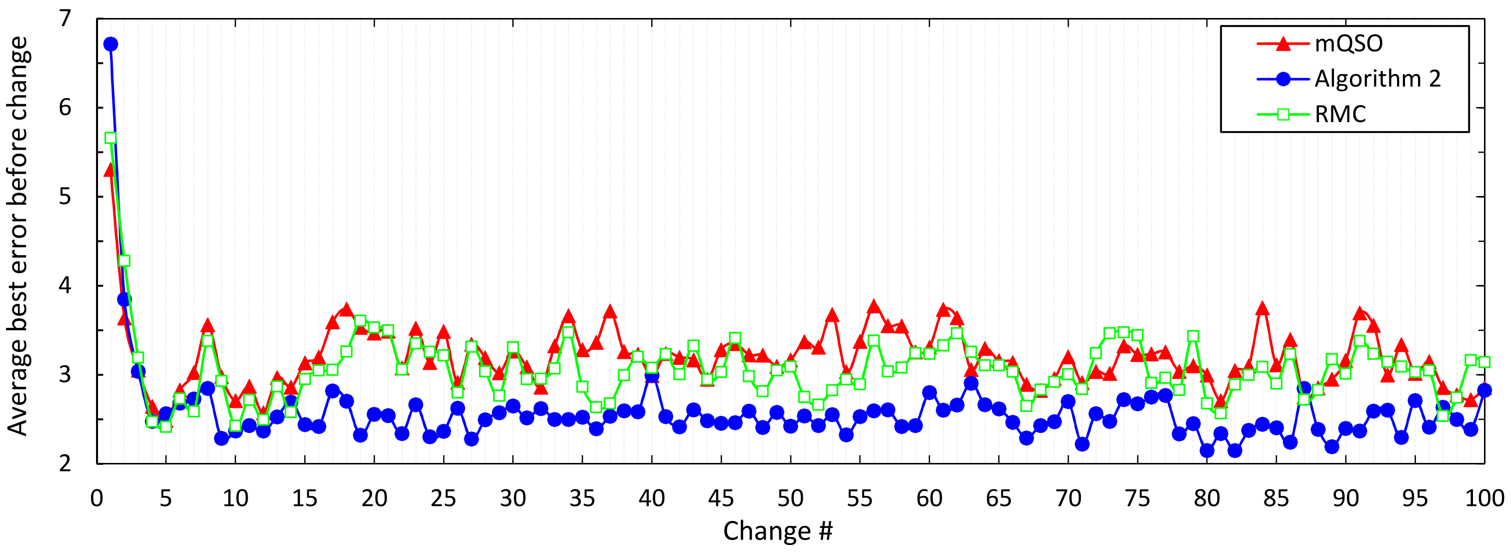


Figure 9. Evolution of the best error before change for the different algorithms in a problem with *m*=100, *f*=5000, *s*=1, *d*=5, and *peak function*=cone. The vertical dotted lines indicate the beginning of a new environment.

5.3.3. Effects of varying the number of peaks

In this experiment, we studied how different methods are affected by DOPs with different number of peaks and peak shape. In this case we considered the following settings $m \in \{1, 5, 10, 20, 30, 40, 50, 200\}$. In order to further investigate the effect of the different exclusion scheme on the performance of the mQSO, we also examined different landscapes with *cone* and *sphere* peak function. The other environmental parameters were set according to the default MPB settings shown in Table 1. The numerical results of different algorithms are reported in Tables 7 and 8.

Table 7. Average offline error ± standard error of different algorithms in DOPs with various number of peaks, *peak function* = *cone*, and shift severity *s*=1, change frequency *f*=5000.

m	Method							
	mQSO	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	B/W	RMC	HVD3P
1	4.44±0.13	5.11±0.16	5.61±0.17	3.73±0.11	4.62±0.14	4.58±0.15	**1.59±0.05**	2.51±0.08
5	1.75±0.06	2.35±0.07	2.69±0.07	1.80±0.06	1.74±0.05	**1.71±0.06**	1.79±0.08	1.76±0.07
10	1.71±0.06	2.34±0.06	2.72±0.06	1.81±0.05	1.72±0.05	**1.69±0.04***	2.09±0.07	2.05±0.08
20	2.81±0.06	3.19±0.06	3.09±0.06	2.82±0.06	**2.77±0.05**	**2.77±0.06**	2.94±0.06	2.90±0.06
30	3.35±0.07	3.49±0.06	3.32±0.05	3.27±0.06	**3.25±0.06**	3.26±0.05	3.34±0.06	3.27±0.06
40	3.49±0.05	3.73±0.06	**3.35±0.05**	3.53±0.06	3.50±0.05	3.56±0.06	3.59±0.06	3.51±0.06

50	3.55±0.06	3.71±0.06	**3.29±0.04**	3.57±0.06	3.66±0.06	3.58±0.06	3.61±0.06	3.55±0.06
200	3.92±0.05	3.62±0.04	**2.92±0.03**	3.87±0.05	3.91±0.05	3.90±0.05	3.88±0.05	3.87±0.05

Best values are shown using *boldface*.

