IEEE Computational Intelligence Society
Transactions on Evolutionary Computation

An Adaptive Multi-Population Framework for Locating and Tracking Multiple Optima

Journal:	*Transactions on Evolutionary Computation*
Manuscript ID:	TEVC-00201-2015
Manuscript Type:	Regular Papers
Date Submitted by the Author:	16-May-2015
Complete List of Authors:	Li, Changhe; China University of Geosciences(Wuhan), Hubei Key Laboratory of Intelligent Geo-Information Processing Nguyen, Trung Thanh; Liverpool John Moores University, School of Engineering Yang, Ming; China University of Geosciences, School of Computer Science Mavrovouniotis, Michalis; De Montfort University, School of Computer Science and Informatics Yang, Shengxiang; De Montfort University, School of Computer Science and Informatics;
Keywords:	Multi-population optimization, Dynamic optimization, Multi-modal optimization, Population adaptation

An Adaptive Multi-Population Framework for Locating and Tracking Multiple Optima

Changhe Li, Trung Thanh Nguyen, Ming Yang, Michalis Mavrovouniotis, Shengxiang Yang

***Abstract*—Multi-population methods are important tools to solve dynamic optimization problems. However, to effectively track multiple optima, algorithm designers need to address a key issue: adaptation of the number of populations. In this paper, an adaptive multi-population framework is proposed to address this issue. A database is designed to collect heuristic information of algorithm behavior changes. The number of populations is adaptively adjusted according to statistic information related to the current evolving status in the database as well as a heuristic value. Several other techniques are also introduced, including a heuristic clustering method, a probabilistic prediction scheme, a population hibernation rule, and a peak hiding method. The particle swarm optimization and differential evolution algorithms are implemented into the framework, respectively. A set of multi-population based algorithms are chosen to compare with the proposed algorithms on the moving peaks benchmark using four different performance measures. The proposed algorithms are also compared with two peer algorithms on a set of multi-modal problems in static environments. Experimental results show that the proposed algorithms outperform the other algorithms in most scenarios.**

***Index Terms*—Multi-population optimization, dynamic optimization, multi-modal optimization, population adaptation.**

I. Introduction

Generally speaking, multi-population methods (MPMs) use more than one population to cooperatively search in different local areas in the fitness landscape to locate multiple optima or the global optimum. They are widely used to track multiple optima/peaks in parallel for dynamic optimization problems (DOPs) in continuous space. The motivation is that each population covers a different peak, so that tracking the global optimum would be easy if the global optimum moves to the area where a population is covering. For a DOP with a certain number of peaks, tracking the global optimum would be inefficient if the number of populations is far less than the number of peaks, on the other hand, the tracking would also be inefficient if the number of populations is far more than the number of peaks. Therefore, to effectively solve DOPs by MPMs, one key issue is to adapt the number of populations [23], [30].

A good practice is to choose the number of populations in relation to the number of optima if known [21]. Many experiments have shown that an inappropriate number of populations would negatively affect the performance of MPMs [3], [6], [21], [23], [38]. This problem becomes more challenging for DOPs with a changing number of optima/peaks [23], [38]. In the literature of MPMs for DOPs, many studies focus on a fixed number of populations [6], [16], [25]. Although algorithms based on a dynamic number of populations were proposed [9], [20], [32], the total number of individuals is fixed. This limitation constrains the adaptability of such algorithms. For example, it would be out of the capability of such algorithms to track all optima in parallel when the number of optima is more than the total number of individuals. To the best of our knowledge, only three versions of adaptive MPMs [3], [23], [38] have been proposed for DOPs so far. The difficulties in developing such algorithms lie in that: a) the number of optima in the fitness landscape is unknown and b) the relationship between a good choice of the number of populations and the number of optima is also unknown even if a priori knowledge of the number of optima is available.

To address the aforementioned issues for MPMs, this paper proposes an adaptive multi-population (AMP) framework based on an adaptive mechanism. The adaptive mechanism learns from algorithm behavior changes by means of interacting with environments and, in turn, guides the changes of the algorithm behavior toward a promising direction. In the framework, a probabilistic scheme is used to determine whether to increase, decrease, or make no change to the total number of individuals. A peak hiding method is proposed to hide peaks that have been explored so that no more populations would move to those peaks any more.

The most important difference between this work and existing studies [3], [23], [38] is that the adaptive behavior of this work is obtained by a learning process based on historical data. In contract, the adaptation in studies [3], [23], [38] is obtained only based on the knowledge of current evolving status (see the detailed discussions in Sect. II and Sect. III-B).

The rest of this paper is organized as follows. Sect. II reviews related research in the literature of MPMs for DOPs. The components of the proposed AMP and two instantiated algorithms are introduced in detail in Sect. III. Experimental studies are provided in Sect. IV. Finally, Sect. V concludes the paper.

Manuscript received May 16, 2015. This work was supported in part by the National Natural Science Foundation of China under Grants 61203306 and 61305086, the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1, and a British Council UK-ASEAN Knowledge Partnership grant and a British Council Newton Institutional Links grant.

C. Li and M. Yang are with the Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China (email: changhe.lw@gmail.com, yangming0702@gmail.com).

T. T. Nguyen is with the School of Engineering, Technology and Maritime Operations, Liverpool John Moores University, Liverpool L3 3AF, U. K. (T.T.Nguyen@ljmu.ac.uk).

S. Yang and M. Mavrovouniotis are with the School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U. K. (syang@dmu.ac.uk, mmavrovouniotis@dmu.ac.uk).

II. Related Work

One early version of MPM is the self-organizing scouts (SOS) algorithm [9] proposed to solve the moving peaks benchmark (MPB). SOS starts from a parent population that explores new promising peaks. Child populations, which are used to track peaks, are generated by splitting off from the parent population when a certain condition (*forking generations* are detected) is satisfied. Based on the scout model of SOS, several other algorithms were proposed. A multi-swarm algorithm was developed in [4], where a part of swarms exploit peaks that have been detected and remaining swarms keep exploring new peaks. Another multi-swarm forking algorithm [44] was developed to solve the MPB by applying the same idea with SOS to PSO. A fast multi-swarm optimization algorithm was proposed in [19] using a similar idea with SOS to organize multiple swarms except that child swarms are created when changes are detected. To give more computing time to those productive swarms than those unproductive swarms, a hibernation multi-swarm optimization (HmSO) was proposed in [16]. A child swarm is forced to hibernate if its radius is less than a converging threshold value and the fitness of its best particle is less than the fitness of the global best particle by a predefined level.

Instead of the splitting idea, many algorithms use a re-grouping idea to create populations. A popular algorithm is the speciation-based PSO (SPSO) [32]. As SOS, SPSO also starts with a large size swarm. Differently, the initial swarm is divided into a number of sub-swarms by a speciation-based rule. A swarm is created by combining the best particle with particles that are close to that best particle (i.e., the distance to the best particle is less than a predefined value). Once a swarm is constructed, its particles are removed from the initial swarm. These procedures are repeated until the initial swarm is empty. The construction of swarms is performed every iteration. Based on this re-grouping idea, many improved versions of SPSO and variants were proposed.

A mechanism to remove duplicated particles was introduced to enhance the performance of SPSO [33]. Another improved version of SPSO (rSPSO) was developed by integrating a least square regression method [2]. Recently, a multi-population harmony search algorithm was proposed [40], where a harmony search method [13] is used for each population to locate peaks. The best individuals of converged populations are kept to replace redundant ones.

In addition to the speciation-based approaches, niching techniques are also widely used to maintain multiple populations. An adaptive niching PSO was proposed in [1] where niching radii can be calculated adaptively. A vector-based PSO (VBPSO) was developed in [35], in which the *dot* product of two vectors is used to identify niches. The algorithm shows a competitive performance for multi-modal optimization. Thereafter, VBPSO was extended in [36] for DOPs. Recently, a cluster-based DE algorithm for niching was proposed in [29], in which different kinds of strategies were introduced to efficiently track multiple peaks.

A composite PSO was proposed in [24], where swarms are constructed in a similar way with SPSO except that the construction starts from the worst particle and all swarms contain a fixed number of three particles. Particles are randomly re-grouped into several swarms after a certain number of iterations in [15]. A multi-nation genetic algorithm was proposed in [42], where a hill-valley detection method was introduced to create sub-populations (note that, this algorithm was proposed before SPSO [32] but it has not been widely used due to the limitation of the hill-valley detection method).

Instead of creating populations during the runtime, many MPMs start with a fixed number of populations. One of the most popular algorithms is the atomic swarm model [5], where three kinds of particles with different roles are defined. They are charged particles, quantum particles, and neutral particles. In each swarm in the model, either changed particles (mCPSO [6]) or quantum particles (mQSO [6]) play the role of maintaining diversity and neutral particles are used to locate peaks. An exclusion principle ensures that only one swarm surrounds a single peak and an anti-convergence principle is also introduced to explore new promising peaks.

Motivated by the atomic model [5], several similar algorithms have been proposed. A multi-population dynamic differential evolution (DynDE) [28] algorithm was proposed, where four types of individuals, named DE, entropy DE, quantum and Brownian, are defined. An improved version of mQSO was proposed in [11], where two heuristic rules are applied to further enhance the diversity when changes occur. A fuzzy-C-means strategy was introduced to adapt the exclusion radius in [34]. In the algorithm, all particles are transformed to quantum particles till the next iteration when a change is detected. Recently, a cooperative quantum PSO [41] was proposed by using a cooperative framework introduced in [7]. A multi-swarm algorithm, called finderCtracker multi-swarm PSO (FTMPSO), was proposed in [48] by integrating several schemes, including a finder scheme, a tracker scheme, a change detection scheme, a wakening and sleeping scheme, and a local search scheme. Thereafter, the authors proposed a multi-swarm algorithm based on a new artificial fish swarm algorithm (mNAFSA) [47]. In the algorithm, a mechanism of finding and covering potential optimum peaks was proposed.

Instead of using different types of individuals in each population, several MPMs use different search algorithms for different populations. A collaborative evolutionary swarm optimization (CESO) algorithm was proposed in [25], where two swarms, using the crowding DE [39] and PSO, respectively, cooperate with each other using a collaborative principle to track the global optimum. Thereafter, a new version of CESO was proposed in [26], called evolutionary swarm cooperative algorithm (ESCA), where another swarm using PSO was added and the cooperation principle was also updated. A dual-swarm PSO was proposed in [49], where the information of the two best particles of the two swarms is transmitted at the dimensional level with a certain probability. A multi-environmental cooperative model [17] was introduced to deal with DOPs that have different sub-problems or environments.

Another important kind of MPMs are clustering-based algorithms. A popular example is the clustering PSO (CPSO) [46], where a hierarchical clustering method is used to create multiple swarms by clustering a random swarm whenever

a change is detected. In CPSO, an overlapping detection principle was proposed to identify whether two swarms crowd around one single peak when they move into each other's search area. One of the two swarms is removed if they are not overlapped but overcrowded (i.e., they crowd around one single peak). Thereafter, CPSO was enhanced to a version, called CPSOR, which does not need to detect changes. CPSOR introduces a principle that the population diversity is automatically increased once the total number of individuals is less than a threshold value. Recently, a new cluster-based DE algorithm was proposed in [14], where *k-means* is used to create populations whenever a change is detected. The number of populations may vary after every certain time span depending on the performance of the algorithm.

All methods mentioned above do not address one important issue of MPMs for DOPs, which is how to adapt the number of populations to dynamic environments, especially in the situation where the number of peaks changes overtime. Several attempts have been made to address this difficult issue. A self-adaptive multi-swarm optimizer (SAMO) [3] was developed based on the mQSO [6]. SAMO starts with a single free swarm. The number of free swarms is decreased when some of them are converging. SAMO creates a new free swarm if there are no free swarms. Converging swarms are identified by simply checking their radius against a predefined value. This way, the number of populations will be adaptively adjusted. Accordingly, the search area of each swarm is also adjusted by a formula taking the number of populations into account. The motivation of SAMO was adopted in a DE algorithm, called DynPopDE [38]. A new free population is created when one population is identified as a stagnating population. A stagnating population will be removed if it is identified for re-initialization due to exclusion.

Note that, in this paper a population is considered "converging" if the average distance between individuals begins to decrease. A population is considered "converged" if the average distance between individuals is less than a threshold value. A population is considered "stagnating" if it stops improving permanently but does not show any trend of converging [18].

