Adaptive Control of Sub-Populations in Evolutionary Dynamic Optimization

Danial Yazdani, Ran Cheng, Cheng He, and Jürgen Branke,

***Abstract*—Multi-population methods are highly effective in solving dynamic optimization problems. Three factors affect this significantly: the exclusion mechanisms to avoid the convergence to the same peak by multiple sub-populations, the resource allocation mechanism which assigns the computational resources to the sub-populations, and the control mechanisms to adaptively adjust the number of sub-populations by considering the number of optima and available computational resources. In the existing exclusion mechanisms, when the distance (i.e. the distance between their best found positions) between two sub-populations becomes less than a predefined threshold, the inferior one will be removed/reinitialized. However, this leads to incapability of algorithms in covering peaks/optima that are closer than the threshold. Moreover, despite the importance of resource allocation due to the limited available computational resources between environmental changes, it has not been well studied in the literature. Finally, the number of sub-populations should be adapted to the number of optima. However, in most existing adaptive multi-population methods, there is no predefined upper bound for generating sub-populations. Consequently, in problems with large numbers of peaks, they can generate too many sub-populations sharing limited computational resources. In this paper, a multi-population framework is proposed to address the aforementioned issues by using three adaptive approaches: sub-population generation, double-layer exclusion, and computational resource allocation. The experimental results demonstrate the superiority of the proposed framework over several peer approaches in solving various benchmark problems.**

***Index Terms*—Dynamic optimization problems, Tracking moving optima, Multi-population, Computational resource allocation.**

I. Introduction

SEARCH spaces of many real-world optimization problems are dynamic and changing over time. These problems need to be solved online by an optimization method and are referred as dynamic optimization problems (DOPs) [1]. Several classes of DOPs have been investigated in the literature, such as combinatorial [2], [3], continuous [4], [5], multi-objective [6], [7], and constrained DOPs [8], [9]. This paper focuses on dynamic continuous unconstrained single-objective global optimization problems. For brevity, we use the term DOPs to refer to the considered DOPs in this paper. For solving DOPs, optimization methods not only need to find the global optimum but also track it after environmental changes [10]. DOPs are usually represented as follows:

$$F(\mathbf{x}) = f\left(\mathbf{x}, \theta^{(t)}\right), \tag{1}$$

where f is the objective function, $\mathbf{x}$ is a solution in the search space, $\theta^{(t)}$ are environmental parameters that change over time, t is the time index with $t \in [0, \hat{t}]$, and $\hat{t}$ is the number of environments of problem. In this paper, like most previous studies in the DOP domain, we focus on DOPs whose environmental changes happen discretely over time, i.e., $t \in \{1, \ldots, \hat{t}\}$ with static periods between environmental changes. For a DOP with $\hat{t}$ environmental states, there is a sequence of $\hat{t}$ stationary environments:

$$F(\mathbf{x}) = \left[f(\mathbf{x}, \theta^{(1)}), f(\mathbf{x}, \theta^{(2)}), \ldots, f(\mathbf{x}, \theta^{(\hat{t})})\right]. \tag{2}$$

The effectiveness of a DOP algorithm depends heavily on its speed in responding to the environmental changes and finding the new global optimum position. To this end, DOP algorithms must address some specific challenges of DOPs:

- Global diversity loss: this issue arises due to the nature of the evolutionary algorithms in converging to promising areas. In such a circumstance, the exploration capability of the algorithms will deteriorate significantly. Consequently, the algorithms become incapable of locating the new position of the global optimum when the relocation severity is intense.
- Local diversity loss: when a sub-population has converged to an optimum position (i.e. peak summit), its individuals are close to each other. Consequently, after an environmental change, when the optimum position has shifted, the sub-population may be incapable of tracking it. In such circumstances, the local diversity loss issue happens, which results in deteriorating the tracking and exploitation capabilities of the sub-population in the new environments.
- Limited computational resources: between successive environmental changes, it can be time consuming to perform over-exploitation or over-exploration without prioritizing more promising regions of the search space.

One of the most suitable and commonly used approach to tackle DOPs are multi-population methods [11], where more than one population are used to cooperatively locate and track multiple local optima simultaneously. By covering multiple optima/peaks, multi-population DOP algorithms can locate and track the global optimum more efficiently [11]. The performance of the multi-population methods is highly dependent on the sub-population control and computational resource

D. Yazdani, R. Cheng and C. He are with Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mails: danial.yazdani@gmail.com, ranchengcn@gmail.com, chenghehust@gmail.com).

J. Branke is with the Operational Research and Management Sciences Group in Warwick Business school, University of Warwick, Coventry CV4 7AL, United Kingdom (email: Juergen.Branke@wbs.ac.uk).

management. Different methods have been used to control sub-populations such as fixed number of sub-populations [4], regrouping methods [12], clustering based methods [13], and methods with adaptive number of sub-populations [14]. In all aforementioned methods, efficiencies of *covering* and performing *mutual exclusion* of peaks significantly affect the overall performance of the multi-population approach.

Since the number of peaks/optima is unknown in real-world applications or may change over time, the algorithms whose number of sub-populations cannot adapt to the number of peaks/optima are inefficient [15]. Adaptive multi-population methods have ameliorated this challenge by adapting the number of sub-populations to the number of discovered peaks [4]. However, a major shortcoming of the most adaptive multi-population methods is that there is no upper bound for the number of sub-populations. Therefore, in problems with large numbers of peaks, these methods may create a large number of sub-populations. As a result, their performance deteriorates due to the shortage of computational resources.

Another considerable issue with the multi-population methods is to establish mutual exclusion for each peak between sub-populations. Residing more than one sub-population on a peak deteriorates the performance by wasting computational resources. To address this issue, exclusion mechanism was introduced in [4]. Using this method, if the Euclidean distance between two sub-populations (i.e. the distance between their best found positions) becomes less then a threshold, the sub-population with inferior best found position will be removed/re-initialized. This mechanism with some minor modifications has been used in many DOP algorithms/frameworks [10], [11].

The performance of the exclusion mechanism is highly dependent on the value of its threshold. On one hand, if the exclusion threshold is set too low, it takes additional time before two sub-populations on the same peak are detected. As a result, a considerable amount of computational resource is wasted. On the other hand, large exclusion radius makes the algorithms incapable of covering peaks whose Euclidean distance is less than the threshold. In addition, if two or more peaks that are covered by tracker sub-populations become closer than the threshold, the tracker(s) on the worse peak(s) will be removed/reinitialized. Consequently, the coverage performance deteriorates. This issue arises due to the lack of considering type/role of the involved sub-populations. Furthermore, using Euclidean distance as the exclusion radius could cause difficulties due to the nature of the Euclidean norm. In fact, Euclidean norm considers all differences in dimensions together. Therefore, higher differences in peak locations in some dimensions can be dominated by lower distances in other dimensions. Consequently, peaks/sub-populations that are far away in some dimensions, but close in other dimensions, could be involved in the mutual exclusion.

Another important consideration in DOPs is to manage computational resources. This is vital due to the limited available computational resources between successive environmental changes. Despite the large body of literature on DOPs, little attention has been given to computational resource allocation between sub-populations. Indeed, most of the existing DOP algorithms simply used a Round Robin scheme for running sub-populations in each iteration [1], [10], [11], [16]. However, this method has some shortcomings which are consequences of the uniform distribution of computational resource among all sub-populations over time. Other existing computational resource allocation methods can be categorized into two groups of hibernating [15], [17]–[19] and progress based [20]–[22] approaches. In the hibernating methods, the Round Robin scheme is used but the sub-populations that have converged/stagnated would be removed from the Round Robin list. In the progress based methods, algorithms usually choose the sub-population with the best progress and rank to be run in the next iteration. Although these two resource allocation methods improved the performance of the algorithms, they have some shortcomings. On one hand, in the hibernating methods, algorithms do not take the type, role or rank of the sub-populations into the account. On the other hand, progress based approaches do not consider the convergence status or the type of the sub-populations. Moreover, the use of progress based methods deteriorated the tracking and covering capabilities of sub-populations that are residing on the peaks with lower fitness values.

In this paper, an adaptive control framework (ACF) for DOPs is presented to address the aforementioned DOP difficulties and considerations. ACF tries to cover and track the promising multiple moving optima using multiple sub-populations. By using multiple sub-populations, global diversity is systematically maintained. Consequently, the algorithm is capable of tracking the global optimum quickly if it relocates to the promising covered areas by the algorithm's sub-populations. One key issue in handling multiple sub-populations is how to adapt the number of sub-populations to the number of moving optima. ACF is an adaptive multi-population framework using three types of sub-populations: explorer, exploiter, and tracker. The explorer is used to explore the search space and find peaks. The exploiter further optimizes the results achieved by the explorer to exploit the peaks and move toward their summits while the explorer is reinitialized to find another peak. The tracker aims to cover the peak summit, and tracks it after environmental changes. Additionally, the tracker provides some information to estimate the number of peaks and shift severities. These three types of sub-populations cooperate in order to efficiently locate and track multiple optima. By discovering new peaks, additional sub-populations will be added to the search space effectively by adapting to the number of peaks. In problem instances with large numbers of peaks, the number of sub-populations is restricted to an upper bound threshold. The sub-population generation mechanism of ACF prevents creating further sub-populations when the number of existing sub-populations reaches the upper bound threshold. This is achieved by removing the worst sub-population when a new sub-population is initialized. Consequently, ACF maintains its exploration capability by initializing new sub-populations. Moreover, ACF utilizes an adaptive double-layer exclusion mechanism to meliorate losing trackers when their peaks become close. Additionally, this mechanism increases the possibility of discovering and covering peaks that are close.

Furthermore, ACF benefits from an adaptive computational resource allocation method that considers 1) roles of the sub-populations such as the best and the explorer sub-populations, 2) rank of the sub-populations based on their best found position, and 3) achievements of the sub-populations. Consequently, the sub-populations that have done their tasks will be removed from the resource allocation list, such that, more computational resources can be assigned to the remaining sub-populations.

In short, this paper makes the following major contributions:

- An adaptive sub-population generation method considering three types of sub-populations and an upper bound for the number of sub-populations.
- An adaptive double-layer exclusion mechanism that increases the capability of covering close peaks.
- An adaptive resource allocation method that considers role, task achievements, and rank of the sub-populations.

The rest of this paper is organized as follows. Section II covers the literature review on multi-population methods for DOPs. Section III describes the proposed framework in detail. Section IV conducts a comprehensive experimental studies on the proposed framework. Finally, Section V concludes this paper.

II. Related works

Tracking moving optima (TMO) using multi-population approaches is one of the most efficient and popular methods for tackling DOPs [1], [11], [16], [23]. Since this paper focuses on TMO using a multi-population framework, we only focus on the most relevant methods in which the multi-population approaches are utilized for TMO. More general surveys can be referred to [1], [11], [16], [23]

Self organizing scouts (SOS) [24] is the first method which is capable of fulfilling TMO. SOS uses two types of sub-populations, i.e., a sub-population for exploring the landscape and finding peaks, and a number of smaller sub-populations for exploiting peaks and tracking them. Later, the framework of SOS with some modifications has been utilized in many multi-population algorithms [25]–[28].