As can be seen in Table 7, for unimodal DOPs, Algorithm 2 fail to reach acceptable performance. One plausible explanation of such poor performance can be the ongoing focus on preventing populations from moving toward the same position. Therefore, if one population stands on a position near the optimum of the problem, the other populations are unable to contribute to the search process hence wasting the FEs. However, it is observed that the Algorithm 2 is more effective for problems with larger number of peaks. On problems with small and medium number of peaks, i.e. 5, 10, 20, 30, the algorithm B/W is an effective strategy. It is also observed that the performance of RMC on unimodal DOP is significantly better than others.

Table 8. Average offline error ± standard error of different algorithms in DOPs with various number of peaks, *peak function* = *sphere*, and shift severity *s*=1, change frequency *f*=5000.

m	Method							
	mQSO	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	B/W	RMC	HVD3P
1	21.33±0.22	26.12±0.33	26.08±0.37	14.41±0.19	22.95±0.25	21.19±0.25	**3.07±0.06**	7.71±0.13
5	3.29±0.09	4.16±0.11	4.72±0.08	2.66±0.06	3.74±0.10	3.40±0.08	**0.88±0.02**	1.19±0.02
10	0.97±0.03	1.29±0.04	2.02±0.05	0.88±0.02	0.99±0.04	0.99±0.04	**0.79±0.02**	0.81±0.02
20	2.34±0.06	2.56±0.05	2.98±0.05	**2.29±0.06**	2.36±0.06	2.32±0.06	**2.29±0.06**	2.30±0.05
30	2.90±0.06	3.92±0.07	3.51±0.06	2.92±0.07	**2.86±0.06***	2.91±0.05	2.90±0.06	2.90±0.06
40	3.19±0.06	3.32±0.06	3.64±0.06	3.19±0.06	3.22±0.07	3.21±0.06	**2.90±0.06**	3.30±0.07
50	3.49±0.07	3.47±0.06	3.87±0.06	3.46±0.08	**3.34±0.06**	3.54±0.07	3.43±0.08	3.53±0.07
200	3.89±0.06	**3.83±0.07**	3.96±0.06	3.84±0.05	3.92±0.07	3.90±0.06	3.97±0.07	3.95±0.07

Best values are shown using *boldface*.

Besides, the reported results in Table 7 and Table 8 can bring us to the conclusion that native exclusion in mQSO is not a good strategy for separating the search territory of the populations. Results also revealed that RMC is the most effective method on spherical peaks in terms of best-error-before-change.

5.3.4. Effect of the combining different schemes

Since the exclusion schemes can be categorized into mutually independent groups, they can thus be combined to form various algorithms. Regarding the exclusion, the optimization process can be divided into four phases:

1. moving populations according to a certain strategy (mQSO, Algorithm 1 and Algorithm 2);
2. deciding on whether or not the exclusion has happened between populations (Native exclusion, Algorithm 3, RMC, HVD3P);
3. saving the information obtained by the weaker population before reacting to the exclusion (No information sharing and B/W), and
4. handling the weaker population (Restart and Algorithm 4).

With these in mind we can construct new algorithms by choosing different strategies for each phase. Thus, new algorithms are specified using the general convention P1→P2→P3→P4 where "P" stands for "Phase" and → means followed by. For example, mQSO→native exclusion→no information sharing→restart describes the principles of canonical mQSO. Similarly, Algorithm 2→ RMC→B/W→Algorithm 4 describes a new algorithm in which:

1. The particles are moving by the update rule of Algorithm 2;
2. The collision between two populations is detected using RMC;
3. The best individual of the weaker population is transferred to the worst individual of the stronger population, and
4. The weaker population is reinitialized into the search space using Algorithm 4.

Although numerous combinations are possible $3 \times 4 \times 2 \times 2 = 48$, here we highlight some major results on a typical DOP with the default parameters as described in Table 1. In Table 9, each row with highlighted background represents the results obtained by combining the methods shown in its two previous rows.