Recently, an adaptive multi-swarm optimizer (AMSO) [23] was proposed. AMSO maintains a number of populations obtained by the clustering method used in [46]. Due to the overlapping handling principle [46], the number of populations will decrease after each diversity increasing point. A certain number of individuals are introduced when the drop rate of the number of populations over a time span is less than a small value. The number of individuals to be adjusted depends on the difference of the number of populations between the current increasing point and the previous increasing point. This way, AMSO is able to adaptively adjust the number of populations during the runtime.

Although there are several adaptive MPMs for DOPs, the principles to adjust the number of populations simply rely on the current information available, e.g., no free populations in SAMO [3], appearance of stagnating populations in DynPopDE [38], and the difference of the number of populations at the current and previous increasing points in AMSO [23]. Simply relying on the current information to adjust the number of populations may lead to mistakes due to the complex nature of dynamic environments. In order to address the aforementioned issues of adapting the number of populations, we introduce the AMP framework in this paper, which is described below.

Algorithm 1 $AMP()$

1: $\mathbb{P} \leftarrow 0$; $\mathbb{D} \leftarrow 0$; $t \leftarrow 0$
2: Create an initial population C with $gSize$ individuals;
3: $\mathbb{P} \leftarrow Cluster(C)$;
4: **while** stopping criteria are not satisfied **do**
5: **for** each population $\mathbb{P}[i]$ **do** $\mathbb{P}[i].search()$; **end for**
6: Hibernate populations when they find new peaks;
7: Derate individuals if they fall into the attraction area of any peak;
8: Remove excluded populations;
9: **if** $r_{avg}^{conv} < \theta \cdot S$ **then**
10: Wake up hibernating populations;
11: Create a map and put it to database $\mathbb{D}$;
12: Estimate the number of individuals for the next phase;
13: Create a random population C';
14: $\mathbb{P} \leftarrow \mathbb{P} \cup Cluster(C')$;
15: $t \leftarrow t + 1$;
16: **end if**
17: **end while** ▷ $\mathbb{P}$ is a list to store all populations, $\mathbb{D}$ is a database to store maps from the number of populations to the total number of individuals, $gSize$ is the initial population size, S is the search range.

III. Adaptive Multi-population Framework

The work flow of the proposed AMP framework consists of three major components: clustering, tracking and adapting. The clustering component is used to create a number of non-overlapped populations, which have no overlapping search areas with each other without the need of manually setting any parameters. The tracking component allows locating and tracking local optima using any population-based search algorithm. Finally, the adapting component adjusts the total number of populations by predicting what will be the best number of populations.

Algorithm 1 presents the framework of the AMP. The adaptation mechanism is triggered whenever the average radius (r_{avg}^{conv}) of non-stagnating populations is less than a small value $\theta \cdot S$. Then, Algorithm 3 (introduced later in Sect.III-B) is called to estimate the number of individuals for the next phase. A random population (C') is created and clustered into a number of small populations, which are appended to a list ($\mathbb{P}$). The detailed description about each component is given in the following subsections.

A. Heuristic Clustering

A single linkage hierarchical clustering method [27] is used to create non-overlapping populations. Let:

- $d(i, j)$ be the Euclidean distance between two individuals i and j in the D-dimensional space.
- $D(C_1, C_2)$=$\min_{i \in C_1, j \in C_2} d(i, j)$ be the distance between two clusters C_1 and C_2.
- $R(C) = \frac{1}{|C|} \sum_{i \in C} d(i, i^*)$ be the radius of cluster C, where i^* is the average position of all individuals in C.
- d_{inter}=$\sum_{C_1, C_2 \in \mathbb{C}} D(C_1, C_2)$ be the sum of inter-cluster distances between each pair of clusters in a list $\mathbb{C}$.
- d_{intra}=$\sum_k \sum_{i,j \in C_k} d(i, j)$, $k = 1, 2, ..., |\mathbb{C}|$ be the sum of intra-cluster distances of all clusters in $\mathbb{C}$, where the

Algorithm 2 $Cluster(C')$

1: **for** $i < |C'|$ **do** $C_i \leftarrow C'[i]$; **end for** ▷ C' is a population
2: Calculate d_{inter} and d_{intra} of $\mathbb{C}$; ▷ $\mathbb{C}$=[$C_1, C_2, ...$]
3: **while** $d_{intra} < d_{inter}$ **do**
4: Merge C_i and C_j, where $D(C_i, C_j)$=min$_{i \neq j < |\mathbb{C}|} D(C_i, C_j)$;
5: Update d_{inter} and d_{intra};
6: **end while**
7: Return $\mathbb{C}$;

intra-cluster distance of cluster C_k is the sum of all the distances between each pair of individuals in C_k.

Algorithm 2 shows the work flow of the heuristic clustering method to cluster a population C'. It first creates a list $\mathbb{C}$ of clusters with each cluster containing only a single individual. Then, for each iteration, it merges a pair of clusters which have the smallest distance among all pairs of clusters in $\mathbb{C}$ and satisfies the condition $d_{intra} < d_{inter}$. When d_{intra} is equal to or greater than d_{inter}, the clustering procedure terminates. Then, all clusters in $\mathbb{C}$ are appended to a population list $\mathbb{P}$ (see Steps 3 and 14 in Algorithm 1), which is empty initially. The benefit of this heuristic clustering method, compared to the existing method in [21], [23], [46] is that it does not need users to manually set parameter such as the lower and upper bounds for cluster size. Note that, this method cannot guarantee that the sizes of all obtained clusters meet a required minimum population size for an algorithm (e.g., in this paper the minimum population size is five for the DE and two for PSO). In this paper, such clusters do not take part in the optimization process and their individuals are used as random individuals when a random population (C') is created (see Step 13 in Algorithm 1).

B. Adapting Populations

There are two main concerns regarding adapting the number of populations to changes: a) when to make an adjustment and b) how many populations to be adjusted.

For the first concern, many researchers consider the moment when a change occurs as the time to make an adjustment (e.g., increasing/introducing diversity or reusing information learnt from the past in many studies reviewed above [6], [8], [20], [25], [26], [33], [46]). However, this strategy has a limitation: it may not work for changes that are hard or impossible to detect by re-evaluating a set of points in complex environments, such as dynamic environments where a part of the fitness landscape changes [23], [30] or there is noise.

The adaptive MPM algorithms in [3], [23], [38] do not use the above strategy. However, these algorithms have other issues. In SAMO [3], the issue is that a stagnating swarm with a large radius due to having individuals searching on different peaks can be mistaken as a converging swarm if the total number of swarms is far less than the number of peaks. As a result, this stagnating swarm will not be eliminated and consequently no new swarm can be created. Althouth SAMO still works in this case because stagnating swarms normally will transform to converging swarms after a change occurs, but, this situation may affect the performance of SAMO. In DynPopDE [38], this stagnating issue is taken into account, where a population is taken as a stagnating population if its best individual does not improve over two successive iterations. However, this way of identifying stagnating populations is not very effective and it may cause too many populations to be generated. In AMSO [23] new populations are added if the number of populations over a time span is dropped beyond a threshold. However, this requires setting the correct values for the time span and drop threshold, which may not be easy [23].

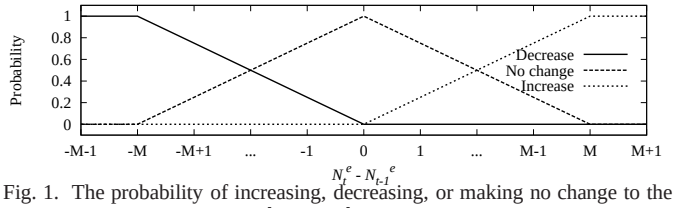

Fig. 1. The probability of increasing, decreasing, or making no change to the number of individuals, where $[-M, M]$ is the probabilistic range, N_t^e is the number of populations at the end of evolving phase t.

Based on the above considerations, we aim to find a simple yet effective way to identify the moment when populations are converging. To this aim, the average radius (r_{avg}^{conv}) of non-stagnating populations is monitored. Population adjustments are triggered when r_{avg}^{conv} is less than a threshold value $\theta \cdot S$ (S denotes the range of the solution space). Stagnating populations should be excluded because they can seriously affect the judgement of whether non-stagnating populations are converging if they have large search radii. In this paper, a population C is regarded as a stagnating population if all the three conditions below are met: a) its radius does not shrink after a certain number of successive iterations, which is equal to its size $|C|$ [45]; b) its radius is greater than the average radius of all populations in $\mathbb{P}$; c) its radius is greater than $\theta \cdot S$. Note that, this method does not guarantee that a population, which satisfies the three conditions, is a real stagnating population. However, a real stagnating population will satisfy the three conditions.

The second concern is very difficult to address directly due to the two difficulties mentioned in Sect. I, namely the lack of knowledge on the number of optima and on the correlation between the number of optima and the number of populations. However, we may still be able to indirectly address this concern by observing changes of algorithm behaviours. For example, it can be easily inferred that in an MPM algorithm, the number of survived populations (i.e., populations that find new, unexplored peaks – these populations will survive the exclusion procedure in MPMs) is proportional with the number of peaks in the fitness landscape. If the number of peaks increases, so does the number of survived populations, and vice versa. Therefore, we can use this heuristic relationship between the number of survived populations and the number of peaks to predict the number of populations to be adjusted.

Given this heuristic, we can address the second concern by following a two-step process. The first step is to decide whether to increase, decrease, or make no change to the total number of individuals. To make a choice from the three actions, a probabilistic prediction scheme, as illustrated in Fig. 1, is introduced, where N_t^e is the number of populations at the end of evolving phase t.

Algorithm 3 *Adjust( )*

```
ratio ← |N_t − N_{t−1}|/M;
if ratio >= 1 then f ← 1;
else if ratio = 0 then f ← 0;
else
    p ←rand();                          ▷ p is a random number within [0,1];
    if p < ratio then f ← 1;
    elsef ← 0;
    end if
end if
I^b_{t+1} ← N(D_μ[N^e_t], D_σ[N^e_t]) + 5f · (N^e_t − N^e_{t−1});
```

In this scheme, one of the three actions is taken depending on the value of N_t^e-N_{t-1}^e and a probability, which is related to $|N_t^e$-$N_{t-1}^e|/M$. The value of M >0 determines the probabilistic range (the scheme is deterministic if M is one). As illustrated in Fig. 1, the larger the difference between N_t^e and N_{t-1}^e, the larger the probability of increasing/decreasing the number of individuals. Algorithm 3 presents the pseudo-code of this probabilistic scheme. Note that, this scheme cannot guarantee that an action made by it is always right. However, it is expected that this scheme will improve the adaptation of algorithms in adjusting the number of individuals.

The second step is to estimate how many individuals should be set in the near future. Different from the existing methods [3], [23], [38], where the number of populations changes based on the current evolving status, in this paper the adaptation is achieved based on historical data and a heuristic adjustment. To achieve this objective, we first create a map from the current number of populations (N_t^e) to the total number of individuals at the beginning of the current phase (I_t^b) whenever a new adjustment is triggered. Note that, for the first map an over large initial population (far larger than needed, e.g, 500 individuals for the MPB with ten peaks) would affect the estimation if we use I_0^b (the initial population size) to create it. To address this issue, we use I_0^e for the first map instead of I_0^b. Then, the map is added into a database $\mathbb{D}$, which records all maps created since the start of the run. To estimate the number of individuals, we average the number of individuals of all maps which have the same number of populations as the current map. Then, the number of individuals for the next evolving phase is estimated by the formula below:

$$I_{t+1}^b \leftarrow N(\mathbb{D}_\mu[N_t^e], \mathbb{D}_\sigma[N_t^e]) + 5f \cdot (N_t^e - N_{t-1}^e), \qquad (1)$$

where $N(a, b)$ is a normally distributed random number with the mean a and the standard variation b, $D_\mu[N_t^e]$ and $D_\sigma[N_t^e]$ are the mean and standard deviation, respectively, of individuals with maps that have N_t^e populations, $f \in \{0, 1\}$ denotes if to take the corresponding action into account.

Finally, a random population C' is created with the size of I_{t+1}^b-I_t^e (see Step 13 in Algorithm 1). Note that, I_{t+1}^b will need to repair in the case of $I_{t+1}^b <= I_t^e$, where C' is created with ten individuals to guarantee that the diversity is increased when all non-stagnating populations are converging. In this situation, the size of C' should be small as many over-crowing populations would be created if a large value is used. Although the exclusion scheme (introduced in Sect.III-C later) is able to remove over-crowding populations, such populations still waste computing resources. Preliminary experimental results show that the value of ten is a reasonable choice.