A parent-child approach, denoted as fast multi-swarm optimization (FMSO), is proposed in [29]. In FMSO, a large parent sub-population is responsible for finding promising areas. When the best found position by the parent sub-population has been improved, a new child sub-population is created around the best found position to exploit this area. Child sub-populations have fewer numbers of individuals in comparison to the parent sub-population. On the one hand, the core optimizer for the parent sub-population must be a diversity maintaining method with a high exploration capability. On the other hand, the child sub-populations should benefit from optimizers with fast convergence speeds and high exploitation capabilities. If any individual from the parent sub-population lies inside the search region of any child sub-population, it will be randomized.

Two PSO based multi-swarm algorithms, called mQSO and mCPSO, were proposed in [4]. In these two algorithms, the used methods to addresses the local diversity loss issue are different. On the one hand, in mCPSO, charged particles are used. The update rules of these particles include an acceleration part to avoid collision and to create repulsion between individuals. Consequently, the local diversity of each sub-population is maintained over time due to the created repulsion between individuals. On the other hand, in mQSO, quantum particles are used to maintain the local diversity of each sub-population over time. The quantum particles are generated randomly in a hyper-ball whose center is the best found position by the sub-population and its radius is defined with $r\mathrm{cloud}$. Except for the used methods that address the local diversity loss issue, other components of these two algorithms are identical. The number of sub-populations is fixed in both algorithms. To ensure continued search for possible uncovered promising peaks, an anti-convergence approach is utilized in which the worst sub-population is re-initialized when all the sub-populations have been converged. On one hand, a shortcoming of having a fixed number of sub-populations is that the algorithm wastes computational resources due to redundant sub-populations when the number of peaks is smaller than the number of sub-populations. On the other hand, the algorithm misses some peaks when the number of peaks is larger than the number of sub-populations. Moreover, an exclusion mechanism is utilized to avoid covering a peak with multiple sub-populations. In the exclusion mechanism, a threshold is used to determine the radius of the mutual exclusion. If the Euclidean distance between the best found positions of two sub-populations is less than the threshold, the inferior one is re-initialized. An improved version of mQSO is proposed in [30], in which the procedure of increasing diversity after each environmental change, is improved. To this end, individuals of each sub-population are sorted based on their fitness and divided into three groups. The individuals of the best group remain unchanged. The individuals of the second group are randomized around the best found position to address local diversity loss. And individuals of the worst group are randomized across the search space to increase the global diversity of the population.

Adaptive mQSO (AmQSO) [14] is the first adaptive multi-population method whose number of sub-populations is adapted to the number of discovered peaks. AmQSO starts with a single sub-population. After converging to a peak, another sub-population is created and initialized. Considering the unknown number of peaks, this algorithm performs a continual search to find uncovered peaks by initializing new sub-populations. Unlike mQSO that uses quantum particles during the course of optimization to maintain the local diversity of each sub-population over time, AmQSO only utilizes them right after each environmental change to rediversify each sub-population and address the local diversity loss issue. Consequently, unlike mQSO, AmQSO does not waste any computational resources to maintain the local diversity over time. In addition, to determine exclusion radius, AmQSO uses the number of sub-populations as the estimated number of peaks. However, this estimation is very sensitive to the accuracy of the convergence detection method and could be error prone. This exclusion mechanism has been used widely in designing DOP algorithms [11].

DynDE is a multi-population differential evolution (DE) with a fixed number of sub-populations [31]. This algorithm utilizes Brownian individuals around the best found position in order to maintain the diversity of sub-populations. Plessis and Engelbrecht [20] proposed an improved DynDE which utilizes a modified exclusion mechanism. In this mechanism, a mid-point is used in order to save some of the tracker sub-populations whose peaks' distance become less than the exclusion radius/threshold. If the fitness value of the mid-point is less than the best found positions of the involved sub-populations, then both would be kept. This mechanism can only cover the circumstances when the mid-point is located in the valley between the pair of involved peaks, and it cannot improve the performance significantly. Moreover, the improved DynDE uses a resource allocation mechanism which prioritizes the optimization of promising peaks. Later, two different resource allocation methods based on learning automata and performance index were added to DynDE [22]. One major shortcoming of the aforementioned DynDE methods is the fixed number of sub-populations. To address this shortcoming, Pleassis and Engelbrecht introduced DynPopDE [21] which is a DynDE with adaptive sub-population number.

FTmPSO [18] uses an adaptive multi-population approach in which a finder sub-population with a larger size is responsible for locating peaks. In addition, FTmPSO uses tracker sub-populations with smaller size to do local search and peak tracking. In FTmPSO, when the finder is converged, it creates a tracker sub-population. Finder convergence detection is determined based on the differences between the fitness values of its best found positions during a predefined time window. A sleep/awakening mechanism is used to deactivate sub-populations whose velocity is lower than a threshold to avoid wasting computational resources. In addition, a local search method is utilized to improve the exploitation process around the best found position.

Dynamic species-based PSO (DSPSO) [32] is the first regrouping based multi-population algorithm for DOPs. Each species in DSPSO is defined using the Euclidean distances between particles as the similarity metric, and a radius to determine species region. At the beginning of each iteration, all particles are sorted according their fitness values and species seeds are chosen sequentially from the best particles. A particle becomes species seed if its Euclidean distances to the current species seeds are more than the radius. Then, each species seed with the particles inside its radius forms a sub-population whose neighborhood best position is the species seed. Modified versions of DSPSO have been used for high-dimensional DOPs [33] and dynamic constrained optimization problems [9].

Some other methods use clustering approaches to create sub-populations [34]. In [13], a hierarchical clustering method is used for developing sub-populations after environmental changes. In addition, an overlapping detection mechanism is utilized to determine whether two sub-populations have been attracted to the same peak. The algorithm merges overlapped sub-populations and removed the redundant individuals. Moreover, the sub-populations whose diversities fall under a predefined threshold are deactivated. Besides, global diversity is increased after environmental changes. In [35], a clustering-based multi-population framework for undetectable DOPs, where detecting environmental changes is challenging, is proposed. In this method, the diversity is increased when a predefined portion of the initial population has been removed/deactivated. Thereafter, this algorithm increases diversity by keeping the previous best found positions and randomizing the rest of the population. Afterward, the clustering method is performed to form sub-populations. Since this method does not depend on the change detection, it needs to re-evaluate the memory (such as $Pbest$ positions in PSO [36]) in each iteration. AMP [15] is another clustering based method that uses an adaptive population size method based on the historical data. In addition, AMP uses a peak hiding technique to remove the attraction of the already discovered peaks.

Most existing multi-population methods utilize an exclusion method to guarantee that each peak is covered by not more than one sub-population. However, most existing exclusion mechanisms suffer from some shortcomings as stated in Section I. In addition, most multi-population algorithms do not consider the importance of a systematic resource allocation. As explained in Section I, the existing hibernating [15], [17], [18] and progress based [20]–[22] methods have some considerable flaws. Moreover, most existing adaptive multi-population algorithms do not consider an upper-bound for the number of sub-populations. Consequently, they suffer from the shortage of computational resources to support a large number of sub-populations. In the next section, a new DOP framework is proposed to address the above mentioned issues.

III. Proposed DOP framework

A. Motivations

As stated in Section I, the effectiveness of a DOP algorithm depends on its speed in responding to the environmental changes by addressing global and local diversity losses, and managing computational resources consumptions. Multi-population approaches usually try to maintain global diversity by utilizing multiple sub-populations. To maintain the exploitation capability of the sub-populations that are responsible for tracking optima, their local diversity must be adjusted after each environmental change. The diversity of trackers, must be increased based on the shift severity of peaks. Since the size of shift severities is unknown to the algorithm, it must be estimated using the gathered information by the sub-populations. However, another crucial decision is that how to select sub-populations to participate in gathering the information.

As stated in Section I, multi-population methods benefit from exclusion methods to prevent waste of computational resources due to performing exploitation with redundant sub-populations. However, an important challenge of designing an improved exclusion mechanism is to distinguish whether sub-populations residing on separate peaks or the same peak, especially when peaks are close to each other. Controlling and monitoring the cost of computational resources by different sub-populations must be considered in order to improve the effectiveness of DOP algorithms. To

this end, sub-populations must be prioritized based on their tasks, progress/achievements, or fitness. Accordingly, computational resources should be adaptively allocated to each sub-population in each iteration.

The rest of this section describes the proposed *adaptive control framework (ACF)* that addresses the above-mentioned considerations by: 1) an adaptive sub-population generation approach to cover and track multiple optima, and maintain the global diversity, 2) an adaptive double-layer exclusion mechanism to avoid exploiting a peak with more than one sub-population, 3) an adaptive resource allocation method to control the cost of computational resource by sub-populations, and 4) an adaptive change reaction mechanism to increase the local diversity of sub-populations based on the estimated shift severity. The details of these approaches are described in the rest of this section.

B. Adaptive sub-population generation

In ACF, sub-populations are categorized into three groups based on the diversity of their individuals, which is used to determine their convergence status. These three categories are: 1) sub-populations in the first stage of their life cycle, denoted as *explorers*, 2) sub-populations in their second stage of life cycle, denoted as *exploiters*, and 3) sub-populations in the third stage of their life cycle, denoted as *trackers*. Each sub-population in ACF is initialized randomly across the search space as an *explorer*, whose main responsibility is to explore the search space for finding a peak. When the diversity of an explorer becomes less than a threshold $\mathbf{R}$, ACF assumes that it has been attracted to a peak. This assumption is made on the basis of the observations that the individuals of converged populations are relatively closer to each other [4], which has been used in many multi-population methods for convergence detection [10], [11], [37]. Thus, when the diversity of an explorer becomes less than $\mathbf{R}$, it enters the second stage of its life cycle and becomes an *exploiter*. Figure 1(a) and 1(b) show an example of how an explorer becomes an exploiter as the diversity changes. The main responsibility of an exploiter is to move toward the peak summit. When an exploiter has reached close enough to the peak summit, it enters the third stage of its life cycle and turns into a *tracker*, controlled by a threshold $\mathbf{r}$. The main responsibilities of a tracker are to find the new peak summit position after environmental changes, and provide some information for estimating the shift severities and the number of optima.

The diversity in ACF is calculated in a dimension-wise manner, i.e. it is a D-dimensional vector where D is the number of problem dimensions. Diversity of a sub-population i in jth dimension is calculated as follows:

$$d_{i,j} = \max_{k \in pop_i} (|x_{k,j} - \bar{x}_{i,j}|), \tag{3}$$

where

$$\bar{x}_{i,j} = \frac{\sum_{k \in pop_i} x_{k,j}}{n}, \tag{4}$$

where $x_{k,j}$ is the position of kth individual of ith sub-population in jth dimension, $\bar{x}_{i,j}$ is the average of jth dimension of all individuals in ith sub-population, and n is the population size of each sub-population. In (3), $d_{i,j}$ is the largest distance in jth dimension between $\bar{x}_{i,j}$ and $x_{k \in pop_i, j}$.

As stated before, each sub-population in ACF has three stages in its life cycle, determined by its $\mathbf{d}$. The life cycle stage of a sub-population i is determined using the following formulation:

$$l_i = \begin{cases} 1 & \exists j \in \{1, \cdots, D\} : d_{i,j} > R_j \\ 2 & \forall j \in \{1, \cdots, D\} : d_{i,j} \leq R_j \wedge \exists j : d_{i,j} > r_j \\ 3 & \forall j \in \{1, \cdots, D\} : d_{i,j} \leq r_j \end{cases} \tag{5}$$

where l_i is the life cycle stage of ith sub-population, and $\mathbf{r}$ and $\mathbf{R}$ are two D-dimensional threshold vectors where $r_j < R_j$ for all dimensions $j \in \{1, \cdots, D\}$.