Table 9. Average offline error ± standard error obtained by different algorithms

Phase 1	Phase 2	Phase 3	Phase 4	Results
mQSO	Native exclusion	B/W	Restart	3.88±0.06
Algorithm 1	Native exclusion	No information sharing	Restart	3.82±0.05
Algorithm 1	Native exclusion	B/W	Restart	3.75±0.05
mQSO	Native exclusion	No information sharing	Algorithm 4	3.87±0.06
Algorithm 1	Native exclusion	No information sharing	Restart	3.82±0.05
Algorithm 1	Native exclusion	No information sharing	Algorithm 4	3.76±0.05
mQSO	HVD3P	No information sharing	Restart	3.85±0.05

Algorithm 1	Native exclusion	No information sharing	Restart	3.82±0.05
Algorithm 1	HVD3P	No information sharing	Restart	3.79±0.05

The reported results in Table 9 show that combining different exclusion variants can also give a better final performance.

Several conclusions can be drawn from the results of the above experiments. Here we highlight a number of important conclusions as follows:

1. It is clearly observed that the exclusion operator has a significant influence on the performance of the multi-swarm PSO when solving DOP.
2. Among the proposed algorithms, Algorithm 3 and Algorithm 4 are the best choice for fast-changing environments, i.e. f=500, (See Table 5 and Table 6).
3. RMC is the best-performing strategy for 1-peak environments or when the DOP contains spherical peaks.
4. The best possible approach is to decrease the tendency of the swarms to converge to the captured sub-areas of the fitness landscape (Algorithm 2).
5. It is also possible to combine different exclusion strategies into a single method which can also yield better results in some cases.

6. Conclusions

The exclusion is the most evident operator for separating the search territory of populations in multi-population evolutionary dynamic optimization techniques. In this paper, different exclusion schemes suggested to improve the performance of the multi-population methods for dynamic optimization problems (DOPs) were classified and reviewed. Furthermore, four methods were proposed to alleviate the drawbacks of native exclusion. The performance of the proposed methods along with some of the state-of-the-art exclusion variants was empirically studied on moving peaks benchmark, which is the most widely used dynamic test function for continues space domain. The experimental results revealed the importance of exclusion and the applicability of our extensions. More specifically, the results suggested that decreasing the tendency of populations to move toward the captured sub-areas of the fitness landscape can be beneficial to the performance of multi-swarm optimization for DOPs.

Several research directions can be pursued as future works. First, the values of the parameters of proposed methods were adjusted based on some preliminary experiments and of course they are not the optimal. Specifically, c_3 is the most important parameter of Algorithm 2 that controls the impact of the repulsive term and should be further investigated in detail. Therefore, a comprehensive sensitivity analysis on the effect of parameters could be a direction for future research. Second, it is interesting to study exclusion schemes in other multi-population methods. Third, it is also very valuable to study the effect of proposed approaches on real-world problems. Finally, evaluating the performance of the proposed methods on more challenging benchmark functions like GDBG would be another subject for future research.

References

[1] C. Li, T. T. Nguyen, M. Yang, S. Yang, and S. Zeng, "Multi-population methods in unconstrained continuous dynamic environments: The challenges," *Inf. Sci.*, vol. 296, pp. 95–118, Mar. 2015.

[2] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computing*, 2004, pp. 489–500.

[3] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, Aug. 2006.

[4] I. G. del Amo, D. A. Pelta, and J. R. González, "Using heuristic rules to enhance a multiswarm PSO for dynamic environments," presented at the Evolutionary Computation (CEC), 2010 IEEE Congress on, 2010, pp. 1–8.

[5] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A New Particle Swarm Optimization Algorithm for Dynamic Environments," in *Swarm, Evolutionary, and Memetic Computing: First International Conference on Swarm, Evolutionary, and Memetic Computing, SEMCCO 2010, Chennai, India, December 16-18, 2010. Proceedings*, B. K. Panigrahi, S. Das, P. N. Suganthan, and S. S. Dash, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 129–138.

[6] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A hibernating multi-swarm optimization algorithm for dynamic environments," 2010, pp. 363–369.

[7] S. Yang and C. Li, "A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 959–974, Dec. 2010.

[8] C. Li and S. Yang, "A General Framework of Multipopulation Methods With Clustering in Undetectable Dynamic Environments," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 556–577, Aug. 2012.

[9] Rui Mendes and A. S. Mohais, "DynDE: a Differential Evolution for Dynamic Optimization Problems," 2005, vol. 3, pp. 2808–2815.

[10] D. Yazdani, M. R. Akbarzadeh-Totonchi, B. Nasiri, and M. R. Meybodi, "A new artificial fish swarm algorithm for dynamic optimization problems," 2012, pp. 1–8.