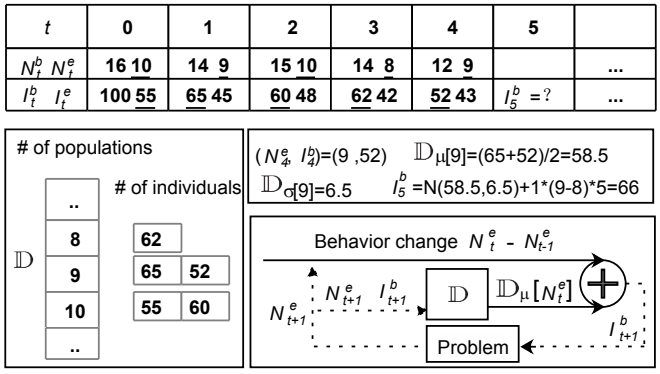

Fig. 2. The adaptation mechanism of the proposed AMP.

Fig. 2 shows an example of the estimation process from the database $\mathbb{D}$ with five maps. In Fig. 2, the number of populations at the current adjustment point is nine and the current map is (9,52). Then, the number of individuals for the next evolving phase (I_5^b) can be obtained by Eq. (1), which is 66. As shown in the bottom right graph in Fig. 2, the adaptation of the AMP is a feedback loop. The number of populations changes over time by interacting with the problem. For example, if increasing the number of population makes the AMP find new peaks, the AMP will keep doing so until no new peaks can be found. On the other hand, if the number of peaks decreases, the number of populations will be likely to decrease accordingly. The database $\mathbb{D}$ keeps receiving feedback of the algorithm behavior changes, in turn, it further guides the changes of the algorithm behavior. Therefore, the AMP is a self-regulating framework.

Note that, the constant value 5 in Eq. (1) is a step size. Its value should be small and roughly equal to the average population size. This way, the number of populations for the next evolving phase would be close to the expected value. A too large/small value will cause too many/few populations generated. Considering the average population size for all instances tested in this paper (4.687) and the minimum population size for the DE (5) and PSO (2), a value of 5 is chosen for this constant.

C. Population Exclusion

Over-crowding populations, which are multiple populations that surround the same peak, are not allowed in the AMP. In this paper, two populations are detected as over-crowding if both have at least one individual in each other's search areas. Then, the population with the worse best individual is removed from $\mathbb{P}$ (see Step 8 in Algorithm 1). Note that, this method cannot guarantee that two populations, which are detected as over-crowding, really search on a same peak.

D. Avoidance of Explored Peaks

Exploring peaks that have already been explored wastes computational resources and hence deteriorates an algorithm's performance. This is an important issue for EAs [31]. To address this issue, we propose a peak hiding technique in this

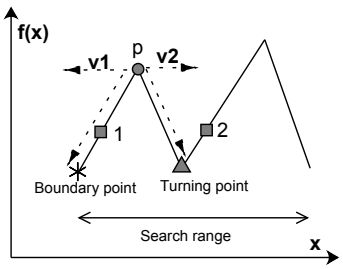

Fig. 3. Illustration of the peak hiding method.

paper. All peaks that have been explored are kept. A peak is assumed to be explored when there exists a population where the distance between its best individual and the peak is less than a value (ϵ_s) and the difference between the objective value of the best individual and the peak height is also less than a small value (ϵ_o) in the case that the peak location is known. Otherwise, a peak is regarded as an explored peak only if a population's radius shrinks to a small value of $1E$-9, and the location where the population converges is assumed to be the location of the peak.

To avoid populations re-searching peaks that have been explored, an idea is to remove the attraction of those explored peaks. The idea works as follows. For each explored peak, a set of vectors from the peak location to the boundary of its basion of attraction will be created as necessary. Initially, the set is empty. For an individual i, we find the closest explored peak p. In the vector set with peak p, if there does not exit a vector v' that makes its angle to $\vec{x}_i - \vec{x}_p$ less than three degrees, then a new vector v is created from p in the direction of $\vec{x}_i - \vec{x}_p$. The length of v gradually increases by a small step ($0.1\epsilon_s$) until a turning or a boundary point is found. Then v is added to the vector set of p. The fitness value of i is set to the fitness of the worst solution found so far if $|\vec{x}_i - \vec{x}_p|$ is less than $|v|$ or $|v'|$, depending on whether the condition above is met. Note that, it is possible to set the values of the angle threshold and the step to be smaller than the default values so that the vector v can be more precise but in such a case more evaluations will be needed. Our experimental studies show that the two default values are small enough for all the test problems in this paper.

Fig. 3 illustrates the procedures of finding such vectors ($v1$ and $v2$) for an explored peak p in a 1-D problem $f(x)$. In the figure, the triangle point is a turning point with peak p and the star point is a boundary point. The fitness of individual 2 does not change as $|x_2 - x_p|$ is greater than $|v2|$. However, the fitness of individual 1 will be set to the fitness of the worst individual found so far as $|x_1 - x_p|$ is less than $|v1|$. This way, all individuals fall in the basin of the attraction of peak p will not move forward to p any more since they will get the worst fitness by doing so, that is, peak p seems to have disappeared for such individuals.

E. Population Hibernation and Wakening

To save function evaluations, inspired by [16] a population will hibernate once it finds a peak and will wake up when the diversity adjustment is performed. Hibernating populations will not evolve until they wake up. Note that, waking up hibernating populations (Step 10 in Algorithm 1) is necessary when the AMP is running in dynamic environments. This is because when a change occurs, those hibernating populations are holding outdated memories and they must wake up in time to re-locate peaks that have moved. In the AMP, we choose the diversity adjustment point as the time point to wake up hibernating populations.

F. Movements for the Best Individual

For a non-stagnating population C, in order to quickly track a moving peak or a better peak within its search area but not covered by any population, the best individual $\vec{x}_{best}$ performs a Brownian movement [28] within the search area of C at each iteration as follows:

$$\vec{x'}_{best} = N(\vec{x}_{best}, R(C)) \tag{2}$$

and it will be replaced if a better solution $\vec{x'}_{best}$ is found.

However, if C is a stagnating population, it may not benefit from the Brownian movement. Jumping out its search area could help it transform to a converging population. To achieve this, we allow its best individual to perform a Cauchy movement at each iteration as follows:

$$\vec{x'}_{best} = Q(\vec{x}_{best}, S/2) \tag{3}$$

where $Q(a, b)$ is a random number of Cauchy distribution with location parameter a and scale parameter b.

G. Instantiation of the AMP Framework

In the framework, any population-based algorithm can be used to search for optima. In this paper, it will be instantiated with a PSO algorithm and a DE algorithm, respectively.

1) AMP with PSO : The PSO with an inertia weight [37] is used in this paper. The velocity and position of particle i are updated as below:

$$\vec{v'}_i = \omega\vec{v}_i + \eta_1\vec{r}_1(\vec{x}_{p_i} - \vec{x}_i) + \eta_2\vec{r}_2(\vec{x}_g - \vec{x}_i) \tag{4}$$

$$\vec{x'}_i = \vec{x}_i + \vec{v'}_i, \tag{5}$$

where $\vec{x'}_i$ and $\vec{x}_i$ represent the current and previous positions of particle i, respectively; $\vec{v'}_i$ and $\vec{v}_i$ are the current and previous velocities of particle i, respectively; $\vec{x}_{p_i}$ and $\vec{x}_g$ are the best positions found by particle i so far and found by the whole swarm so far, respectively; $\omega = 0.7298$ and $\eta_1 = \eta_2 = 1.496$ are constant parameters, whose values were suggested by [43]; $\vec{r}_1$ and $\vec{r}_2$ are vectors of random numbers uniformly generated within $[0.0, 1.0]$ for each dimension. Note that, the maximum velocity of each particle is set to the initial search radius of its swarm in the AMP. Algorithm 4 presents the framework of the PSO.

2) AMP with DE: The DE with $DE/best/2/bin$ mutation strategy is used in this paper. Algorithm 5 shows the procedures of the algorithm. The parameters F and CR are set to 0.5 and 0.6, respectively, which were suggested by [28].

Algorithm 4 PSO

1: **for** each particle i **do**
2: Update particle i according to Eqs. (4) and (5);
3: **if** $f(\vec{x}_i) < f(\vec{x}_{pbest_i})$ **then**
4: $\vec{x}_{pbest_i} := \vec{x}_i$;
5: **if** $f(\vec{x}_i) < f(\vec{x}_{gbest})$ **then** $\vec{x}_{gbest} := \vec{x}_i$; **end if**
6: **end if**
7: **end for**

Algorithm 5 DE

1: **for** each individual i **do**
2: Generate a donor vector $\vec{v}$ by: $\vec{v} := \vec{x}_{best} + F \cdot (\vec{x}_{r1} - \vec{x}_{r2}) + F \cdot (\vec{x}_{r3} - \vec{x}_{r4})$; ▷ F is mutation factor in [0,2] and $\vec{x}_{r1}$, $\vec{x}_{r2}$, $\vec{x}_{r3}$, and $\vec{x}_{r4}$ are randomly selected individuals (indices of i, $r1$, $r2$, $r3$, and $r4$ are distinct)
3: Generate a trial vector $\vec{u}$ as follows:
$u^d := \begin{cases} v^d, & \text{if } r <= CR \text{ or } d = I_{rand} \\ x^d, & \text{if } r > CR \text{ and } d! = I_{rand} \end{cases}$
where CR is a probability constant and I_{rand} is a random integer within [1,D].
4: **if** $f(\vec{u}) < f(\vec{x}_i)$ **then** $\vec{x}_i := \vec{u}$; **end if**
5: **end for**

IV. Experimental Studies

To investigate the performance of the AMP, two groups of experiments are carried out in dynamic and static environments, respectively. For the first group, ten peer MPMs are selected for DOPs. They are mQSO [6], SAMO [3], SPSO [33], AMSO [23], CPSO [46], CPSOR [21], FTMPSO [48], DynDE [28], DynPopDE [38], and mNAFSA [47]. The two proposed algorithms are named AMP/PSO and AMP/DE, respectively. Among these algorithms, AMP/PSO, AMP/DE, SAMO, DynPopDE, and AMSO are adaptive algorithms in terms of the number of populations used in the run time. Comparison in this group is conducted based on the MPB problem [8]. For the second group in static environments, two popular multi-modal optimization algorithms: CRDE [39] and DE/nrand/1 [12], which are baseline models in [22], are chosen and the comparison is performed on ten multi-modal problems.

A. Problem Description

1) The Moving Peaks Benchmark: The MPB problem [8] is constructed by a number of peaks, which change in the location, height, and width. For the D-dimensional landscape, the problem is defined as follows:

$$F(\vec{x}, t) = \max_{i=1,\ldots,P} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{D} (x_j(t) - X_{ij}(t))^2}, \quad (6)$$

where $W_i(t)$ and $H_i(t)$ are the height and width of peak i at time t, respectively, and $X_{ij}(t)$ is the j-th element of the location of peak i at time t. The P independently specified peaks are blended together by the max function. The position of each peak is shifted in a random direction by a vector $\vec{v_i}$ of a distance s (s is also called the shift length, which determines the severity of the problem dynamics), and the move of a single peak can be described as follows:

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)), \quad (7)$$

where the shift vector $\vec{v}_i(t)$ is a linear combination of a random vector $\vec{r}$ and the previous shift vector $\vec{v}_i(t-1)$ and is normalized to the shift length s. The correlated parameter λ is set to 0, which implies that the peak movements are uncorrelated. A change of a single peak can be described as follows:

$$H_i(t) = H_i(t-1) + height\_severity_i * \sigma \quad (8)$$

$$W_i(t) = W_i(t-1) + width\_severity_i * \sigma \quad (9)$$

$$\vec{X}_i(t) = \vec{X}_i(t)(t-1) + \vec{v}_i(t), \quad (10)$$

where σ is a normal distributed random number with mean 0 and variation 1.

In this paper, a new feature, the change in the number of peaks introduced in [23], is used to test an algorithm's performance in terms of the adaptation of the number of populations. If this feature is enabled, the number of peaks changes using one of the following formulas:

$$Var1 : P = P + sign \cdot 2, \quad (11a)$$

$$Var2 : P = P + sign \cdot r(1,5), \quad (11b)$$

$$Var3 : P = r(10,100), \quad (11c)$$

where $sign = 1$ if $P <= 10$, $sign = -1$ if $P >= 100$, and the initial value of $sign$ is one; $r(a,b)$ returns a random value in $[a,b]$.