The efficiency of the convergence detection in ACF depends on the values of $\mathbf{r}$ and $\mathbf{R}$. In ACF, their values are calculated as:

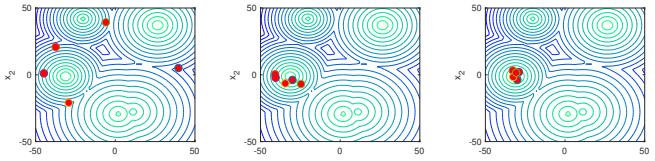
$$R_j = R' \frac{Ub_j - Lb_j}{\max(1, |N_{tr}|)}, \tag{6}$$

$$r_j = r' \frac{Ub_j - Lb_j}{\max(1, |N_{tr}|)}, \tag{7}$$

where R' and r' are two positive constants where $R' > r'$, Ub_j and Lb_j are upper and lower bounds of the jth dimension of the search space, N_{tr} is the set of trackers and $|N_{tr}|$ is the number of trackers, and $\max(1, |N_{tr}|)$ is used to avoid division by zero when there is no tracker.

According to (6) and (7), $\mathbf{r}$ and $\mathbf{R}$ are determined adaptively based on the estimated number of peaks ($|N_{tr}|$) and the search range in different dimensions. The reason that we use $|N_{tr}|$ instead of the total number of sub-populations is mainly attributed to the fact that the tracker sub-populations are expected to be very close to the peak summits. Consequently, their number is a suitable representative for the number of peaks. Therefore, the outputs of (6) and (7) are adaptive to the number of found peaks. In addition, these equations are suitable for generic problems with different search ranges in different dimensions or similar search ranges in all dimensions.

Using the above mentioned process, ACF starts with a randomized explorer and when it becomes an exploiter, a new explorer is initialized. Since there is no preliminary information about the number of peaks or their number may change over time, this procedure continues over the optimization process in order to locate and cover promising peaks. A shortcoming of the above mentioned sub-population generation method is that when the number of peaks is high, ACF may create too many sub-populations. As stated in Section I, this circumstance results in the shortage of computational resources for each sub-population. Therefore, the number of sub-populations must be bounded to an upper bound threshold $N_{\max}$. In ACF, when the number of sub-populations reaches $N_{\max}$, an anti-convergence mechanism [4] will be activated. By activating this mechanism, instead of creating a completely new sub-population, the sub-population with the worst best found position is re-initialized to become the new explorer. The proposed adaptive sub-population generating approach in this section makes the framework capable of covering up to $N_{\max}$ most promising peaks in the environments. In addition,



(a) $l_i = 1$ (b) $l_i = 2$ (c) $l_i = 3$

Fig. 1. An example of the three life cycles of a sub-population based on (5) when it is an explorer 1(a), after converging to a peak and become an exploiter 1(b), and when it converges to the peak's summit and becomes a tracker 1(c).

in a case where the global optimum is not covered, the exploration capability of the proposed framework is increased by frequently reinitializing and giving chances to the explorer sub-populations to converge to the highest peak.

C. Adaptive exclusion mechanism

As stated in Section I, establishing mutual exclusion around each optimum/peak is crucial to avoid the waste of computational resources. Exclusion mechanisms are utilized by the multi-population algorithms to address this issue. However, as described in Section I, most existing exclusion mechanisms are unable to locate and track peaks that are closer than the predefined thresholds. To address this issue, ACF utilizes a new adaptive double-layer exclusion mechanism that ameliorates the aforementioned shortcoming of the traditional single-layer exclusion mechanism. The exclusion mechanism of ACF works based on two different infinity norm distances $\mathbf{E}$ and $\mathbf{e}$ with $E_j > e_j : \forall j \in \{1, \cdots, D\}$. $\mathbf{E}$ and $\mathbf{e}$ are two thresholds where the former is related to a conditional mutual exclusion and the latter is related to a restricted and conservative one. The values of these two thresholds in each dimension are calculated as follows:

$$E_j = E' \frac{Ub_j - Lb_j}{\max(1, |N_{tr}|)}, \tag{8}$$

$$e_j = e' \frac{Ub_j - Lb_j}{\max(1, |N_{tr}|)}, \tag{9}$$

where E' and e' are two positive constants with $E' > e'$. To be specific, the procedure of the exclusion in ACF is designed based on the categories and roles of the sub-populations and their infinity norm distances.

Two sub-populations i and j enter the exclusion according to $\mathbf{e}$ if:

$$\forall k \in \{1, \cdots, D\} : |g^\star_{i,k} - g^\star_{j,k}| \leq e_k, \tag{10}$$

where $g^\star_{i,k}$ is kth dimension of the best found position by the ith sub-population. Moreover, two sub-populations i and j enter the exclusion according to $\mathbf{E}$ if:

$$\forall k \in \{1, \cdots, D\} : |g^\star_{i,k} - g^\star_{j,k}| \leq E_k, \exists k : |g^\star_{i,k} - g^\star_{j,k}| > e_k, \tag{11}$$

According to the ACF exclusion procedure in Alg. 1, if two sub-populations enter the exclusion based on $\mathbf{e}$ (line 1), one of them will be removed or re-initialized. When two

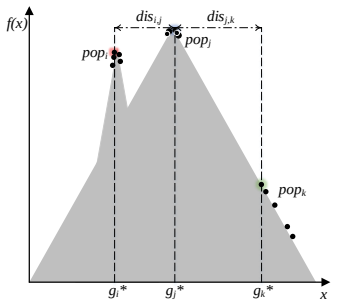


Fig. 2. An example of three sub-populations, including two trackers pop_i and pop_j and an exploiter pop_k. In the traditional single-layer exclusion mechanism, if the threshold of the exclusion is smaller than $dis_{j,k}$, both pop_i and pop_k will be discarded. By contrast, in the proposed double-layer exclusion mechanism, it is tried to keep the pop_i and discard pop_k.

sub-populations enter the exclusion area of each other based on $\mathbf{E}$ (line 6), several situations can happen based on their types. Trackers are robust to this layer of exclusion (line 8). Moreover, for two involved exploiters, the inferior one will be removed (line 10). When an explorer or exploiter enters the mutual exclusion with a tracker, two situations can happen: 1) when the tracker has a better $f(\mathbf{g}^\star)$, it is highly possible that the explorer/exploiter is moving toward the same peak, such that the explorer/exploiter will be re-initialized/removed (line 13) to save computational resources; 2) when the tracker has a worse $f(\mathbf{g}^\star)$, it is highly possible that the explorer/exploiter is moving toward another peak, so both sub-populations can remain (line 15). Furthermore, when the explorer enters the mutual exclusion with an exploiter, the explorer will be re-initialized since in that area there is an exploiter moving toward an optimum (line 17).

The main purpose of $\mathbf{e}$ is to control mutual exclusion between trackers. Therefore, e' is set to smaller values to avoid removing trackers when their peaks are close together but not closer than $\mathbf{e}$. $\mathbf{E}$ is a threshold for controlling mutual exclusion when explorer and exploiters are involved. The values of $\mathbf{E}$ are larger than $\mathbf{e}$ to avoid that explorers or exploiters move toward a peak which is already covered by a tracker. On one hand, the exclusion mechanism of ACF tries to avoid losing precious trackers whose under covered peaks are closer than the outer layer threshold $\mathbf{E}$. On the other hand, it tries to avoid wasting valuable computational resources when *non-tracker* sub-populations are closer than $\mathbf{E}$.

D. Adaptive resource allocation

When there are several sub-populations in the search space, it is very important to manage their computational resource consumption. In this section, we propose an adaptive computational resource allocation by considering the following factors;

a) Role: The tracker residing the best peak is very important since the performance of the framework in each environment is highly dependent on the best tracker performance. In addition, the role of the explorer sub-population is very significant since it is responsible for finding uncovered peaks. If an uncovered peak becomes the best peak in an environment, it can considerably deteriorate the performance. Therefore, a

Algorithm 1: Adaptive double-layer exclusion mechanism.

```
if pop_i and pop_j has entered the exclusion based on e in (10) then
    if One of them is explorer then
        Reinitialize the explorer;
    else
        Remove the one with worse g*;
if pop_i and pop_j has entered the exclusion based on E in (11) then
    if both are trackers then
        Do nothing;
    else if both are exploiter then
        Remove the one with worse g*;
    else if pop_i is a tracker and pop_j is a non-tracker then
        if g_i* is better than g_j* then
            Remove/re-initialize pop_j;
        else
            Do nothing;
    else if pop_i is the explorer and pop_j is an exploiter then
        Re-initialized pop_i;
```

suitable amount of computational resource must be allocated to the explorer to increase the efficiency of finding uncovered peaks.

b) Distance to peak summit: the main task of the trackers is to track peak summits after environmental changes. Except the best tracker, the rest of them do not need to continue exploitation after they get close enough to the peak summit. In such a situation, continuing exploitation for non-best trackers does not noticeably influence the overall performance, while it considerably consumes computational resources.

c) Rank: After an environmental change, the best tracker according to the $f(\mathbf{g}^\star)$ may not be the same as the tracker on the best peak. This can happen due to the characteristics of peaks such as height and gradient. For example, the ratio between the distance to the peak summit and the fitness value can be significantly different in narrow and wide peaks. Therefore, it is important to assign computational resources to some better sub-populations (not only the best) according to their best found position. However, even the inferior sub-populations must be considered in the computational resource allocation process. In fact, despite the low quality of some trackers, it is important to maintain a short distance to their peak summits in order to achieve an acceptable tracking efficiency. In other words, when the distance of a tracker to its peak summit becomes larger, it is possible that the tracker will be attracted to another nearby peak, thus missing its own peak. Therefore, it is important to allocate computational resources to all sub-populations; however, the amount and the priority must be considered.

d) Quick recovery: In many real-world applications, after an environmental change, a new solution must be achieved [1] sooner than the next environmental change.

ACF benefits from an adaptive resource allocation mechanism to address the above mentioned considerations. This mechanism allocates the computational resources to $|N|$ sub-populations based on their $f(\mathbf{g}^\star)$ ranks, their types (i.e. explorer and best sub-population), and their task achievements. In each iteration of ACF, the procedure of the core optimizer (e.g. PSO or DE) is performed for some *active* sub-populations that are chosen using a selection method. The activity status of a sub-population is determined based on a threshold a'. If all dimensions of the diversity vector of a sub-population fall below a', it will be deactivated until the end of the current environment [19]. Consequently, no computation resource is allocated to the inactivated sub-populations. In fact, when the diversity of a sub-population i is less than a' in all dimensions, it means that it has probably converged to its target. Therefore, ACF assumes that this sub-population has achieved its goals in the current environment by getting close enough to the new position of the peak summit. Therefore, to avoid unnecessary over-exploitation, the resource allocation mechanism does not consider the deactivated sub-populations.