[11] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, “A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization,” *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2144–2158, Apr. 2013.
[12] N. Fouladgar and S. Lotfi, “A novel approach for optimization in dynamic environments based on modified cuckoo search algorithm,” *Soft Comput.*, vol. 20, no. 7, pp. 2889–2903, 2016.
[13] J. K. Kordestani, H. A. Firouzjaee, and M. Reza Meybodi, “An adaptive bi-flight cuckoo search with variable nests for continuous dynamic optimization problems,” *Appl. Intell.*, vol. 48, no. 1, pp. 97–117, Jan. 2018.
[14] R. I. Lung and D. Dumitrescu, “A new collaborative evolutionary-swarm optimization technique,” in *Proceedings of the 9th annual conference companion on genetic and evolutionary computation*, 2007, pp. 2817–2820.
[15] R. I. Lung and D. Dumitrescu, “Evolutionary swarm cooperative optimization in dynamic environments,” *Nat. Comput.*, vol. 9, no. 1, pp. 83–94, 2010.
[16] J. K. Kordestani, A. Rezvanian, and M. R. Meybodi, “CDEPSO: a bi-population hybrid approach for dynamic optimization problems,” *Appl. Intell.*, vol. 40, no. 4, pp. 682–694, 2014.
[17] C. Li and S. Yang, “Fast Multi-Swarm Optimization for Dynamic Optimization Problems,” 2008, pp. 624–628.
[18] T. Blackwell, “Particle Swarm Optimization in Dynamic Environments,” in *Evolutionary Computation in Dynamic and Uncertain Environments*, S. Yang, Y.-S. Ong, and Y. Jin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 29–49.
[19] I. Rezazadeh, M. R. Meybodi, and A. Naebi, “Adaptive Particle Swarm Optimization Algorithm for Dynamic Environments,” in *Advances in Swarm Intelligence*, 2011, pp. 120–129.
[20] M. C. du Plessis and A. P. Engelbrecht, “Using Competitive Population Evaluation in a differential evolution algorithm for dynamic environments,” *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 7–20, Apr. 2012.
[21] R. K. Ursem, “Multinational GAs: Multimodal optimization techniques in dynamic environments,” presented at the Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation, 2000, pp. 19–26.
[22] X. Zuo and L. Xiao, “A DE and PSO based hybrid algorithm for dynamic optimization problems,” *Soft Comput.*, vol. 18, no. 7, pp. 1405–1424, 2014.
[23] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, “A novel population initialization method for accelerating evolutionary algorithms,” *Comput. Math. Appl.*, vol. 53, no. 10, pp. 1605–1614, 2007.
[24] M. C. du Plessis and A. P. Engelbrecht, “Differential evolution for dynamic environments with unknown numbers of optima,” *J. Glob. Optim.*, vol. 55, no. 1, pp. 73–99, 2013.
[25] L. Xiao and X. Zuo, “Multi-DEPSO: A DE and PSO based hybrid algorithm in dynamic environments,” presented at the Evolutionary Computation (CEC), 2012 IEEE Congress on, 2012, pp. 1–7.
[26] J. Lepagnot, A. Nakib, H. Oulhadj, and P. Siarry, “A multiple local search algorithm for continuous dynamic optimization,” *J. Heuristics*, vol. 19, no. 1, pp. 35–76, 2013.
[27] Y. Shi, “Particle swarm optimization: developments, applications and resources,” presented at the evolutionary computation, 2001. Proceedings of the 2001 Congress on, 2001, vol. 1, pp. 81–86.

[28] T. Blackwell, “Particle Swarm Optimization in Dynamic Environments,” in *Evolutionary Computation in Dynamic and Uncertain Environments*, vol. 51, S. Yang, Y.-S. Ong, and Y. Jin, Eds. Springer Berlin Heidelberg, 2007, pp. 29–49.

[29] J. Branke, “Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems,” in *Proceedings of the Congress on Evolutionary Computation*, 1999, vol. 3, pp. 1875–1882.

[30] J. Branke, *Evolutionary Optimization in Dynamic Environments*, vol. 3. Springer Science & Business Media, 2002.

[31] J. Branke and H. Schmeck, “Designing evolutionary algorithms for dynamic optimization problems,” in *Advances in Evolutionary Computing: Theory and Applications*, Berlin, Heidelberg: Springer, 2003, pp. 239–262.

[32] K. Trojanowski and Z. Michalewicz, “Searching for optima in non-stationary environments,” in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, 1999, vol. 3, p. (2348).

[33] T. T. Nguyen, S. Yang, and J. Branke, “Evolutionary dynamic optimization: A survey of the state of the art,” *Swarm Evol. Comput.*, vol. 6, no. 0, pp. 1–24, 2012.