2) Multi-modal Functions: In this paper, ten multi-modal functions are chosen. Table II gives a description of these functions, where the major properties are as follows.

- The Waves function (F1) is asymmetric and has ten peaks (one global optimum and nine local optima), which are irregular disposed. Some of the peaks are difficult to find as they lie on the border or on flat hills.
- The Vincent function (F2) has 6^D global optima and no local optima. The global optima have vastly different spacing between them. A part of the optima are very difficult to find as they take a very narrow space in the fitness landscape.
- The Six-hump Camel Back function (F3) has six optima, which are disposed in a smooth landscape and two of the optima are global optima.
- The Shubert function (F4) contains $D3^D$ global optima unevenly disposed. These global optima are divided into 3^D groups, with each group having D global optima being close to each other. F2 also contains many other local optima, which are between the global optima.

TABLE I
Default settings for the MPB, where the term "change frequency (u)" means that the environment changes every u fitness evaluations, S denotes the range of allele values, $cPeaks$ denotes the percentage of changing peaks, and I,H,W denote the initial height, height range, and width range, respectively, for all peaks.

Parameter	Value	Parameter	Value
number of peaks (P)	10	number of dimensions (D)	5
change frequency (u)	5000	correlation coefficient (λ)	0
height severity	[1,10]	number of peaks change	no
width severity	[0.1,1.0]	S	[0, 100]
peak shape	cone	H	[30, 70]
basic function	no	W	[1, 12]
shift length (s)	1.0	I	50.0

TABLE II
DESCRIPTION OF TEN MULTI-MODAL FUNCTIONS, WHERE D IS THE NUMBER OF DIMENSIONS

Name	D	Function	# of global/local Opt.
Waves	2	$F1(\vec{x}) = (0.3x_1)^3 + 3.5x_1x_2^3 - 4.7cos(3x_1 - x_2^2(2+x_1))sin(2.5\pi x_1)$ $-0.9 \leq x_1 \leq 1.2, -1.2 \leq x_2 \leq 1.2$	1/9
Vincent	D	$F2(\vec{x}) = (\sum_{i=1}^{D} sin(10logx_i))/D$ $0.25 \leq x_i \leq 10$	6^D/0
Six-hump Camel Back	2	$F3(\vec{x}) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2)$ $-1.9 \leq x_1 \leq 1.9, -1.1 \leq x_2 \leq 1.1$	2/4
Shubert	D	$F4(\vec{x}) = \Pi_{j=1}^{D}(\sum_{i=1}^{5} icos((i+1)x_j + i))$ $-10 \leq x_i \leq 10$	$D3^D$/many
Modified Shekel	D	$F5(\vec{x}) = \sum_{i=1}^{8}(\sum_{j=1}^{D}(x_j - a_{ij})^2 + c_j)^{-1}$ $0 \leq x_i \leq 11$	1/7
IBA	2	$F6(\vec{x}) = (x_1^2 + x_2^2)/(1 + x_1^2 + x_2^2) + k(14(x_1^2 + x_2^2) + (x_1^2 + x_2^2)^2\chi^2 - 2\sqrt{14}(x_1^3 - 3x_1x_2^2)\chi)/(14(1 + x_1^2 + x_2^2)^2)$ $-4 \leq x_i \leq 4$	3/1
Himmenblau	2	$F7(\vec{x}) = (x_1^2 + x_2 - 11)^2 + (x_2^2 + x_1 - 7)^2 + 0.1((x_1 - 3)^2 + (x_2 - 2)^2)$ $-6 \leq x_i \leq 6$	1/3
Five hills	2	$F8(\vec{x}) = sin(2.2\pi x_1 + 0.5\pi)(2 - \lvert x_2\rvert)/2(3 - \lvert x_1\rvert)/2 + sin(0.5\pi x_2 + 0.5\pi)(2 - \lvert x_2\rvert)/2(2 - \lvert x_1\rvert)/2$ $-2.5 \leq x_1 \leq 3, -2 \leq x_2 \leq 2$	1/4
Center peak	2	$F9(\vec{x}) = 3sin(0.5\pi x_1 + 0.5\pi)(2 - \sqrt{x_1^2 + x_2^2})/4$ $-2 \leq x_i \leq 2$	1/4
BraninRCOS	2	$F10(\vec{x}) = (x_2 - 5.1 * x_1^2/(4\pi^2) + 5 * x_1/\pi - 6)^2 + 10(1 - 1/(8\pi))cosx_1 + 10$ $-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15$	3/0

- The Modified Shekel function (F5) has eight optima, one of which is the global optimum. Most of the optima are separated from each other by wide flat valleys. The parameters a_{ij} and c_j are given by

$$a_{ij} = \begin{pmatrix} 4 & 4 & 6.3 & 4 & 4 \\ 1 & 1 & 8.5 & 1 & 1 \\ 6 & 6 & 9.1 & 6 & 6 \\ 3.5 & 7.5 & 4 & 9 & 4 \\ 5 & 5 & 3 & 3 & 9 \\ 9.1 & 8.2 & 2 & 3 & 9 \\ 1.5 & 9.3 & 7.4 & 3 & 9 \\ 7.8 & 2.2 & 5.3 & 9 & 3 \end{pmatrix},$$

$$c_j = \left(0.10.20.40.150.60.20.060.18\right).$$

  The location of the global optimum is given by vector row $a_{7j}, j \in [1, D]$, and the other seven local optima are determined by other vector rows of the matrix $||a||$.
- The IBA function (F6) has three global optima and one local optimum. In the function, $k = -0.95$ and $\chi = -1.26$ is used.
- The Himmenblau function (F7) has one global optimum and three local optima with different objective values.
- The Five hills function (F8) and the Center peak function (F9) have each one global optimum and four local optima. F8 has five very close hills with lines of valleys between them. The four local optima in F9 are on the edge of the intervals and the global optimum is in the middle.
- The BraninRCOS function (F10) has three global optimum, which are distributed within an irregular and asymmetric landscape. In addition, the function has no local optima.

B. Experimental Setup

1) Performance Evaluation: To evaluate an algorithm's performance in tracking the global optimum, the offline error (E_O) [10] and the best-before-change error (E_{BBC}) are used. The offline error used in this paper is the average of the best error found every two objective evaluations and the best-before-change error is the average of the best error achieved at the fitness evaluation just before a change occurs.

TABLE III
THRESHOLD VALUES OF ϵ_s AND ϵ_o FOR IDENTIFYING OPTIMA, WHERE GO IS THE OBJECTIVE VALUE OF THE GLOBAL OPTIMA (THE SETUP OF FUNCTIONS F2-F4, AND F7 IS REFERRED TO [22]).

	MPB	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
ϵ_s	-	0.15	0.2	0.1	0.5	0.2	0.08	0.5	0.2	0.2	1.0
ϵ_o	0.1	1E-3	1E-4	1E-4	1E-3	1E-3	1E-6	1E-4	1E-4	1E-5	1E-5
GO(D=2)	-	7.307	1	-1.0316	186.731	16.832	-8.016E-3	2	2.5	1.5	0.397887

In addition, to evaluate an algorithm's performance in tracking multi-optima, the ratio of of peaks that are traced (PR) and the success rate of tracking all peaks (SR) are used. A peak is assumed to be traced if the difference of objective value between any individual and the peak is less than ϵ_o and the Euclidian distance between the individual and the peak is less than ϵ_s (see Table III for the values of ϵ_o and ϵ_s for all problems tested in this paper). For the value of ϵ_s for the MPB, it is set to $min(min_{i \neq j \leq P} d(X_i(t), X_j(t))/2, 0.1)$ at time t.

A two-tailed t-test with 58 degrees of freedom at a 0.05 level of significance was conducted for each pair of algorithms on E_O and E_{BBC}. The t-test results are given with the letters "w", "l", or "t", which denote that the performance of an algorithm is significantly better than, significantly worse than, or statistically equivalent to its peer algorithms, respectively.

2) Algorithm Configurations: For the AMP, there are three parameters to be investigated (see Sect. IV-C later). They are the initial population size ($gSize$), the convergence threshold (θ), and the probabilistic range M, whose default values are set to 100, $0.005 \times S$, and 3, respectively. For parameters of all the peer algorithms, default values suggested in their proposals are used if the algorithms show their best results. For example, the equations of setting the exclusion radius for each population suggested by Blackwell [6], [3] work well in our test. Therefore, we also use the default setting regarding this parameter for mQSO [6], SAMO [3], SPSO [33], DynDE [28], mNAFSA [47], and DynPopDE [38] as their authors used. However, for mNAFSA [47] try_number=2 and N=10

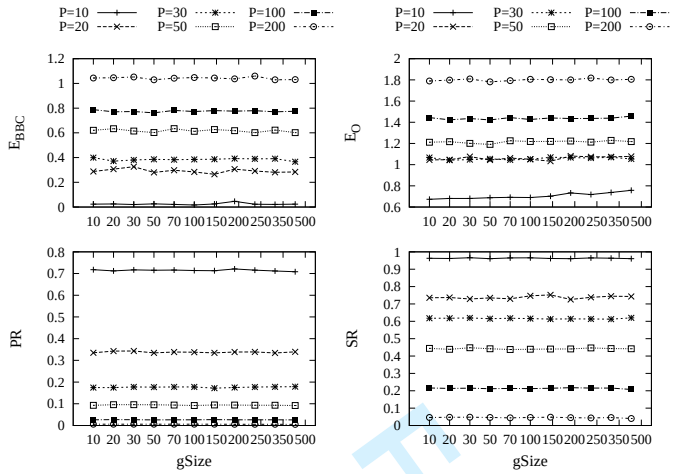

Fig. 4. The results of AMP/PSO with different initial sizes ($gSize$) on the MPB with different numbers of peaks.

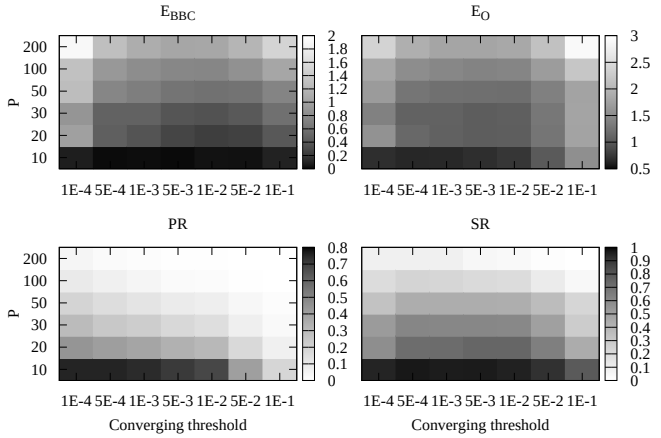

Fig. 5. The results of AMP/PSO with different converging thresholds on the MPB with different numbers of peaks.

are used instead of try_number=4 and N=2 suggested in the original paper [47]. The stopping criterion is 200 changes for the MPB problem and 2.0E+5 function evaluations or finding all peaks for multi-modal problems in static environments. All the results reported in the paper are averaged over 30 independent runs.

3) Open Framework of Evolutionary Computation: The open framework of evolutionary computation (OFEC) is a template library written in C++ based on the Boost library. It supports any population based EC methods running in parallel on CPU (a new feature of parallel running on GPU will be available in the next release). OFEC has so far collected a number of algorithms from PSO, GA, DE, and several other domains for global optimization problems, multi-modal optimization problems, dynamic optimization problems, multi-objective problems, and travelling salesman problems. Common algorithm performance evaluation measurements are also available. The source code of the AMP will be available in OFEC. The OFEC-v0.3.0 has been released on github at https://github.com/Changhe160/OFEC.

C. Experimental Investigation of the AMP

In this section, the AMP is investigated regarding its key parameters and mechanisms, based on AMP/PSO.

1) Sensitivity Analysis of Parameter $gSize$: To test the effect of varying the initial population size, $gSize$ was chosen from the range of 10 to 500. Fig. 4 presents the results of E_O, E_{BBC}, PR, and SR of AMP/PSO with different values of $gSize$ on the MPB with different numbers of peaks.