The resource allocation mechanism prioritizes the active sub-populations based on their ranks. At the beginning of each iteration, a selection procedure based on the rank of sub-populations is performed. Only the active exploiters and trackers participate in the selection process. First, a set T_i is created for each active exploiter or tracker i as

$$T_i = \bigcup_{j=1}^{|T_i|} \{o_{i,j}\}, \tag{12}$$

where $o_{i,j}$ is a token belonging to the active exploiter or tracker i, and $|T_i|$ is calculated as:

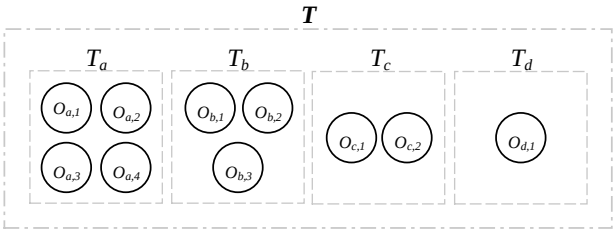
$$|T_i| = |N_{ac}| - rank_i + 1, \tag{13}$$

where N_{ac} denotes the set of all active trackers and exploiters, and $rank_i$ is the rank of the ith active sub-population based on $f(g_i^\star)$. Therefore, the best active sub-population is ranked first and will have $|N_{ac}|$ tokens and the worst sub-population will have one. All T_i are accumulated in a set T as:

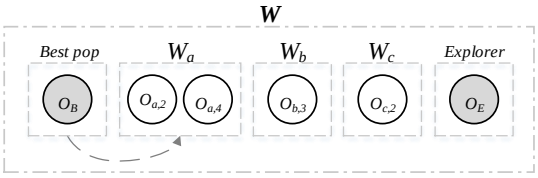
$$T = \bigcup_{i=1}^{|N_{ac}|} T_i, \tag{14}$$

where T is the set of all tokens and $|T| = \frac{|N_{ac}|(|N_{ac}|+1)}{2}$. Then, $|N_{ac}|$ tokens are chosen randomly from T without replacement which form a set of selected tokens W. Then, each tracker or exploiter i can perform $|W_i|$ internal iterations in the current iteration of ACF where $W_i \subseteq T_i$ is the set of tokens belonging to the tracker or exploiter i that has been selected in the random selection. If $|W_i| > 1$, the tracker or exploiter i performs $|W_i|$ internal iterations; however, if it becomes deactivated during this period, it will skip its internal iterations. In fact, if an exploiter or tracker becomes deactivated or removed by the exclusion mechanism, its possible remaining selected tokens are removed from W.

The explorer is excluded from the selection process since its rank can be worse than most sub-populations. In fact, the crucial task of the explorer, i.e. finding peaks, is independent from its $f(\mathbf{g}^\star)$ rank. Therefore, the explorer performs an internal iteration in each iteration of ACF to maintain the capability



(a) Constructing T by (14) based on (12) and (13). There are 4 active sub-populations $N_{ac} = \{a, b, c, d\}$ where $f(\mathbf{g}^\star_a) > f(\mathbf{g}^\star_b) > f(\mathbf{g}^\star_c) > f(\mathbf{g}^\star_d)$.



(b) Constructing W by randomly choosing $|N_{ac}| = 4$ tokens from T (white circles) and adding a token for the best sub-population and another token for the explorer (grey circles). In a case that the best sub-population has not been deactivated, sub-population a is the same as the best sub-population, then $W_a = \{o_{a,2}, o_{a,4}, o_B\}$.

Fig. 3. An example of the selection procedure in Alg. 2.

Algorithm 2: Selection procedure.

```
Sort all {pop_i ∈ N_ac};
foreach {pop_i ∈ N_ac} do
    Create T_i Using (12);
Create T Using (14);
W ← Choose |N_ac| tokens from T randomly;
pop_best ← Determine the sub-population whose g* is the best;
pop_explorer ← {pop_i | l_i = 1};
W ← W ∪ pop_explorer ∪ pop_best;
```

of finding uncovered peaks. In addition to the explorer, the role of the best sub-population in each environment is vital due to its significant influence on the performance of the framework. The best sub-population contributes in the selection process and possesses the highest number of tokens until it becomes deactivated. On one hand, deactivating the best sub-population degrades the performance of the framework. On the other hand, maintaining its involvement in the selection process without considering its activity status leads to the assignment of too much computational resources to it. To address this issue, similar to the explorer and independent from the selection process, the best sub-population performs an internal iteration of the core optimizer in each iteration of ACF. Therefore, in each iteration of ACF, $|N_{ac}| + 2$ sub-populations execute one iteration of the core optimizer. The procedure of the resource allocation method is illustrated in Alg. 2.

E. Environmental changes and ACF reactions

ACF is a reaction based framework [11]. Consequently, it needs to react to the environmental changes in order to maintain its efficiency. Thus, it is necessary to detect environmental changes, or get informed about them. Since detecting environmental changes is a separate issue, in this paper, it is assumed that the algorithms are informed about environmental changes which is the case of many real-world DOPs [10], [38]. Section S-IV of the supplementary document discusses the change detection mechanisms and incorporates a representative one into the ACF. After each environmental change, ACF performs the following actions: 1) shift severity estimation, 2) increasing diversity, 3) re-evaluating all stored solutions, and 4) activating all inactivated sub-populations.

In ACF, to address the local diversity loss issue, the local diversity of each tracker is increased right after each environmental change. One crucial matter in designing this mechanism is the degree of local diversity that should be injected. On the one hand, if the local diversity is increased too much, it delays exploitation. On the other hand, if the local diversity is too low, the tracking is hampered and the individuals may collapse on a position before reaching the optimum (i.e. peak summit). One suggestion is to increase the local diversity based on the shift severity values (i.e., the peak relocation length) [4]. To estimate the shift severity values, the Euclidean distances between the best found positions by each sub-population in the successive environments are usually used in the literature [10]. However, in real-world problems, different variables may have different shift severity values due to their specific nature (e.g. temperature, pressure, and speed). Therefore, in ACF, the shift severity value is estimated in each dimension separately. To this end, ACF considers the estimated shift severity as a D-dimensional vector calculated using the $g^\star$ position of trackers:

$$\delta_j = \frac{1}{t-1}\sum_{\acute{t}=2}^{t}\left(\frac{1}{|\ddot{N}_{tr}^{(\acute{t})}|}\sum_{i\in\ddot{N}_{tr}^{(\acute{t})}}\left(\left|{g^\star_{i,j}}^{(\acute{t})} - {g^\star_{i,j}}^{(\acute{t}-1)}\right|\right)\right), \tag{15}$$

where δ_j is jth dimension of the estimated shift severity vector δ, t is the current environment number, $\ddot{N}_{tr}^{(\acute{t})}$ is the set of trackers in $\acute{t}$th environment which at least experienced another environmental change $\left(\ddot{N}_{tr}^{(\acute{t})} \subseteq N_{tr}^{(\acute{t})}\right)$ and $|\ddot{N}_{tr}^{(\acute{t})}|$ is its cardinality, and ${g^\star_{i,j}}^{(\acute{t})}$ is jth dimension of the best found position by the ith tracker in $\acute{t}$th environment where $i \in \ddot{N}_{tr}^{(\acute{t})}$. Therefore, δ is the average of dimension-wise distances of the best found positions in successive environments by trackers $i \in \ddot{N}_{tr}$ until the current environment.

It is very important to appropriately increase the diversity of the trackers in order to maintain their tracking capabilities. Using δ is a suitable way to adapt the diversity of trackers to the new peak summit positions. In ACF, in order to increase the diversity of the tracker i, the ${\mathbf{g}^\star}_i$ position is kept as an individual (e.g. particles in PSO), and the rest of them will be randomized using the following formula:

$$x_{i,j,k} = {g^\star_{i,j}}^{(t-1)} + r\{-1, 1\}\delta_j \tag{16}$$

where $x_{i,j,k}$ is jth dimension of kth individual in ith tracker, ${g^\star_{i,j}}^{(t-1)}$ is the jth dimension of the best found position by ith tracker in the previous environment, δ_j is the estimated shift severity in jth dimension, and $r\{-1, 1\}$ generates a random

Algorithm 3: ACF change reaction.

```
1 Update shift severity vector using (15);
2 Activate all sub-populations;
3 foreach tracker i do
4     Keep g_i^⋆ as an individual;
5     Randomize the rest of individuals based on (16);
6 forall pop_i do
7     Re-evaluate stored positions and update memory;
```

sign (-1 or 1) with uniform distribution. By (16), the individuals are randomized on the corners of a hyper-rectangle whose center is $\mathbf{g}^{\star}{}_i$ and its sides are $2 \times \delta$. Consequently, using (16), ACF addresses the local diversity loss issue by increasing the local diversity of each tracker after each environmental change, which results in regaining its exploitation capability in the new environment. Note that since the diversity of the explorer and exploiters are not low, their individuals are kept unchanged.

After rediversifying the trackers, all individuals in all sub-populations including the explorer, exploiters and trackers are re-evaluated to update the stored fitness values and address the outdated memory issue [11]. Note that, at the end of each environment, the best found positions by trackers can be considered as representatives of the optima positions. Therefore, by keeping their $\mathbf{g}^{\star}$ from the previous environment and randomizing other individuals around them, the framework tries to utilize the obtained information from the previous environment about the optima positions to accelerate the search of new global optimum. The pseudo code of the reaction procedure of ACF to the environmental changes is illustrated in Alg. 3.

F. Procedure of ACF

Alg. 4 shows the work-flow of the proposed framework (a more detailed flow chart is given at the end of this document). As shown in Alg. 4, ACF starts with a randomized sub-population (line 1). At the beginning of each iteration, the resource allocation mechanism selects the sub-populations that are going to consume computational resources in the current iteration (line 3). For each selected token in the selection process, an internal iteration of the core optimizer (e.g. PSO and DE) is executed for its owner (5). Then, the exclusion will be checked for the sub-populations (line 7) and if any sub-population has been removed by the exclusion mechanism, its possible selected tokens will be cancelled (line9). In case that a sub-population becomes deactivated in line 10, its possible remained selected tokens will be cancelled (line12). According to the maximum number of sub-populations $N_{\max}$, when the explorer becomes an exploiter, a new explorer will be initialized by creating a new sub-population (line17) or re-initializing the worst one (line15). At the end of each iteration of ACF, the change reaction is done if an environmental change has been reported. Note that if an environmental change is reported/detected, ACF will skip other processes and run the change reaction procedure immediately.

The time complexity of ACF is $O(|N_{ac}| \cdot N \cdot D)$, which is almost similar to many existing DOP algorithms [11].

Algorithm 4: ACF procedure.

```
1  Initialize pop_1;
2  repeat
3      W ← Alg.2 ;
4      foreach selected token o_k ∈ W do
5          Execute the core optimizer for {pop_i | o_k ∈ T_i};
6          Update l_i using (5);
7          Check exclusion between pop_i and others using
             Alg. 1;
8          if any pop_j has been removed by exclusion then
9              Remove W_j from W;
10         Update activity status pop_i;
11         if pop_i is deactivated then
12             Remove W_i from W;
13     if {∄pop_i | i ∈ {1, ··· , N} ∧ l_i = 1} then
14         if N = N_max then
15             Reinitialize pop_worst with the worst g^⋆ as the
                 explorer;
16         else
17             Initialize a new pop as the explorer;
18     Update r, R, e, E using (7), (6), (9) and (8);
19     if Environment has changed then
20         React to change using Alg. 3;
21 until stopping criterion is met;
```

The detailed complexity analysis of ACF can be found in Section S-I of the supplementary document.

IV. Experiments and analysis

A. Benchmark functions

In this paper, three different baseline functions of moving peaks benchmark (MPB) [5], DF1 [39], [40], and Gaussian peaks benchmark (GPB) [41], are used to evaluate the performance of algorithms. The details of the aforementioned benchmark functions can be found in Section S-II of the supplementary document.