Fig. 4 shows that varying the value of $gSize$ in all MPB cases does not affect the results too much. This indicates that AMP/PSO has a very stable performance regardless of the initial population size. We would attribute the stable performance of AMP/PSO to its adaptation mechanism. The continuous adjustment to historical data (see Eq. (1)) would enable AMP/PSO to adjust the number of populations to an appropriate level (see evidences in Figs. 6, 8 and 9 later). Therefore, different choices of $gSize$ do not affect the performance of AMP/PSO, and $gSize = 100$ is used for both AMP algorithms in this paper.

2) Sensitivity Analysis of Parameter θ: For the convergence threshold (θ), the smaller it is, the more time will be spent on exploiting local optima. The larger the value of θ, in contrast, the more time will be spent on exploring new optima. To investigate the effect of this parameter on the performance of AMP/PSO, we conduct an experiment with different values of θ. Fig. 5 presents the results of AMP/PSO with different values of θ on the MPB with different numbers of peaks, where the darker the shade is, the better the results are.

Fig. 5 shows that varying the value of θ does affect the performance of AMP/PSO. For E_O, E_{BBC}, SR, in most cases the results get better as θ increases from $1E$-4 to $5E$-3 but then get worse as θ further increases. However, for PR, the smallest value of θ helps AMP/PSO obtain the best performance and the results of PR get worse as θ increases. This is because the measurement PR focuses more on exploitation of local optima than exploration of the global optima. AMP/PSO will spend the largest amount of time on exploiting local optima with the smallest value of θ and hence, the largest number of peaks is tracked. To take a trade-off between exploring new optima and exploiting local optima, θ was set to $5E$-3 for the AMP in all the other experiments in this paper.

3) Effect of the Adaptive Adjustment Mechanism: Fig. 6 shows the distribution of the number of populations of AMP/PSO and SAMO over 1,000 changes. For both algorithms, we take one sample when the number of populations changes. Therefore, the samples of AMP/PSO are the same data as in the database $\mathbb{D}$. Both algorithms are able to maintain a good correlation between the number of populations and the number of peaks. In problems where the number of peaks remain unchanged, each distribution curve has a very narrow shape. However, the performance of AMP/PSO is better than that of SAMO at least on problems with a small number of peaks (e.g., P=10 and 20), where the numbers of populations of AMP/PSO at the curve peaks are equal to the actual numbers of peaks. However, the corresponding numbers obtained by SAMO are larger than the actual numbers

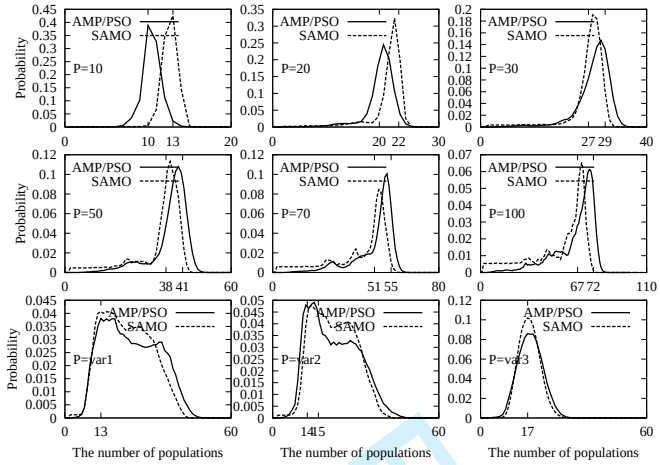

Fig. 6. Distribution of changes of the number of populations of AMP/PSO and SAMO over 1,000 changes.

TABLE IV
EFFECT OF VARYING THE PROBABILISTIC RANGE M ON THE MPB WITH DIFFERENT NUMBERS OF PEAKS, WHERE WAR IS A WRONG ACTION RATE AND NR IS THE NUMBER OF MAPS CREATED IN THE DATABASE $\mathbb{D}$.

P	WAR/NR				
	M=1	M=2	M=3	M=4	M=5
10	0.295/3720	0.188/3864	0.126/4027	0.097/4189	0.069/4239
20	0.320/2689	0.214/2904	0.128/2977	0.114/2985	0.091/2995
30	0.327/2000	0.224/2176	0.153/2310	0.114/2371	0.103/2292

of peaks. For problems with a large number of peaks, the numbers of populations at the curve peaks for both algorithms are smaller than the actual numbers of peaks. However, the numbers for AMP/PSO are closer to the actual numbers of peaks.

In problems with a varying number of peaks by a certain pattern (Var1 and Var2), for both algorithms the distribution curves now have a large band with multiple spikes, corresponding to the variation in the number of peaks of these problems. However, both algorithms do not show such a behavior in Var3 where the number of peaks changes without a pattern (further comparison of detailed changes of the number of populations can be seen later in Figs. 8 and 9).

4) Effect of the Probabilistic Prediction Scheme: To show the effect of using the probabilistic prediction scheme, an experiment was carried out to calculate the average wrong action rate for AMP/PSO with the probabilistic range M in [1,5] on the MPB problem with u=10,000. According to the results in Fig. 6, for problems with $P \leq 30$, we assume that the optimal number of populations is equal to the number of peaks. Hence, a wrong action is an action to increase/decrease the total number of individuals when the number of peaks is less/greater than the current number of populations. Table IV presents the wrong action rate (WAR) and the number of maps created in the database $\mathbb{D}$. The results show that using the probabilistic scheme $M > 1$ does help AMP/PSO greatly decrease the probability of taking a wrong action.

To further investigate the effect of using the probabilistic prediction scheme, an experiment was carried out on the ten static problems. Table V presents the results, where RE is the ratio of the number of evaluations ($eval$) to the largest $eval$ of each test group. Table V shows that AMP/PSO fails to find

TABLE V
PERFORMANCE COMPARISON OF AMP/PSO WITH DIFFERENT VALUES OF M ON MULTIMODAL FUNCTIONS, WHERE RE IS THE RATIO OF $eval$ TO THE LARGEST $eval$ OF EACH PROBLEM, NR IS THE NUMBER OF MAPS CREATED IN THE DATABASE $\mathbb{D}$.

Error	F(Opt.)	M=1	M=2	M=3	M=4	M=5	F(Opt.)	M=1	M=2	M=3	M=4	M=5
SR		**1**	**1**	**1**	**1**	**1**		0.967	**1**	**1**	**1**	**1**
RE	F1(10)	0.981	1	1	**0.962**	0.98	F2(36)	0.996	1	0.936	0.935	**0.884**
NR		33	47	26	**6**	23		194	132	66	99	**44**
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F3(6)	0.945	0.941	**0.938**	0.977	1	F4(18)	0.945	0.933	**0.835**	1	0.835
NR		27	22	21	18	**10**		**51**	67	56	88	60
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F5(8)	0.939	0.879	**0.874**	0.931	1	F6(4)	1	0.961	**0.92**	0.985	0.95
NR		61	38	**26**	46	83		40	**25**	34	29	29
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F7(4)	1	0.972	0.965	**0.956**	0.966	F8(5)	1	**0.96**	0.976	0.967	0.965
NR		**3**	**3**	**3**	**3**	**3**		39	23	**11**	18	18
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F9(5)	0.997	1	0.998	**0.978**	0.981	F10(3)	1	0.999	0.952	**0.932**	0.936
NR		13	11	14	**3**	8		4	**3**	4	4	6

TABLE VI
COMPARISON OF SR AND PR OF AMP/PSO WITH AND WITHOUT THE PEAK HIDING SCHEME ON FUNCTIONS F2 AND F4, WHERE $eval*$ IS THE NUMBER OF EVALUATIONS USED BY THE PEAK HIDING SCHEME AND $eval$ IS THE TOTAL NUMBER OF EVALUATIONS.

Function	Algorithm	SR	PR	$eval$	$eval*$
Vincent(F2,D=2)	AMP/PSO*	0.1	0.95	191937	0
	AMP/PSO	1	1	140139	102871
Shubert(F4,D=2)	AMP/PSO*	0.9	0.99	82437	0
	AMP/PSO	1	1	39144	11138

all the global optima of F2 with M=1. For most problems AMP/PSO with M=1 generates the largest number of maps in the database $\mathbb{D}$, and correspondingly, it also spends a relatively large number of evaluations. Although less maps are generated with $M > 1$ than that with M=1, they enable AMP/PSO to spend less time to find all the optimal peaks than that with M=1. That is, the probabilistic prediction scheme makes the learning more efficient than that without it. In this paper, M=3 is used for all the experiments as AMP/PSO with M=3 shows the best performance on most problems.

5) Effect of the Peak Hiding Scheme: The peak hiding scheme encourages individuals to explore un-discovered peaks. Table VI presents the comparison of AMP/PSO with the peak hiding scheme and without the peak hiding scheme (AMP/PSO*) on the Vincent (F2) and Shubert (F4) function in two-dimensional space. F2 and F4 contain 36 and 18 global optima, respectively. Table VI shows that AMP/PSO finds all peaks on the two functions for all runs. However, the results of SR and PR of AMP/PSO get worse when the peak hiding scheme is disabled. The only disadvantage of the peak hiding scheme is that extra evaluations are needed. However, the total function evaluations are still less than that of AMP/PSO without the scheme. Moreover, the peak hiding scheme greatly improves the performance of AMP/PSO. The peak hiding scheme is rarely triggered for the MPB problem as there is not enough time for populations to converge before changes occur. Therefore, the peak hiding scheme has little effect on the MPB problem.

D. Comparison with Peer Algorithms on the MPB Problem

TABLE VII
COMPARISON OF ERRORS OF E_O AND E_{BBC} ON THE MPB PROBLEM WITH DIFFERENT NUMBERS OF PEAKS

P	error	AMP/PSO	AMP/DE	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
10	E_O	**0.69**±0.03	0.9±0.1	2.4±0.3	4.8±1	4.4±0.5	1.9±0.3	5±0.3	5.2±0.3	2.6±0.2	2.1±0.3	2.5±0.5	3.4±0.5
	w,t,l	11,0,0	10,0,1	6,1,4	0,2,9	3,0,8	9,0,2	1,1,9	0,1,10	5,1,5	8,0,3	5,2,4	4,0,7
	E_{BBC}	**0.016**±0.01	0.069±0.1	1.2±0.3	3.5±1	3.1±0.5	0.8±0.3	0.96±0.3	1±0.3	1.2±0.3	1.2±0.3	1.1±0.6	1.7±0.5
	w,t,l	11,0,0	10,0,1	3,3,5	0,1,10	0,1,10	9,0,2	6,2,3	4,4,3	3,4,4	3,4,4	3,5,3	2,0,9
20	E_O	**1.1**±0.09	1.4±0.1	2.3±0.08	5.1±1	7.2±0.9	3.3±0.4	4±0.2	4.9±0.2	3.2±0.2	2.6±0.3	2.8±0.8	4.8±0.6
	w,t,l	11,0,0	10,0,1	9,0,2	1,2,8	0,0,11	5,1,5	4,0,7	1,2,8	5,1,5	7,1,3	7,1,3	1,2,8
	E_{BBC}	**0.28**±0.1	0.52±0.2	1.3±0.1	3.7±1	6.4±1	2.5±0.4	1.2±0.2	2±0.2	2.3±0.2	1.9±0.3	1.5±0.9	3.6±0.6
	w,t,l	11,0,0	10,0,1	7,1,3	1,1,9	0,0,11	3,0,8	9,0,2	5,1,5	4,0,7	5,1,5	7,1,3	1,1,9
30	E_O	**1.1**±0.05	1.3±0.06	2±0.04	4.6±1	4.9±1	2.3±0.2	3.2±0.2	3.8±0.2	2.8±0.09	2.3±0.3	2.2±0.4	3.3±0.3
	w,t,l	11,0,0	10,0,1	9,0,2	0,1,10	0,1,10	6,2,3	3,1,7	2,0,9	5,0,6	6,2,3	6,2,3	3,1,7
	E_{BBC}	**0.38**±0.04	0.49±0.05	1.1±0.04	3.2±1	3.9±1	1.6±0.2	0.97±0.1	1.2±0.2	2±0.1	1.7±0.3	1±0.4	2±0.2
	w,t,l	11,0,0	10,0,1	7,1,3	1,0,10	0,0,11	4,1,6	8,1,2	6,0,5	2,1,8	4,1,6	7,2,2	2,1,8
50	E_O	**1.2**±0.05	1.4±0.05	1.9±0.04	4.5±1	4.2±0.8	3.2±0.4	2.7±0.1	3.3±0.1	2.7±0.08	2.7±0.3	2.1±0.4	3.4±0.3
	w,t,l	11,0,0	10,0,1	9,0,2	0,1,10	0,1,10	2,2,7	5,2,4	3,1,7	5,2,4	5,2,4	8,0,3	2,1,8
	E_{BBC}	**0.61**±0.05	0.71±0.06	1.2±0.04	2.9±1	3.5±0.8	2.6±0.4	0.93±0.07	1.2±0.09	2±0.06	2.2±0.3	1.1±0.4	2.2±0.2
	w,t,l	11,0,0	10,0,1	6,0,5	1,1,9	0,0,11	1,1,9	8,1,2	7,1,3	5,0,6	3,1,7	7,2,2	3,1,7
100	E_O	**1.4**±0.05	1.7±0.08	2.1±0.06	4.3±0.8	5.4±1	3.3±0.4	2.4±0.07	3.2±0.1	3.2±0.1	3.5±0.6	2.2±0.4	4±0.2
	w,t,l	11,0,0	10,0,1	8,1,2	1,1,9	0,0,11	3,2,6	7,0,4	5,1,5	4,2,5	3,1,7	8,1,2	1,1,9
	E_{BBC}	**0.77**±0.04	0.9±0.07	1.4±0.06	2.7±0.6	4.5±1	2.6±0.4	1±0.06	1.4±0.08	2.5±0.1	2.9±0.6	1.1±0.5	2.7±0.2
	w,t,l	11,0,0	10,0,1	6,0,5	1,4,6	0,0,11	1,4,6	8,1,2	7,0,4	3,2,6	1,3,7	8,1,2	1,3,7
200	E_O	**1.8**±0.04	2±0.04	2.5±0.05	4.6±0.7	6.2±0.9	4.3±0.4	2.5±0.08	3.1±0.07	3.6±0.1	4.4±0.4	2.5±0.3	4.2±0.2
	w,t,l	11,0,0	10,0,1	8,1,2	1,1,9	0,0,11	2,2,7	7,1,3	6,0,5	5,0,6	1,2,8	7,2,2	3,1,7
	E_{BBC}	**1**±0.05	1.2±0.04	1.7±0.04	2.8±0.5	5.4±0.9	3.5±0.4	1.2±0.07	1.4±0.06	2.8±0.1	3.8±0.4	1.3±0.3	2.8±0.2
	w,t,l	11,0,0	9,1,1	6,0,5	3,2,6	0,0,11	2,0,9	9,1,1	7,1,3	3,2,6	1,0,10	7,1,3	3,2,6
w-l		132	107	44	-95	-123	-23	29	-14	-19	-20	48	-66

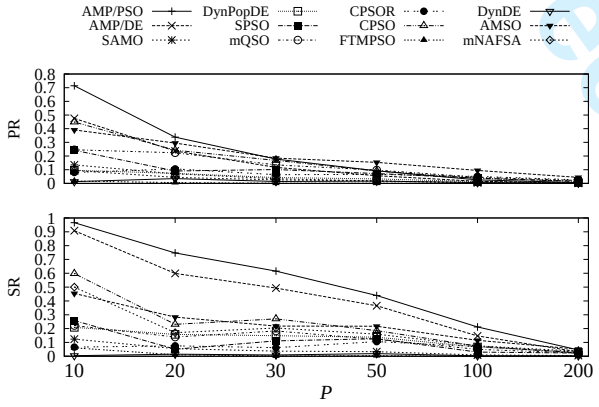

Fig. 7. Comparison of the peak ratio (PR) and success rate (SR) on the MPB with different numbers of peaks.

1) Effect of Varying the Number of Peaks: Table VII presents the offline errors and the best-before-change errors for all algorithms on the MPB with different numbers of peaks. Fig. 7 presents the comparison of the results of SR and PR for all peer algorithms. Fig. 8 plots the changes in the number of populations against time for AMP/PSO, SAMO, AMSO, and DynPopDE.

From Table VII, it can be seen that the results of AMP/PSO are significantly better than those of the other algorithms in all cases. Following with AMP/PSO, AMP/DE achieves the second best performance. The performance of the four adaptive algorithms (AMP/PSO, AMP/DE, SAMO, and AMSO) are better than that of those non-adaptive algorithms. Due to a large number of populations generated with DynPopDE, the algorithm achieves very poor performance compared with the other algorithms. Given the manually configured number of individuals, CPSOR also achieves a good performance.

Regarding the performance of tracking multiple peaks (see Fig. 7), AMP/PSO and AMP/DE achieve the best success rate (SR) in all cases. However, for the peak ratio, AMSO performs better than AMP/PSO and AMP/DE in the cases with many peaks. This is because many populations of the AMP have not converged yet when changes occur. The number of peaks traced will be increased if we give more time for the AMP to evolve before changes occur (see Fig. 10 in Sect. IV-D3 later).

Fig. 8 shows that all the four adaptive algorithms exhibit adaptive behaviors, where the number of populations at the end of the run increases as the number of peaks increases. For problems with a large number of peaks, AMP/PSO, DynPopDE, and SAMO shows similar behaviors where the number of populations gradually increases as the search goes on. Different from the above three algorithms, the number of populations obtained by AMSO quickly converges at a certain level in most cases. Among the four adaptive algorithms, DynPopDE generated the largest number of populations in all cases, followed by AMP/PSO, SAMO, and AMSO.

2) Comparison on Problems with a Varying Number of Peaks: Problems with a fixed number of peaks may be easy to solve. However, problems with a varying number of peaks will challenge an algorithm's adaptability. Table VIII and Fig. 9 present the errors of E_O and E_{BBC}, SR and PR, respectively, on problems with a varying number of peaks.

From Table VIII, AMP/PSO and AMP/DE achieve significantly better performance than all the other algorithms in all the cases. The adaptive algorithms SAMO and AMSO also achieves relatively good results in comparison with the other non-adaptive algorithms. Among the non-adaptive algorithms, the number of populations in CPSOR is configured according to the number of peaks, which makes it behave as an adaptive algorithm. Therefore, CPSOR performs as well as AMSO.

The four adaptive algorithms again show adaptive behaviors to the changing number of peaks, where the number of populations is basically synchronous with the change of the number of peaks in the cases of $Var1$ and $Var2$ (they also show such adaptive behavior in the case of $Var3$ if we observe the curves

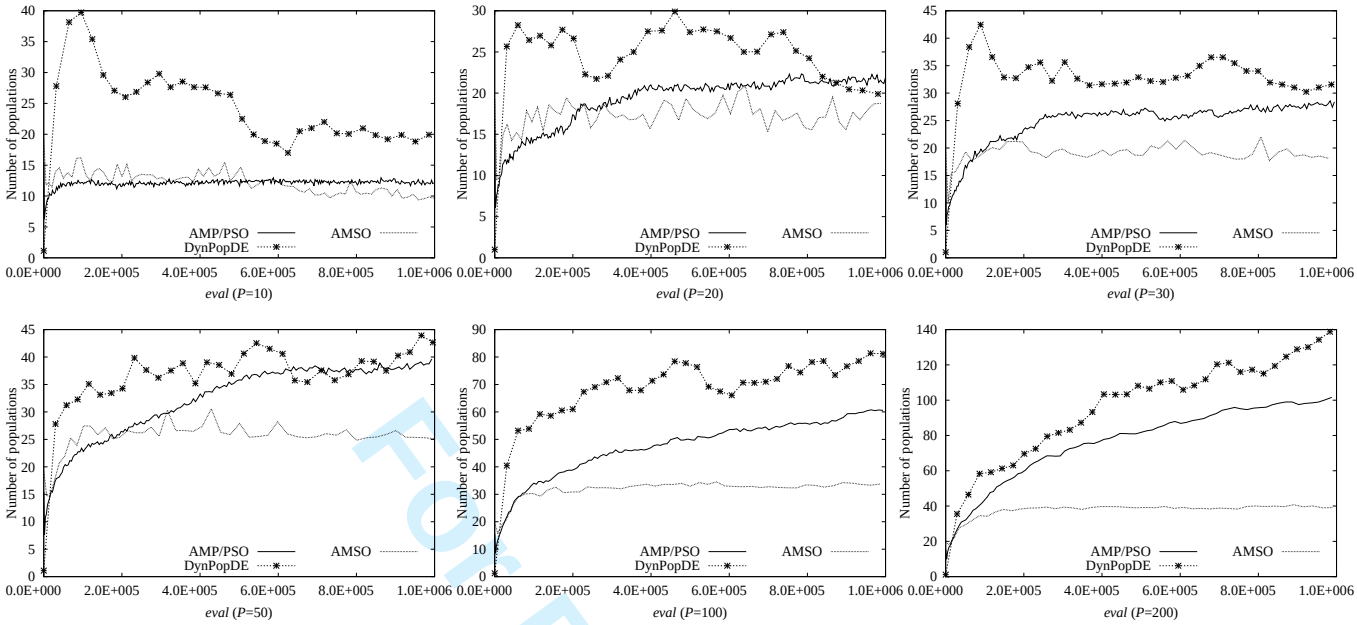

Fig. 8. Changes in the number of populations against time for four adaptive algorithms on the MPB with different numbers of peaks.

TABLE VIII
COMPARISON OF ERRORS OF E_O AND E_{BBC} ON THE MPB PROBLEM WITH A VARYING NUMBER OF PEAKS

P	error	AMP/PSO	AMP/DE	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
Var1	E_O	**1.8**±0.1	2±0.07	2.8±0.2	6.3±0.9	6±0.9	3.8±0.4	3.7±0.2	5.1±0.2	4.1±0.2	3.4±0.3	3±0.3	6.6±0.3
	w,t,l	11,0,0	10,0,1	9,0,2	0,2,9	1,1,9	5,0,6	6,0,5	3,0,8	4,0,7	7,0,4	8,0,3	0,1,10
	E_{BBC}	0.99±0.1	**0.99**±0.08	1.8±0.2	4.3±0.6	4.7±0.9	2.9±0.4	1.5±0.2	2.3±0.2	3.1±0.3	2.7±0.3	1.5±0.4	4.8±0.3
	w,t,l	10,1,0	10,1,0	7,0,4	2,0,9	0,1,10	4,0,7	8,1,2	6,0,5	3,0,8	5,0,6	8,1,2	0,1,10
Var2	E_O	**1.4**±0.08	1.6±0.09	2.2±0.1	5.7±1	6.2±0.6	3.3±0.5	3.1±0.1	3.9±0.2	3.1±0.2	3.1±0.3	3.2±2	5.6±0.4
	w,t,l	11,0,0	10,0,1	9,0,2	1,1,9	0,0,11	4,2,5	5,3,3	3,0,8	5,3,3	4,4,3	4,4,3	1,1,9
	E_{BBC}	**0.74**±0.08	0.86±0.1	1.4±0.1	4±1	5.2±0.7	2.6±0.5	1.2±0.08	1.5±0.2	2.5±0.2	2.4±0.3	2±2	3.8±0.4
	w,t,l	11,0,0	10,0,1	6,2,3	1,1,9	0,0,11	3,3,5	9,0,2	6,2,3	3,3,5	3,3,5	3,5,3	1,1,9
Var3	E_O	**1.9**±0.2	2.2±0.2	2.9±0.1	10±1	4.5±0.7	3.3±0.2	4.1±0.2	4.6±0.2	4±0.1	3.3±0.2	3.3±0.5	6.9±0.6
	w,t,l	11,0,0	10,0,1	9,0,2	0,0,11	2,1,8	6,2,3	4,0,7	2,1,8	5,0,6	6,2,3	6,2,3	1,0,10
	E_{BBC}	**1.2**±0.2	1.4±0.2	2±0.1	6.6±0.8	3.5±0.7	2.5±0.2	1.8±0.1	1.9±0.2	3.7±0.3	2.6±0.2	1.9±0.4	4.9±0.6
	w,t,l	11,0,0	10,0,1	6,2,3	0,0,11	2,1,8	4,1,6	9,0,2	6,2,3	2,1,8	4,1,6	6,2,3	1,0,10
w-l		65	55	30	-54	-52	-6	20	-9	-15	2	18	-54

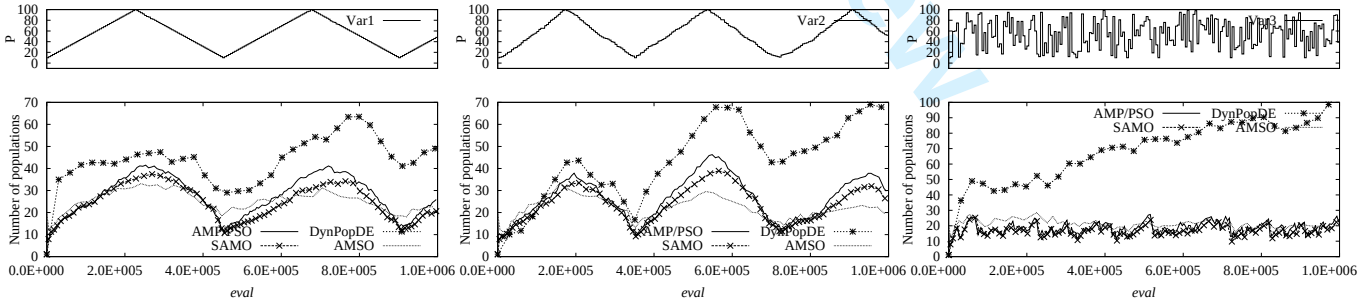

Fig. 9. Changes in number of populations against time for four adaptive algorithms on the MPB with a varying number of peaks.

with $Var3$ closely). Among these four algorithms, AMP/PSO and SAMO show the most similar behavior and they also show the best synchronization, which almost perfectly matches the changes of the number of peaks. DynPopDE again generates the largest number of populations, which makes it perform very poorly. Although all the adaptive algorithms show similar behaviors to AMP/PSO in terms of populations adaptation, they perform much worse than AMP/PSO regarding the errors E_O and E_{BBC}.