B. Performance evaluation

To measure the performance of the algorithms, we use the offline-error [42] (E_O), which is the most well-known performance indicator in the DOP literature. E_O is the average error of the best found position over all fitness evaluations:

$$E_O = \frac{1}{\hat{t}\vartheta} \sum_{t=1}^{\hat{t}} \sum_{b=1}^{\vartheta} \left(f^{(t)}\left(\mathbf{x}^{\star(t)}\right) - f^{(t)}\left(\mathbf{x}^{*((t-1)\vartheta+b)}\right)\right), \tag{17}$$

where $\mathbf{x}^{\star(t)}$ is the optimum position at the tth environment, $\hat{t}$ is the number of environments, ϑ is the change frequency, b is the fitness evaluation counter for each environment, and $\mathbf{x}^{*((t-1)\vartheta+b)}$ is the best found position at bth fitness evaluation in the tth environment.

C. Comparison Algorithms

A set of 10 different DOP algorithms is chosen for comparisons which are listed in Table I. Different population

TABLE I
COMPARISON ALGORITHMS.

Algorithm	Ref.	Optimizer
mCMA-ES	[10]	CMA-ES [44]
AmQSO	[14]	PSO [43]
CPSO	[13]	PSO [43]
DSPSO	[12], [14]	PSO [43]
mPSO	[10]	PSO [43]
mQSO	[4]	PSO [43]
RPSO	[45]	PSO [43]
DynPopDE	[21]	DE/best/2/bin [31]
mbDE	[10]	DE/best/2/bin [20], [31]
mjDE	[10]	jDE [46], [47]

configurations are used among these methods [11], [16], for example mQSO is a multi-population algorithm with constant number of sub-populations, AmQSO, DSPSO, and CPSO are the originators of the adaptive number of sub-populations, re-grouping approaches, and clustering methods, respectively, and RPSO is a single population method that uses randomization approach after environmental changes.

To have a fair comparison, some modifications are done to the algorithms as follows:

- The procedure of change detection is removed from all methods. It is assumed that algorithms are informed about the environmental changes like many real-world cases [1]. All algorithms react immediately to changes.
- All PSO based algorithms use PSO with constriction factor [43].
- The shift severity estimation method from [38] is adopted in all methods that require knowledge about the shift severity (such as AmQSO).

D. Parameter Settings

The parameter settings of the benchmark problems are shown in Table II. The experiments are done on the problem instances with different peak shapes, dimensions, number of peaks, change frequencies, and shift severities.

We use PSO with constriction factor [43] and DE (DE/best/2/bin) [20] as the embedded core optimizers in the proposed framework. The parameter settings of the core optimizers in ACF embedded with PSO ($\mathrm{ACF_{PSO}}$) and DE ($\mathrm{ACF_{DE}}$) are extracted from the sensitivity analysis provided in [10]. For ACF, according to the reported sensitivity analysis in Section IV-E $\tilde{r} = 0.3$, $\tilde{R} = 0.8$, $\tilde{e} = 0.3$, $\tilde{E} = 1.0$. Moreover, according to [10], a' is set to 0.05 for deactivating sub-populations. In addition, $N_{\max}$ is set to 30 according to the empirical study provided by Blackwell et al. [14]. Summary of the parameter settings of $\mathrm{ACF_{PSO}}$ and $\mathrm{ACF_{DE}}$ are shown in Table III. For parameters of the comparison algorithms, default values suggested in their papers are used since our investigations also demonstrate that these algorithms show their best performance with those suggested settings.

E. Sensitivity analysis

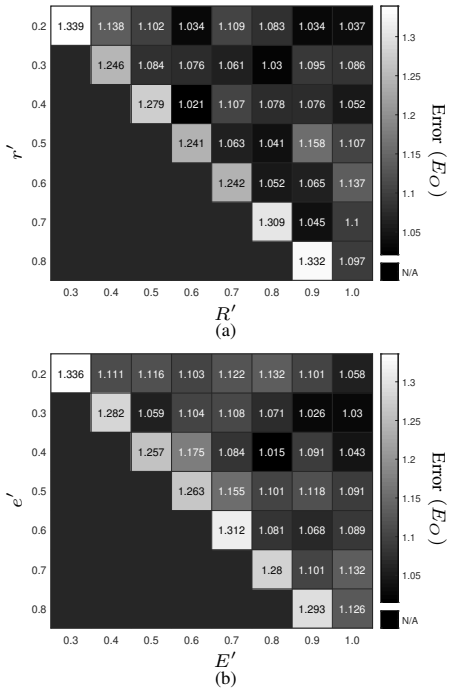
For sensitivity analysis, we use PSO as the core optimizer of ACF. The experiments in this part cover investigations on multi-population controlling parameters i.e. r', R', e' and

TABLE II
PARAMETER SETTINGS OF THE BENCHMARK GENERATORS. THE DEFAULT VALUES ARE HIGHLIGHTED.

Parameter	Symbol	Value
Number of peaks	m	10,25,50,100,200
Dimension	d	2,5,10
Evaluations between changes	ϑ	500, 1000, 2500, 5000
Shift severity	s	1,2,5
Height severity	$\tilde{h}$	7
Width severity	$\tilde{w}$	1
Peaks location range	SR	[-50,50]
Peak function/shape	–	f_1 (S-1), f_2 (S-2) and f_3 (S-3)
Peak height range	–	[30,70]
Peak width range	–	[1,12]
Initial height value	–	50
Number of environments	$\hat{t}$	100

TABLE III
PARAMETER SETTINGS OF $\mathrm{ACF_{PSO}}$ AND $\mathrm{ACF_{DE}}$

Method	Parameter	Value	Reference
PSO	χ	0.729843788	[43]
	C_1, C_2	2.05	[43]
	Neighbourhood topology	global star	[43]
	Population Size	5	[10]
DE	F	0.5	[31]
	Cr	0.6	[31]
	strategy	$DE/best/2/bin$	[20]
	Population Size	6	[10]
ACF	r in (7)	0.3	Fig. 4
	R in (6)	0.8	Fig. 4
	e in (9)	0.3	Fig. 4
	E in (8)	1.0	Fig. 4
	Deactivation threshold a'	0.05	[10]
	Max sub-population # ($N_{\max}$)	30	[14]



(a) (b)

Fig. 4. *(a)* Sensitivity analysis on different values of r' and R' in (6) and (7) where $e' = 0.3$ and $E' = 1.0$, for optimizing the benchmark problems with the default parameter settings from Table II. *(b)* Sensitivity analysis on different values of e' and E' in (8) and (9) where $r' = 0.3$ and $R' = 0.8$.

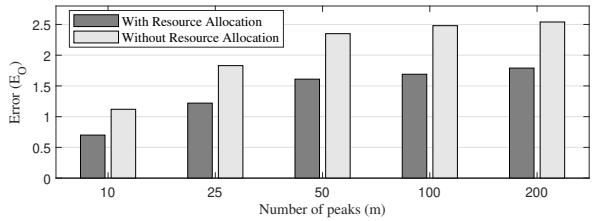


Fig. 5. Effect of using resource allocation mechanism on the performance of the proposed framework on the benchmark problems with different number of peaks (m) and the default parameter settings for the rest of parameters.

E'. In addition, the impact of resource allocation on the performance of the proposed framework is investigated. All reported results in this part are based on 31 independent runs and the figures are constructed based on their average values. Pairwise Wilcoxon signed-rank tests with $\alpha = 0.05$ are used for statistical analysis.

To perform the sensitivity analysis on r', R', e' and E', we disable the resource allocation mechanism to focus on the efficiency of the proposed multi-population controlling approach. Alternatively, the traditional Round Robin method is used for resource allocation between sub-populations. According to the sensitivity analysis that has been done on these parameters, a good combination of their values is reported in Table III. Fig. 4(a) shows the obtained E_O values by the framework with different combinations of r' and R' where $e' = 0.3$ and $E' = 1$. As can be seen, the performance of the framework is not highly sensitive to the values of r' and R', and there are several combinations of these parameters whose results are not significantly different. Fig. 4(b) shows the results of the framework with different combinations of e' and E' where $r' = 0.3$ and $R' = 1.0$. Similar to the reported results in Fig. 4(a), the performance is not very sensitive to the values of e' and E' and there are several combinations with not significantly different performance.

One important observation from Fig. 4(b) is that when the values of e' and E' are very close, i.e. $E' - e' = 0.1$, the obtained results are worse that other combinations where the differences between e' and E' are larger. In fact, in such circumstances, the proposed exclusion mechanism acts similar to the traditional exclusion mechanisms with a single (layer) threshold. This observation indicates the superiority of the double-layer exclusion.

After determining parameter settings for r', R', e' and E', we add resource allocation mechanism to the framework to investigate its impact. Fig. 5 compares the performance of the framework with and without resource allocation mechanism on the problem instances with different number of peaks and default settings for the rest of parameters. As shown in Fig. 5, resource allocation mechanism improves the results significantly.

F. Comparison With other methods

In this section, we compare the performance of $\mathrm{ACF}_{\mathrm{PSO}}$ and $\mathrm{ACF}_{\mathrm{DE}}$ with those of the algorithms from Table I. The experiments of this part are done on several problem instances with various characteristics such as different peak shapes, peak numbers, shift severities, dimensions, and change frequencies. All experiments are done 31 times with different random seed numbers, and their average results and standard errors based on E_O from Section IV-B are reported. Moreover, the results obtained by each algorithm in each problem instance are compared to those of all peer algorithms based on Wilcoxon signed-rank tests with $\alpha = 0.05$. Furthermore, the numbers of wins, ties and losses (*w/t/l*) are reported for each entry based on the statistical analysis. The best result is highlighted in each row and if there are more than one highlighted, it means that they are not significantly different based on the statistical analysis. All results are reported in Tables S-I, S-II, S-III, and S-IV in the supplementary document. Summaries of the reported results in the aforementioned tables, including the overall number of $win - loss$ of all algorithms, are reported in Tables IV, V, VI, and VII of this section.

According to the reported results in Tables S-I, S-II, S-III, and S-IV of the supplementary document, the landscapes generated by (S-1) (f_1) are more challenging to optimize in comparison to those constructed by (S-2) and (S-3). Indeed, peaks generated by f_1 are very narrow (based on the width (w) values) which makes them harder to explore. Additionally, since they are very narrow, algorithms experience a considerable fitness drop after environmental changes in comparison to f_2 and f_3. Moreover, results show that the landscapes generated by f_3 are easier to optimize in comparison to the ones built by conical peaks (f_2). One important reason is the gradient of the regions around the peak summits in Gaussian peaks (f_3) which results in less fitness drops after relocating peak summits. The illustrated current error plots in Fig. S-1 of the supplementary document demonstrate the above mentioned statements regarding the hardnesses and fitness drops of the landscapes generated by f_1, f_2 and f_3.

Table IV summarises results obtained by the algorithms on problem instances with different number of peaks and peak shapes. According to this table, $\mathrm{ACF}_{\mathrm{PSO}}$ and $\mathrm{ACF}_{\mathrm{DE}}$ are ranked first and second among all algorithms, respectively. The superiority of $\mathrm{ACF}_{\mathrm{PSO}}$ against $\mathrm{ACF}_{\mathrm{DE}}$ is a result of the higher convergence speed of PSO. In fact, PSO is the most suitable and dominating core optimizer in the multi-population DOP algorithms [10], [15], [23], [48]. According to Table IV, problem instances with higher number of peaks are often more challenging for the algorithms. This statement is more obvious for the adaptive multi-population methods that create larger numbers of sub-populations to adapt to the larger numbers of peaks. Therefore, this circumstance leads to a decrease in the amount of allocated computational resources to each sub-population. As can be seen, $\mathrm{ACF}_{\mathrm{PSO}}$ and $\mathrm{ACF}_{\mathrm{DE}}$ could maintain their efficiencies on the problem instances with larger numbers of peaks. This observation is more evident on the problem instances generated by f_1 whose peaks are highly narrow (see Table S-I of the supplementary document). In fact, this characteristic of peaks generated by f_1, significantly decreases the possibility of covering smaller peaks by the larger ones. Consequently, search spaces generated by f_1 consist of larger numbers of visible peaks in comparison to those of generated by f_2 and f_3. There are three main reasons behind the less deterioration in the performance of $\mathrm{ACF}_{\mathrm{PSO}}$

and $\mathrm{ACF}_{\mathrm{DE}}$ in problem instances with larger numbers of peaks:

- The adaptive double-layer exclusion mechanism keeps trackers on the close peaks, and increases the capability of locating and covering peaks that are close to the covered peaks. The role of the proposed exclusion mechanism becomes more highlighted where the number of peaks is higher which leads to a larger number of close peaks in the landscape due to the higher peak density.
- The resource allocation mechanism whose importance grows when the framework creates more sub-populations. Such circumstances lead to boost the shortage of the computational resources in each environment.
- The sub-population generation method prevents the overcrowding situation, such that, the framework produces a controllable and suitable number of sub-populations according to the limited computational resources.

Table S-II compares the results obtained by the algorithms on problem instances with different peak shapes and shift severities. According to the reported results in this table, problem instances with larger shift severities are more challenging for the algorithms since the tracking becomes more computational resource and time consuming. Additionally, the fitness drops after environmental changes are higher due to the larger peak relocations. Table V summarizes the reported results in Table S-II. As can be seen in this table, $\mathrm{ACF}_{\mathrm{PSO}}$ and $\mathrm{ACF}_{\mathrm{DE}}$ outperform other methods in problems with larger shift severities. One important reason behind this superiority is the utilized resource allocation mechanism in ACF in which superior sub-populations receive computational resources faster. Consequently, the best found position improves quickly in each environment. In addition, by considering sub-populations residing on the inferior peaks, especially after deactivating superior sub-populations, the framework maintains an appropriate level of tracking across all the discovered peaks.

Table S-III reports the results obtained by the algorithms on problem instances with different peak shapes and dimensions. According to the results from this table, by increasing the problem dimensionality, the performance of the algorithms deteriorates. In the 2-dimensional problem instances, the results obtained by $\mathrm{ACF}_{\mathrm{PSO}}$ and $\mathrm{ACF}_{\mathrm{DE}}$ are not significantly different and they share the first rank. The reason behind this performance equality is that the embedded DE can compete with PSO in lower dimensions. However, in higher dimensions, $\mathrm{ACF}_{\mathrm{PSO}}$ outperforms $\mathrm{ACF}_{\mathrm{DE}}$ due to the superiority of its core optimizer. In addition, in 10-dimensional problem instances generated by f_2 and f_3, $\mathrm{ACF}_{\mathrm{DE}}$ cannot outperform two PSO based algorithms, i.e. AmQSO and mPSO, since the superiority of PSO to DE outweighs the efficiency of the proposed framework. The summary of the reported results in Table S-II is shown in Table V. Similar to the previous experiments, $\mathrm{ACF}_{\mathrm{PSO}}$ and $\mathrm{ACF}_{\mathrm{DE}}$ ranked first and second overall, respectively.

Finally, Table VII shows the summary of results obtained by the algorithms on problem instances with different change frequencies and peak shapes. According to this table, $\mathrm{ACF}_{\mathrm{PSO}}$ shows the overall best performance. However, $\mathrm{ACF}_{\mathrm{DE}}$ is not ranked second in overall in this table. Note that, although $\mathrm{ACF}_{\mathrm{DE}}$ is not ranked second in this set of experiments, its overall performance is still acceptable, ranking first among DE based algorithms. Table VII shows the detailed results whose summaries are shown in Table VII. According to this table, this rank degradation is a consequence of the performance of $\mathrm{ACF}_{\mathrm{DE}}$ on f_3 based problem instances with higher change frequencies. Indeed, according to the experiments, PSO is performing very well on peaks generated by f_3. Therefore, the high convergence speed of some PSO based sub-populations dominates the effectiveness of the proposed framework in $\mathrm{ACF}_{\mathrm{DE}}$.

V. Conclusion

In this paper, we have presented a new adaptive control framework (ACF) for dynamic optimization problems (DOPs). ACF benefits from an adaptive sub-population generation approach whose sub-populations were grouped into three types of explorers, exploiters, and trackers. The proposed sub-population generation approach avoids generating too many sub-populations in the problems with large numbers of peaks. In addition,we have designed an adaptive double-layer exclusion mechanism to improve the capability of the framework in covering close peaks. Finally, an adaptive resource allocation mechanism has been proposed to control the consumption of computational resources by the sub-populations according to their types, task achievements, and ranks. Two instantiated algorithms with the proposed framework and a set of peer algorithms have been used to optimize a set of moving multiple peaks problem instances with different peak shapes, number of peaks, shift severities, dimensions, and change frequencies. The experimental results have demonstrated the superiority of the ACF based algorithms in most test cases. In this paper, we have focused on DOPs whose environmental changes are informed to the algorithms. Nevertheless, dealing with the DOPs with *hard to detect environmental changes* [35] where algorithms are not informed about the environmental changes, is an important topic for future work.

References

[1] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, University of Birmingham, 2011.

[2] S. Yang, "Genetic algorithms with memory-and elitism-based immigrants in dynamic environments," *Evolutionary Computation*, vol. 16, no. 3, pp. 385–416, 2008.

[3] M. Mavrovouniotis, F. M. Muller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743–1756, 2017.

[4] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

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

[6] S. Jiang and S. Yang, "Evolutionary dynamic multiobjective optimization: Benchmarks and algorithm comparisons," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 198–211, 2017.

[7] J. Ding, C. Yang, Q. Xiao, T. Chai, and Y. Jin, "Dynamic evolutionary multiobjective optimization for raw ore allocation in mineral processing," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 1, pp. 36–48, 2018.

TABLE IV
SUMMARY OF THE RESULTS (WIN-LOSS) OBTAINED BY THE ALGORITHMS ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND NUMBER OF PEAKS ($m = \{10, 25, 50, 100, 200\}$). THE CORRESPONDING RESULTS CAN BE FOUND IN TABLE S-I OF THE SUPPLEMENTARY DOCUMENT.

Algorithms	$\mathrm{ACF}_{\mathrm{PSO}}$	$\mathrm{ACF}_{\mathrm{DE}}$	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
win-loss	156	111	76	−83	−62	−10	59	0	−125	67	−26	−165

TABLE V
SUMMARY OF THE RESULTS (WIN-LOSS) OBTAINED BY THE ALGORITHMS ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND SHIFT SEVERITIES ($\tilde{s} = \{1, 2, 5\}$). THE CORRESPONDING RESULTS CAN BE FOUND IN TABLE S-II OF THE SUPPLEMENTARY DOCUMENT.

Algorithms	$\mathrm{ACF}_{\mathrm{PSO}}$	$\mathrm{ACF}_{\mathrm{DE}}$	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
win-loss	98	67	32	−60	−65	39	36	16	−57	32	-35	−96

TABLE VI
SUMMARY OF THE RESULTS (WIN-LOSS) OBTAINED BY THE ALGORITHMS ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND DIMENSIONS ($D = \{2, 5, 10, 20\}$). THE CORRESPONDING RESULTS CAN BE FOUND IN TABLE S-III OF THE SUPPLEMENTARY DOCUMENT.

Algorithms	$\mathrm{ACF}_{\mathrm{PSO}}$	$\mathrm{ACF}_{\mathrm{DE}}$	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
win-loss	121	91	72	−74	−83	−21	22	−3	−40	68	−31	−126

TABLE VII
SUMMARY OF THE RESULTS (WIN-LOSS) OBTAINED BY THE ALGORITHMS ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND CHANGE FREQUENCIES ($\vartheta = \{500, 1000, 2500, 5000\}$). THE CORRESPONDING RESULTS CAN BE FOUND IN TABLE S-IV OF THE SUPPLEMENTARY DOCUMENT.

Algorithms	$\mathrm{ACF}_{\mathrm{PSO}}$	$\mathrm{ACF}_{\mathrm{DE}}$	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
win-loss	115	54	66	−97	−46	41	57	1	−89	69	−35	−132

[8] T. T. Nguyen and X. Yao, "Continuous dynamic constrained optimization—the challenges," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 769–786, 2012.

[9] C. Bu, W. Luo, and L. Yue, "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 14–33, 2017.

[10] D. Yazdani, "Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost," Ph.D. dissertation, Liverpool John Moores University, Liverpool, UK, 2018.

[11] 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, pp. 1–24, 2012.

[12] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 440–458, 2006.

[13] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 06, pp. 959–974, 2010.

[14] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," in *Swarm Intelligence: Introduction and Applications*, C. Blum and D. Merkle, Eds. Springer Lecture Notes in Computer Science, 2008, pp. 193–217.

[15] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multi-population framework for locating and tracking multiple optima," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 05, pp. 590–605, 2016.

[16] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing*, vol. 15, no. 7, pp. 1427–1448, 2011.

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

[18] 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. 04, pp. 2144–2158, 2013.

[19] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," 2019.

[20] M. C. du Plessis and A. P. Engelbrecht, "Using competitive population evaluation in a differential evolution algorithm for dynamic environments," *European Journal of Operational Research*, vol. 218, no. 1, pp. 7–20, 2012.

[21] M. C. du Plessis and A. P. Engelbrecht, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *Journal of global optimization*, vol. 55, no. 1, pp. 73–99, 2013.

[22] J. K. Kordestani, A. E. Ranginkaman, M. R. Meybodi, and P. Novoa-Hernandez, "A novel framework for improving multi-population algorithms for dynamic optimization problems: A scheduling approach," vol. 44, pp. 788–805, 2019.

[23] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[24] J. Branke, T. Kaussler, C. Smidt, and H. Schmeck, "A multi–population approach to dynamic optimization problems," in *Evolutionary Design and Manufacture*, 2000, pp. 299–307.

[25] A. Sepas-Moghaddam, A. Arabshahi, D. Yazdani, and M. M. Dehshibi, "A novel hybrid algorithm for optimization in multimodal dynamic environments," in *International Conference on Hybrid Intelligent Systems*. IEEE, 2012, pp. 143–148.

[26] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. R. Meybodi, and M. Akbarzadeh-Totonchi, "mNAFSA: A novel approach for optimization in dynamic environments with global changes," *Swarm and Evolutionary Computation*, vol. 18, pp. 38–53, 2014.

[27] D. Yazdani, A. Sepas-Moghaddam, A. Dehban, and N. Horta, "A novel approach for optimization in dynamic environments based on modified artificial fish swarm algorithm," *International Journal of Computational Intelligence and Applications*, vol. 15, no. 02, pp. 1 650 010–1 650 034, 2016.

[28] D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, "A multi-objective time-linkage approach for dynamic optimization problems with previous-solution displacement restriction," in *Applications of Evolutionary Computation*. Lecture Notes in Computer Science, 2018.