3) Effect of Varying the Change Frequency: Fig. 10 and Table IX present the results of SR and PR, and the errors of E_O and E_{BBC}, respectively, for all the involved algorithms. From Table IX, it can be seen that AMP/PSO achieves the best results in all cases, followed by AMP/DE. Although AMP/PSO and AMP/DE do not achieve the best results regarding the PR, they show the best results regarding the SR. AMSO shows very competitive performance regarding the PR. Increasing the change frequency means that algorithms will have more evaluations to locate and track optima before changes occur.

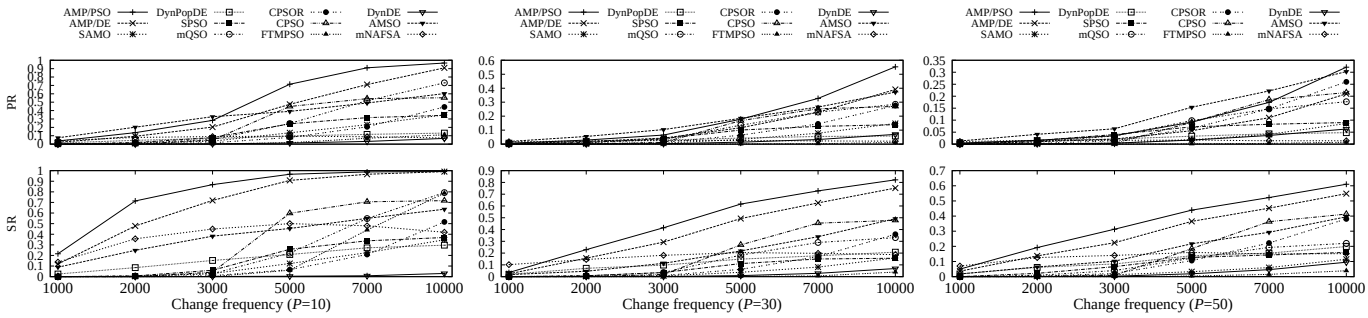

Fig. 10. Comparison of the peak ratio (PR) and success rate (SR) on the MPB with different change frequencies.

TABLE IX
COMPARISON OF ERRORS OF E_O AND E_{BBC} ON THE 200-PEAK MPB PROBLEM WITH DIFFERENT CHANGE FREQUENCIES

u	error	AMP/PSO	AMP/DE	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
1000	E_O	**3.6**±0.1	4.1±0.1	4.3±0.1	10±1	7.4±1	6.7±0.9	7.2±0.2	9.7±0.3	8±0.7	5.8±0.6	5.3±0.2	7.3±0.6
	w,t,l	11,0,0	10,0,1	9,0,2	0,1,10	3,2,6	6,0,5	3,2,6	0,1,10	2,0,9	7,0,4	8,0,3	3,2,6
	E_{BBC}	**2.3**±0.1	2.7±0.09	2.9±0.08	7.3±2	6.6±1	5.8±0.9	3.4±0.1	3.9±0.09	7±1	4.7±0.6	2.7±0.1	4.7±0.5
	w,t,l	11,0,0	9,1,1	8,0,3	0,2,9	0,2,9	3,0,8	7,0,4	6,0,5	0,2,9	4,1,6	9,1,1	4,1,6
2000	E_O	**2.7**±0.07	3±0.08	3.3±0.06	7.7±2	6.6±0.9	4.8±0.4	4.4±0.2	5.7±0.1	5.4±0.3	5±0.4	3.6±0.2	5.7±0.3
	w,t,l	11,0,0	10,0,1	9,0,2	0,0,11	1,0,10	5,1,5	7,0,4	2,1,8	4,0,7	5,1,5	8,0,3	2,1,8
	E_{BBC}	**1.7**±0.06	1.9±0.08	2.3±0.04	5.1±2	5.9±0.9	4±0.4	2.1±0.1	2.7±0.08	4.2±0.3	4.1±0.4	2±0.2	3.6±0.3
	w,t,l	11,0,0	9,1,1	7,0,4	1,0,10	0,0,11	3,1,7	8,0,3	6,0,5	2,1,8	2,2,7	9,1,1	5,0,6
3000	E_O	**2.2**±0.06	2.5±0.08	2.9±0.07	6.3±1	6.3±0.8	4.4±0.4	3.4±0.09	4.2±0.09	4.4±0.2	4.7±0.4	3±0.3	4.8±0.3
	w,t,l	11,0,0	10,0,1	9,0,2	0,1,10	0,1,10	4,1,6	7,0,4	6,0,5	4,1,6	2,1,8	8,0,3	2,1,8
	E_{BBC}	**1.4**±0.06	1.5±0.07	2±0.06	4±1	5.5±0.9	3.6±0.4	1.6±0.07	2±0.07	3.5±0.1	4±0.4	1.7±0.3	3.1±0.2
	w,t,l	11,0,0	10,0,1	6,1,4	1,2,8	0,0,11	2,2,7	8,1,2	6,1,4	3,1,7	1,1,9	8,1,2	5,0,6
5000	E_O	**1.8**±0.04	2±0.04	2.5±0.05	4.6±0.7	6.2±0.9	4.3±0.4	2.5±0.08	3.1±0.07	3.6±0.1	4.4±0.4	2.5±0.3	4.2±0.2
	w,t,l	11,0,0	10,0,1	8,1,2	1,1,9	0,0,11	2,2,7	7,1,3	6,0,5	5,0,6	1,2,8	7,2,2	3,1,7
	E_{BBC}	**1**±0.05	1.2±0.04	1.7±0.04	2.8±0.5	5.4±0.9	3.5±0.4	1.2±0.07	1.4±0.06	2.8±0.1	3.8±0.4	1.3±0.3	2.8±0.2
	w,t,l	11,0,0	9,1,1	6,0,5	3,2,6	0,0,11	2,0,9	9,1,1	7,1,3	3,2,6	1,0,10	7,1,3	3,2,6
7000	E_O	**1.6**±0.04	1.8±0.05	2.2±0.04	4.2±0.5	6.1±1	4±0.5	2.3±0.09	2.6±0.07	3.2±0.07	4.2±0.4	2.2±0.4	3.9±0.2
	w,t,l	11,0,0	10,0,1	8,1,2	1,2,8	0,0,11	2,2,7	7,1,3	6,0,5	5,0,6	1,1,9	7,2,2	3,1,7
	E_{BBC}	**0.86**±0.04	0.99±0.04	1.5±0.04	2.6±0.4	5.3±1	3.3±0.4	1.1±0.07	1.2±0.08	2.5±0.1	3.7±0.4	1.2±0.4	2.6±0.1
	w,t,l	11,0,0	10,0,1	6,0,5	3,2,6	0,0,11	2,0,9	9,0,2	7,1,3	4,1,6	1,0,10	7,1,3	3,1,7
10000	E_O	**1.3**±0.03	1.5±0.04	2±0.03	3.7±0.3	6.2±1	3.7±0.4	2±0.09	2.2±0.07	2.8±0.07	4±0.4	2±0.3	3.5±0.1
	w,t,l	11,0,0	10,0,1	7,2,2	2,1,8	0,0,11	2,1,8	7,2,2	6,0,5	5,0,6	1,0,10	7,2,2	4,0,7
	E_{BBC}	**0.68**±0.03	0.8±0.04	1.3±0.04	2.3±0.2	5.4±1	3.1±0.4	0.93±0.06	1.1±0.06	2.2±0.06	3.6±0.4	1.1±0.3	2.4±0.1
	w,t,l	11,0,0	10,0,1	6,0,5	4,0,7	0,0,11	2,0,9	9,0,2	7,1,3	5,0,6	1,0,10	7,1,3	3,0,8
w-l		132	105	51	-86	-119	-52	52	4	-40	-69	64	-42

Therefore, the performance of all the algorithms improves as the change frequency increases. It is interesting to see that AMP/PSO achieves the greatest improvement in PR among all the algorithms (Fig. 10), especially when $u > 5000$. This indicates that the AMP is able to make full use of the available evaluations to explore as many peaks as possible by adaptively adjusting the number of populations.

E. Comparison on Multi-modal Problems

In order to test the performance of the AMP in locating multiple peaks, an experimental comparison is carried out on ten multi-modal problems. In this paper, we do not choose the best and latest algorithms for the comparison since the motivation of this paper is for DOPs. To compare the performance of the AMP in static environments, two popular algorithms, DE/nrand/1 and CRDE proposed for multi-modal problems, were chosen. Table X presents the results of SR, PR, and the total number of evaluations spent ($eval$).

From Table X, AMP/DE and AMP/PSO outperform the other two algorithms on most problems. AMP/PSO successfully finds all peaks for every run on all problems. AMP/DE also obtains such good results as AMP/PSO except on F2, where one peak is not found for all runs. DE/nrand/1 also finds all peaks for all runs on F4, F7, and F10. However, it spends a much larger number of evaluations than AMP/PSO and AMP/DE on most problems. For the two AMP algorithms,

TABLE X
PERFORMANCE COMPARISON ON MULTI-MODAL FUNCTIONS IN TWO DIMENSIONS, WHERE $eval$ IS THE TOTAL NUMBER OF FUNCTION EVALUATIONS.

Problem	Error	AMP/PSO	AMP/DE	DE/nrand/1	CRDE
F1	PR/SR	**1/1**	**1/1**	0.61/0	0.163/0
	$eval$	**1.53e+004**	1.72e+004	2e+005	2e+005
F2	PR/SR	**1/1**	0.999/0.967	0.416/0	0.195/0
	$eval$	**1.4e+005**	1.7e+005	2e+005	2e+005
F3	PR/SR	**1/1**	**1/1**	0.667/0	0.317/0
	$eval$	2e+004	**1.94e+004**	2e+005	2e+005
F4	PR/SR	**1/1**	**1/1**	**1/1**	0.628/0.2
	$eval$	3.91e+004	6.29e+004	**3.2e+004**	1.68e+005
F5	PR/SR	**1/1**	**1/1**	0.9/0.367	0.188/0
	$eval$	**5.19e+004**	6.73e+004	1.32e+005	2e+005
F6	PR/SR	**1/1**	**1/1**	0.808/0.233	0.592/0.0333
	$eval$	**8.7e+004**	9.13e+004	1.56e+005	1.94e+005
F7	PR/SR	**1/1**	**1/1**	**1/1**	0.25/0
	$eval$	1.03e+004	**6.92e+003**	1.38e+004	2e+005
F8	PR/SR	**1/1**	**1/1**	0.6/0.0333	0.207/0
	$eval$	1.7e+004	**1.03e+004**	1.94e+005	2e+005
F9	PR/SR	**1/1**	**1/1**	0.987/0.933	0.2/0
	$eval$	6.76e+003	**5.48e+003**	1.9e+004	2e+005
F10	PR/SR	**1/1**	**1/1**	**1/1**	0.622/0.167
	$eval$	4.98e+003	**3.85e+003**	1.03e+004	1.68e+005

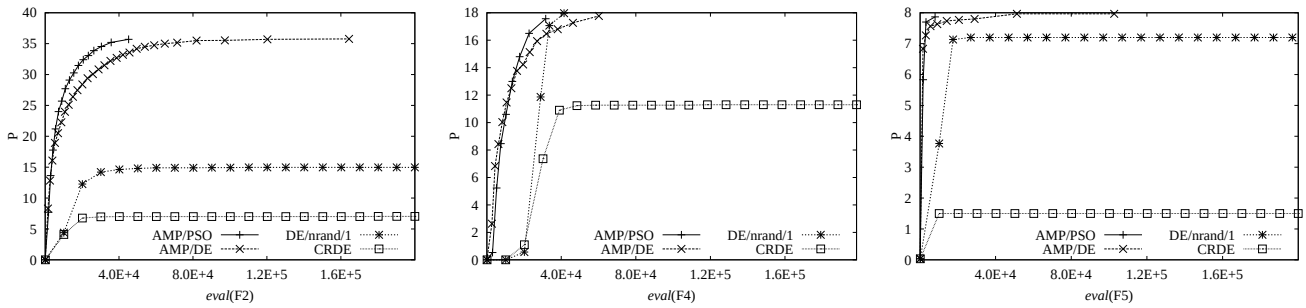

Fig. 11. Comparison of the progress of the number of peaks traced by the four algorithms on three multi-modal problems.

TABLE XI
THE PR AND SR OF AMP/PSO AND AMP/DE ON THE 3D AND 4D VINCENT FUNCTION, WHERE m_eval IS THE MAXIMUM NUMBER OF FUNCTION EVALUATIONS.

F(D,# of Opt.)	Algorithm	PR	SR	m_eval	F(D,# of Opt.)	PR	SR	m_eval
F2(3,216)	AMP/PSO	1	1	5.0E+06	F2(4,1296)	0.95	0	1.0E+07
	AMP/DE	0.9998	0.997	5.0E+06		0.78	0	1.0E+07
F4(3,81)	AMP/PSO	1	1	5.0E+06	F4(4,324)	0.828	0	1.0E+07
	AMP/DE	0.986	0.433	5.0E+06		0.705	0	1.0E+07

although they perform the same in SR and PR on nine out of ten problems, AMP/DE spends slightly less evaluations than AMP/PSO on most problems.

Fig. 11 presents the comparison of the progress of the number of peaks located by the four algorithms on three problems. From the results, it can be seen that AMP/PSO and AMP/DE are the quickest in locating multiple peaks.

In order to test the capability of locating many peaks of the AMP, AMP/DE and AMP/PSO are tested on the Vincent and Shubert function in 3D and 4D space with the corresponding maximum number of function evaluations of 5.0E+06 and 1.0E+07. Table XI shows the results of PR and SR of the two algorithms. For the 3D-Vincent function with 216 global optima, AMP/PSO successfully finds all peaks under the given maximum number of evaluations and AMP/DE misses only one peak in one run. For the 4D-Vincent function with 1296 global optima, although both algorithms fail to find all peaks under the given maximum number of evaluations, the peak ratio is 0.78 for AMP/DE and 0.95 for AMP/PSO, respectively. For the 3D-Shubert function with 81 global optima, AMP/PSO finds all the peaks for all runs and AMP/DE achieves a PR of 0.98 and a SR of 0.43. For the 4-D shubert function with 324 global optima, AMP/PSO and AMP/DE have no successful runs but they achieve the PR of 0.82 and 0.70, respectively.

V. CONCLUSIONS

Identifying the correct number of populations is a key issue to apply MPMs to solving DOPs. In order to address this issue, this paper proposes an adaptive multi-population framework. A database is used to record important information for guiding the adjustment of the total number of populations. Multiple populations are created by a heuristic clustering method without any manual inputs. Learning from historical data makes the AMP robust to solve problems, and continuously giving feedback to the database helps the AMP produce precise solutions.

From the experimental results of the two implemented algorithms with the AMP, several conclusions can be drawn. Firstly, the AMP is able to adaptively adjust the number of populations according to the number of peaks in the fitness landscape. Secondly, the AMP achieves the best performance among all the peer algorithms in most test cases, especially with regards to the capability of tracking multiple peaks. Thirdly, the AMP is also good at tracking multiple peaks in static environments.

Several interesting topics will be addressed for the future work. Firstly, the investigation on performance differences between AMP/PSO and AMP/DE on different problems should be performed. This would help to understand what kinds of EAs are good at solving what kinds of problems. Secondly, how to reduce the number of evaluations spent by the peak hiding method would also be interesting, especially in a high dimensional space. Thirdly, it is also interesting to make the peak hiding method work efficiently on problems with noise or with rugged fitness landscape. Finally, the application of the AMP to real-world problems is also important.

REFERENCES

[1] S. Bird and X. Li, "Adaptively choosing niching parameters in a pso," in *2006 Genetic and Evolutionary Computation Conference*, 2006, pp. 3–10.

[2] ——, "Using regression to improve local convergence," in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 592–599.

[3] T. Blackwell, "Particle swarm optimization in dynamic environments," in *Evolutionary Computation in Dynamic and Uncertain Environments*, ser. Studies in Computational Intelligence. Spinger, 2007, ch. 2, pp. 29–49.

[4] T. Blackwell and P. Bentley, "Don't push me! collision-avoiding swarms," in *2002 IEEE Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1691–1696.

[5] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computation*, vol. 3005. Springer Berlin Heidelberg, 2004, pp. 489–500.

[6] ——, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transaction on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[7] T. M. Blackwell and P. J. Bentley, "Dynamic search with charged swarms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 19–26.

[8] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *1999 IEEE Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1875–1882.

[9] J. Branke, T. Kaußler, C. Schmidth, and H. Schmeck, "A multi-population approach to dynamic optimization problem," in *4th International Conference on Adaptive Computing in Design and Manufacturing*, 2000, pp. 299–308.

[10] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, ser. Natural Computing Series, A. Ghosh and S. Tsutsui, Eds. Springer Berlin Heidelberg, 2003, pp. 239–262.

[11] I. del Amo, D. Pelta, Gonzáez, and J. lez, "Using heuristic rules to enhance a multiswarm pso for dynamic environments," in *2010 IEEE Congress on Evolutionary Conputation*, july 2010, pp. 1 –8.

[12] M. Epitropakis, V. Plagianakos, and M. Vrahatis, "Finding multiple global optima exploiting differential evolution's niching capability," in *Differential Evolution (SDE), 2011 IEEE Symposium on*, April 2011, pp. 1–8.

[13] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *SIMULATION*, vol. 76, no. 2, pp. 60–68, Feb. 2001.

[14] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *Cybernetics, IEEE Transactions on*, vol. 43, no. 3, pp. 881–897, 2013.

[15] Y. Jiang, W. Huang, and L. Chen, "Applying multi-swarm accelerating particle swarm optimization to dynamic continuous functions," in *Knowledge Discovery and Data Mining, 2009. WKDD 2009. Second International Workshop on*, jan. 2009, pp. 710–713.

[16] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A hibernating multi-swarm optimization algorithm for dynamic environments," in *World Congress on Nature and Biologically Inspired Computing, NaBIC2010*, 2010, pp. 363–369.

[17] A. P. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E. Talbi, "Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, may 2011, pp. 395–403.

[18] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing*, 2000, pp. 76–83.

[19] C. Li and S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," in *4th International Conference on Natural Comput.*, vol. 7, 2008, pp. 624–628.

[20] ——, "A clustering particle swarm optimizer for dynamic optimization," in *2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 439–446.

[21] ——, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Tran. Evolutionary Computation*, vol. 16, no. 4, pp. 556–577, 2012.

[22] X. Li, A. Engelbrecht, and M.G. Epitropakis, "Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization", *Technical Report*, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

[23] C. Li, S. Yang, and M. Yang, "An adaptive multi-swarm optimizer for dynamic optimization problems," *Evolutionary Computation*, vol. 22, no. 4, pp. 559–594, 2014.

[24] L. Liu, S. Yang, and D. Wang, "Particle swarm optimization with composite particles in dynamic environments," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 40, no. 6, pp. 1634–1648, 2010.

[25] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 564–567.

[26] ——, "Evolutionary swarm cooperative optimization in dynamic environments," *Natural Computing*, vol. 9, no. 1, pp. 83–94, 2010.

[27] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[28] R. Mendes and A. S. Mohais, "Dynde: a differential evolution for dynamic optimization problems," in *2005 IEEE Congress on Evolutionary Computation*, 2005, pp. 2808–2815.

[29] R. Mukherjee, G. R. Patra, R. Kundu, and S. Das, "Cluster-based differential evolution with crowding archive for niching in dynamic environments," *Information Sciences*, vol. 267, no. 0, pp. 58 – 82, 2014.

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

[31] T. T. Nguyen, I. Jenkinson, and Z. Yang, "Solving dynamic optimisation problems by combining evolutionary algorithms with kd-tree," in *Soft Computing and Pattern Recognition (SoCPaR), 2013 International Conference of*, Dec 2013, pp. 247–252.

[32] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *2004 IEEE Congress on Evolutionary Computation*, 2004, pp. 98–103.

[33] ——, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transaction on Evolutionary Computation*, vol. 10, no. 4, pp. 440–458, 2006.

[34] I. Rezazadeh, M. R. Meybodi, and A. Naebid, "Adaptive particle swarm optimization algorithm for dynamic environments," in *Proceedings of the Second international conference on Advances in swarm intelligence - Volume Part I*, ser. ICSI'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 120–129.

[35] I. Schoeman and A. Engelbrecht, "Using vector operations to identify niches for particle swarm optimization," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol. 1, Dec 2004, pp. 361–366 vol.1.

[36] ——, "Niching for dynamic environments using particle swarm optimization," in *Simulated Evolution and Learning*, ser. Lecture Notes in Computer Science, T.-D. Wang, X. Li, S.-H. Chen, X. Wang, H. Abbass, H. Iba, G.-L. Chen, and X. Yao, Eds. Springer Berlin Heidelberg, 2006, vol. 4247, pp. 134–141.

[37] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE Conference Evolutionary Computation*, 1998, pp. 69–73.

[38] M. C. du Plessis and A. P. Engelbrecht, "Differential evolution for dynamic environments with unknown numbers of optima," *Journal of Global Optimization*, pp. 1–27, 2012.

[39] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *2004 IEEE Congress on Evolutionary Computation*, vol. 2, 2004, pp. 1382–1389.

[40] A. M. Turky and S. Abdullah, "A multi-population harmony search algorithm with external archive for dynamic optimization problems," *Information Sciences*, vol. 272, no. 0, pp. 84 – 95, 2014.

[41] N. Unger, B. Ombuki-Berman, and A. Engelbrecht, "Cooperative particle swarm optimization in dynamic environments," in *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, April 2013, pp. 172–179.

[42] R. K. Ursem, "Multinational gas: Multimodal optimization techniques in dynamic environments," in *2000 Genetic and Evolutionary Computation Conference*, 2000, pp. 19–26.

[43] F. Van Den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, Department of Computer Science, University of Pretoria, South Africa, 2002.

[44] H. Wang, N. Wang, and D. Wang, "Multi-swarm optimization algorithm for dynamic optimization problems using forking," in *Control and Decision Conference, 2008. CCDC 2008. Chinese*, july 2008, pp. 2415–2419.

[45] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *Cybernetics, IEEE Transactions on*, vol. 45, no. 2, pp. 302–315, Feb 2015.

[46] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Transaction on Evolutionary Computation*, vol. 14, no. 6, pp. 959–974, 2010.

[47] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. Meybodi, and M. Akbarzadeh-Totonchi, "mnafsa: A novel approach for optimization in dynamic environments with global changes," *Swarm and Evolutionary Computation*, vol. 18, no. 0, pp. 38 – 53, 2014.

[48] 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," *Applied Soft Computing*, vol. 13, no. 4, pp. 2144 – 2158, 2013.

[49] X. Zheng and H. Liu, "A cooperative dual-swarm pso for dynamic optimization problems," in *Natural Computation (ICNC), 2011 Seventh International Conference on*, vol. 2, july 2011, pp. 1131 –1135.