[29] C. Li and S. Yang, "Fast multi-swarm optimization for dynamic opti-

mization problems," in *International Conference on Natural Computation*, vol. 7. IEEE, 2008, pp. 624–628.
[30] P. Novoa-Hernández, D. A. Pelta, and C. C. Corona, *Improvement Strategies for Multi-swarm PSO in Dynamic Environments*. Springer Berlin Heidelberg, 2010, pp. 371–383.
[31] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," in *Congress on Evolutionary Computation*, vol. 3, 2005, pp. 2808–2815.
[32] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 04, pp. 440–458, 2006.
[33] W. Luo, B. Yang, C. Bu, and X. Lin, "A hybrid particle swarm optimization for high-dimensional dynamic optimization," in *Simulated Evolution and Learning*, Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin, Eds. Springer Lecture Notes in Computer Science, 2017, vol. 10593, pp. 981–993.
[34] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *Congress on Evolutionary Computation*, 2009, pp. 439–446.
[35] C. Li and S. Yang, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 556–577, 2012.
[36] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, vol. 04, 1995, pp. 1942–1948.
[37] C. Li, "An efficient benchmark generator for dynamic optimization problems," in *Bio-inspired Computing – Theories and Applications*, M. Gong, L. Pan, T. Song, and G. Zhang, Eds. Communications in Computer and Information Science, 2016, vol. 682, pp. 60–72.
[38] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 01, pp. 143–155, 2019.
[39] R. W. Morrison and K. A. D. Jong, "A test problem generator for non-stationary environments," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, vol. 3, 1999, pp. 2047–2053.
[40] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*. Springer-Natural Computing Series, 2004.
[41] J. J. Grefenstette, "Evolvability in dynamic fitness landscapes: a genetic algorithm approach," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, vol. 3, 1999, pp. 2031–2038.
[42] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. Springer Natural Computing Series, 2003, pp. 239–262.
[43] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Congress on Evolutionary Computation*, vol. 1, 2001, pp. 84–88.
[44] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.
[45] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1666–1670.
[46] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.
[47] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "Dynamic optimization using self-adaptive differential evolution," in *Congress on Evolutionary Computation*, 2009, pp. 415–422.
[48] D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, "A new multi-swarm particle swarm optimization for robust optimization over time," in *Applications of Evolutionary Computation*, G. Squillero and K. Sim, Eds. Springer Lecture Notes in Computer Science, 2017, vol. 10200, pp. 99–109.

Supplementary Document of 'Adaptive Control of Sub-Populations in Evolutionary Dynamic Optimization'

CONTENTS

S-I. COMPLEXITY ANALYSIS

Since any population-based optimization algorithm can be embedded in ACF, herein, we only investigate the extra computation burden of the ACF's mechanisms. The overall time complexity of ACF mechanisms in each iteration is defined according to the time complexity of the exclusion mechanism, which is the most complex mechanism. To perform the exclusion mechanism in each iteration, the distance between the best found position of each sub-population, which executes an internal iteration, with all other sub-populations are calculated. Consequently, the time complexity of exclusion mechanism in each iteration is $O(|N_{ac}| \cdot N \cdot D)$, and the best and worst cases are $O(N_{\max}^2 \cdot D)$ and $O(N \cdot D)$, respectively.

S-II. DETAILED INFORMATION OF THE UTILIZED BENCHMARK FUNCTIONS

The baseline functions of MPB [1], DF1 [2], [3] (which also is used as the Scenario 2 of MPB), and GPB [4] are formulated as follows:

$$f_1^{(t)}(\mathbf{x}) = \max_{i \in \{1,\dots,m\}} \left\{ \frac{h_i^{(t)}}{1 + w_i^{(t)} \sum_{j=1}^{d} (\mathbf{x}_j - \mathbf{c}_{i,j}^{(t)})^2} \right\}, \quad \text{(S-1)}$$

$$f_2^{(t)}(\mathbf{x}) = \max_{i \in \{1,\dots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \mathbf{x} - \mathbf{c}_i^{(t)} \right\| \right\}, \quad \text{(S-2)}$$

$$f_3^{(t)}(\mathbf{x}) = \max_{i \in \{1,\dots,m\}} \left\{ h_i^{(t)} \exp\left(-\frac{\left\| \mathbf{x} - \mathbf{c}_i^{(t)} \right\|^2}{2\left(w_i^{(t)}\right)^2} \right) \right\}, \quad \text{(S-3)}$$

where f_1, f_2 and f_3 are the baseline functions of MPB, DF1, and GPB, respectively, $\max\{\cdot\}$ determines the basin of attraction of peaks in the landscape, $\|\cdot\|$ calculates Euclidean distance, m is the number of peaks, $\mathbf{x}$ is a solution in the D-dimensional problem space, $h_i^{(t)}$, $w_i^{(t)}$, and $\mathbf{c}_i^{(t)}$ are the height, width, and the center of the ith peak in the tth environment, respectively. In this paper, we use the same set of dynamics for all used benchmarks. The width, height, and location of peaks in these benchmarks change over time using the following update rules:

$$h_i^{(t+1)} = h_i^{(t)} + \tilde{h}\,\mathcal{N}(0,1), \quad \text{(S-4)}$$

$$w_i^{(t+1)} = w_i^{(t)} + \tilde{w}\,\mathcal{N}(0,1), \quad \text{(S-5)}$$

$$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)} + \mathbf{v}_i, \quad \text{(S-6)}$$

where $\mathcal{N}(0,1)$ is a random number drawn from a Gaussian distribution with mean 0 and variance 1, $\tilde{h}$ is the height severity, $\tilde{w}$ is the width severity, and $\mathbf{v}_i$ is the vector of ith peak center movement which is calculated by:

$$\mathbf{v}_i = \tilde{s}\frac{\mathbf{u}}{\|\mathbf{u}\|}, \quad \text{(S-7)}$$

where $\mathbf{u}$ is a D-dimensional vector whose elements are randomly generated numbers with uniform distribution in $[-0.5, 0.5]$.

S-III. EXPERIMENTAL RESULTS

Presented tables in this section contain the full experimental results corresponding to Tables IV, V, VI, and VII of Section IV-F in the manuscript.

Fig. S-1 illustrates the current error plots of $\mathrm{ACF}_{\mathrm{PSO}}$ and mPSO as two competitive algorithms with the same core optimizer for 20 environments on the benchmark problems with different peak functions/shapes from (S-1), (S-2) and (S-3) and the default settings for the rest of the parameters.

S-IV. A NOTE ON CHANGE DETECTION

In many real-world DOPs, the occurrence of environmental changes is obvious, and algorithms are informed about them [5]. For example, the arrival of new orders, fault in a part of the system, change in temperature, change in the number of resources, and change in costs, are detected by sensors, operators, and agents, which inform the algorithms. Consequently, to solve many real-world DOPs, algorithms do not need to be equipped with a change detection mechanism. To solve DOPs in which algorithms are not informed about the environmental changes, change detection mechanisms must be used. The most commonly used change detection approaches in DOP algorithms are re-evaluation based methods [6]. In these approaches, DOP algorithms regularly re-evaluate some specific individuals or detectors. When the re-evaluated fitness values are different from the stored values, the change is detected. It is shown in [6] that the re-evaluation approaches are capable of performing *robust 100% detection* if a sufficient number of detectors are used.

To make the proposed framework capable of handling DOPs in which the algorithm needs to detect environmental changes, we add a simple re-evaluation change detection mechanism to the framework (ACF^*). In this mechanism, the best found

TABLE S-I

RESULTS OBTAINED BY THE ALGORITHMS (MEAN, STANDARD ERROR IN PARENTHESIS) ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND NUMBER OF PEAKS (m) WHERE THE REST OF THE BENCHMARK PARAMETERS ARE SET TO THE DEFAULT VALUES FROM TABLE II.

Fun.	m	Algorithms											
		ACF_{PSO}	ACF_{DE}	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
f_1	10	3.79(0.06)	4.23(0.08)	6.25(0.09)	19.29(0.20)	17.86(0.23)	6.77(0.10)	5.11(0.08)	6.56(0.07)	17.13(0.35)	6.22(0.11)	11.09(0.14)	28.82(0.58)
	25	5.61(0.07)	5.93(0.06)	10.72(0.12)	18.88(0.15)	18.40(0.16)	9.51(0.17)	8.30(0.11)	11.99(0.12)	20.01(0.49)	11.00(0.15)	12.10(0.19)	29.60(0.67)
	50	6.38(0.07)	6.37(0.08)	14.15(0.16)	18.31(0.12)	17.72(0.12)	10.80(0.15)	10.04(0.11)	15.62(0.17)	20.39(0.51)	14.27(0.16)	13.07(0.26)	32.32(0.52)
	100	6.38(0.08)	6.42(0.09)	16.11(0.15)	17.71(0.11)	18.43(0.13)	11.16(0.15)	11.06(0.10)	18.65(0.18)	19.47(0.38)	16.16(0.15)	12.87(0.19)	32.07(0.50)
	200	6.31(0.07)	6.68(0.12)	16.95(0.16)	17.62(0.10)	18.61(0.12)	11.30(0.13)	11.55(0.11)	19.96(0.19)	20.17(0.68)	16.99(0.10)	12.79(0.15)	31.42(0.48)
f_2	10	0.70(0.01)	0.98(0.04)	1.25(0.04)	4.19(0.15)	4.38(0.12)	1.32(0.04)	1.36(0.09)	1.42(0.04)	3.66(0.16)	1.25(0.03)	2.11(0.05)	13.93(0.55)
	25	1.22(0.03)	1.66(0.05)	2.05(0.04)	4.22(0.09)	4.04(0.07)	2.86(0.06)	2.00(0.07)	2.29(0.04)	4.65(0.13)	1.99(0.03)	3.03(0.10)	12.65(0.46)
	50	1.61(0.04)	2.06(0.08)	2.45(0.04)	3.88(0.07)	3.84(0.07)	3.54(0.11)	2.41(0.05)	2.67(0.04)	4.68(0.11)	2.48(0.03)	3.22(0.09)	11.51(0.32)
	100	1.69(0.04)	2.04(0.04)	2.64(0.04)	3.50(0.05)	3.41(0.03)	3.61(0.07)	2.49(0.05)	2.93(0.04)	4.57(0.12)	2.68(0.03)	3.12(0.06)	9.83(0.23)
	200	1.79(0.03)	2.13(0.05)	2.69(0.03)	3.20(0.05)	3.21(0.03)	3.50(0.07)	2.50(0.03)	3.17(0.05)	4.52(0.09)	2.71(0.03)	3.02(0.04)	9.35(0.11)
f_3	10	0.29(0.01)	0.50(0.03)	0.42(0.04)	1.99(0.18)	1.01(0.04)	0.43(0.04)	0.76(0.10)	0.53(0.04)	1.67(0.17)	0.41(0.03)	0.70(0.06)	13.38(0.55)
	25	0.57(0.02)	0.77(0.04)	0.66(0.02)	1.99(0.07)	1.05(0.03)	1.56(0.07)	1.01(0.05)	0.76(0.02)	2.61(0.11)	0.73(0.03)	1.51(0.05)	12.76(0.43)
	50	0.88(0.03)	1.02(0.04)	0.89(0.02)	2.08(0.05)	1.54(0.01)	2.14(0.09)	1.26(0.04)	0.99(0.02)	3.18(0.14)	0.88(0.20)	2.10(0.09)	12.04(0.36)
	100	1.05(0.02)	1.28(0.04)	1.12(0.03)	2.06(0.04)	1.91(0.02)	2.31(0.07)	1.48(0.03)	1.25(0.02)	3.42(0.12)	1.10(0.02)	2.36(0.08)	10.49(0.24)
	200	1.22(0.02)	1.42(0.04)	1.23(0.02)	2.26(0.04)	2.09(0.14)	2.46(0.06)	1.58(0.03)	1.38(0.02)	3.74(0.15)	1.23(0.02)	2.50(0.80)	10.49(0.24)

TABLE S-II

RESULTS OBTAINED BY THE ALGORITHMS (MEAN, STANDARD ERROR IN PARENTHESIS) ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND SHIFT SEVERITIES ($\tilde{s}$) WHERE THE REST OF THE BENCHMARK PARAMETERS ARE SET TO THE DEFAULT VALUES FROM TABLE II.

Fun.	$\tilde{s}$	Algorithms											
		ACF_{PSO}	ACF_{DE}	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
f_1	1	3.79(0.06)	4.23(0.08)	6.25(0.09)	19.29(0.20)	17.86(0.23)	6.77(0.10)	5.11(0.08)	6.56(0.07)	17.13(0.35)	6.22(0.11)	11.09(0.14)	28.82(0.58)
	2	6.09(0.09)	6.78(0.10)	10.21(0.12)	23.16(0.18)	17.82(0.17)	9.25(0.11)	7.63(0.09)	10.02(0.09)	27.03(0.05)	10.12(0.11)	18.59(0.20)	29.29(0.60)
	5	10.48(0.13)	10.43(0.13)	17.11(0.19)	24.77(0.15)	20.48(0.21)	14.31(0.14)	14.21(0.21)	16.25(0.16)	39.24(0.61)	17.25(0.17)	29.24(0.23)	28.95(0.48)
f_2	1	0.70(0.01)	0.98(0.04)	1.25(0.04)	4.19(0.15)	4.38(0.12)	1.32(0.04)	1.36(0.09)	1.42(0.04)	3.66(0.16)	1.25(0.03)	2.11(0.05)	13.93(0.55)
	2	1.16(0.02)	1.51(0.09)	2.00(0.03)	5.88(0.16)	8.36(0.24)	1.83(0.04)	1.79(0.11)	2.04(0.05)	5.20(0.18)	1.96(0.05)	3.26(0.08)	14.33(0.51)
	5	2.29(0.08)	2.80(0.10)	3.52(0.09)	7.52(0.21)	19.52(0.43)	3.25(0.07)	3.17(0.10)	3.97(0.09)	8.56(0.29)	3.61(0.08)	6.26(0.13)	14.43(0.52)
f_3	1	0.29(0.01)	0.50(0.03)	0.42(0.04)	1.99(0.18)	1.01(0.04)	0.43(0.04)	0.76(0.10)	0.53(0.04)	1.67(0.17)	0.41(0.03)	0.70(0.06)	13.38(0.55)
	2	0.43(0.01)	0.73(0.09)	0.71(0.06)	2.28(0.15)	2.70(0.12)	0.69(0.04)	0.89(0.70)	0.73(0.05)	2.23(0.18)	0.69(0.02)	1.08(0.07)	12.97(0.61)
	5	1.10(0.06)	1.42(0.11)	1.45(0.07)	4.02(0.15)	7.67(0.18)	1.43(0.05)	1.47(0.06)	1.74(0.08)	4.29(0.20)	1.38(0.04)	2.40(0.07)	13.64(0.57)

TABLE S-III

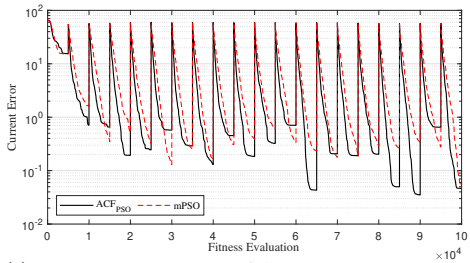
RESULTS OBTAINED BY THE ALGORITHMS (MEAN, STANDARD ERROR IN PARENTHESIS) ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND DIMENSIONS (D) WHERE THE REST OF THE BENCHMARK PARAMETERS ARE SET TO THE DEFAULT VALUES FROM TABLE II.

Fun.	D	Algorithms											
		ACF_{PSO}	ACF_{DE}	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
f_1	2	1.25(0.05)	1.29(0.04)	2.15(0.13)	4.64(0.09)	5.27(0.10)	4.93(0.18)	2.63(0.17)	2.18(0.14)	3.92(0.15)	2.16(0.12)	3.68(0.15)	13.20(0.34)
	5	3.79(0.06)	4.23(0.08)	6.25(0.09)	19.29(0.20)	17.86(0.23)	6.77(0.10)	5.11(0.08)	6.56(0.07)	17.13(0.35)	6.22(0.11)	11.09(0.14)	28.82(0.58)
	10	12.19(0.25)	14.85(0.26)	19.01(0.26)	31.44(0.42)	27.72(0.36)	22.04(0.28)	21.02(0.39)	23.08(0.42)	34.01(0.57)	18.94(0.23)	33.03(0.39)	36.42(0.57)
f_2	2	0.38(0.01)	0.40(0.01)	0.58(0.03)	1.06(0.3)	1.68(0.04)	1.04(0.04)	1.28(0.09)	0.75(0.05)	1.08(0.04)	0.61(0.03)	0.87(0.03)	2.98(0.12)
	5	0.70(0.01)	0.98(0.04)	1.25(0.04)	4.19(0.15)	4.38(0.12)	1.32(0.04)	1.36(0.09)	1.42(0.04)	3.66(0.16)	1.25(0.03)	2.11(0.05)	13.93(0.55)
	10	2.68(0.11)	2.97(0.13)	3.01(0.08)	7.41(0.25)	8.31(0.26)	5.19(0.13)	4.07(0.18)	4.44(0.14)	6.77(0.29)	3.02(0.06)	5.28(0.12)	19.25(0.75)
f_3	2	0.24(0.008)	0.23(0.008)	0.33(0.03)	0.44(0.03)	0.56(0.02)	0.42(0.02)	0.76(0.06)	0.42(0.02)	0.58(0.02)	0.36(0.02)	0.34(0.02)	1.90(0.10)
	5	0.29(0.01)	0.50(0.03)	0.42(0.04)	1.99(0.18)	1.01(0.04)	0.43(0.04)	0.76(0.10)	0.53(0.04)	1.67(0.17)	0.41(0.03)	0.70(0.06)	13.38(0.55)
	10	0.83(0.07)	0.90(0.08)	0.82(0.06)	3.16(0.18)	2.55(0.11)	1.67(0.12)	1.39(0.07)	1.42(0.05)	3.15(0.21)	0.84(0.05)	1.52(0.09)	16.57(0.71)

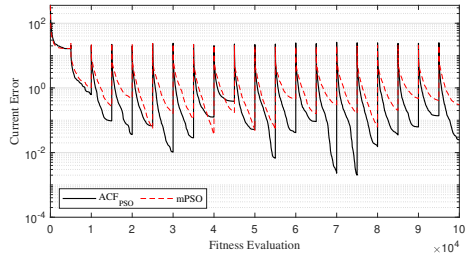
positions by sub-populations are re-evaluated in each iteration to detect environmental changes. We then examine the perfor-

TABLE S-IV
RESULTS OBTAINED BY THE ALGORITHMS (MEAN, STANDARD ERROR IN PARENTHESIS) ON THE PROBLEM INSTANCES WITH DIFFERENT PEAK SHAPES (f_1, f_2, f_3) AND CHANGE FREQUENCIES (ϑ) WHERE THE REST OF THE BENCHMARK PARAMETERS ARE SET TO THE DEFAULT VALUES FROM TABLE II.

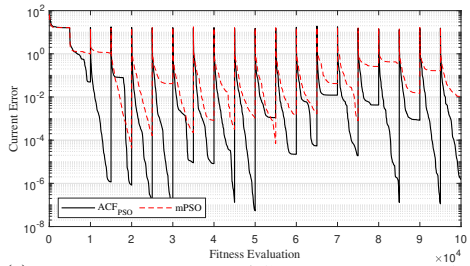
Fun.	ϑ	Algorithms											
		ACF_{PSO}	ACF_{DE}	AmQSO	CPSO	DynPopDE	DSPSO	mCMA-ES	mbDE	mjDE	mPSO	mQSO	RPSO
f_1	500	26.11(0.32)	26.96(0.62)	37.24(0.40)	54.90(0.33)	29.41(0.37)	37.26(0.43)	29.92(0.36)	38.18(0.36)	50.65(1.12)	37.10(0.45)	42.15(0.41)	63.69(0.21)
	1000	16.17(0.32)	16.83(0.51)	25.40(0.38)	41.25(0.36)	24.93(0.25)	25.30(0.30)	19.97(0.34)	25.81(0.33)	40.73(0.78)	24.91(0.27)	29.99(0.36)	58.39(0.49)
	2500	7.25(0.14)	7.38(0.17)	12.17(0.14)	26.41(0.35)	19.32(0.23)	12.91(0.19)	9.79(0.13)	13.00(0.18)	26.02(0.48)	11.93(0.18)	17.91(0.24)	40.59(0.55)
	5000	3.79(0.06)	4.23(0.08)	6.25(0.09)	19.29(0.20)	17.86(0.23)	6.77(0.10)	5.11(0.08)	6.56(0.07)	17.13(0.35)	6.22(0.11)	11.09(0.14)	28.82(0.58)
f_2	500	6.30(0.28)	6.86(0.20)	6.29(0.13)	13.13(0.35)	8.01(0.25)	6.76(0.14)	5.78(0.16)	7.59(0.20)	12.93(0.47)	6.32(0.14)	8.91(0.21)	23.56(0.61)
	1000	3.27(0.12)	5.02(0.18)	4.06(0.11)	9.62(0.30)	6.12(0.17)	4.25(0.08)	4.00(0.13)	4.97(0.11)	8.89(0.40)	4.10(0.10)	5.37(0.12)	19.09(0.53)
	2500	1.49(0.05)	2.20(0.25)	2.17(0.06)	5.93(0.19)	4.85(0.14)	2.14(0.06)	2.18(0.08)	2.49(0.07)	5.52(0.22)	2.14(0.06)	3.02(0.09)	15.99(0.52)
	5000	0.70(0.01)	0.98(0.04)	1.25(0.04)	4.19(0.15)	4.38(0.12)	1.32(0.04)	1.36(0.09)	1.42(0.04)	3.66(0.16)	1.25(0.03)	2.11(0.05)	13.93(0.55)
f_3	500	2.82(0.20)	3.46(0.24)	2.38(0.10)	6.02(0.21)	4.00(0.22)	2.26(0.06)	2.54(0.15)	3.39(0.14)	9.16(0.57)	2.24(0.09)	3.49(0.13)	19.04(0.80)
	1000	1.34(0.06)	1.70(0.38)	1.32(0.05)	3.59(0.16)	2.09(0.09)	1.48(0.06)	1.68(0.09)	1.72(0.06)	6.01(0.48)	1.30(0.05)	1.74(0.07)	16.62(0.59)
	2500	0.50(0.02)	0.95(0.11)	0.70(0.06)	2.20(0.12)	1.14(0.04)	0.68(0.04)	0.85(0.05)	0.83(0.05)	2.61(0.16)	0.68(0.04)	0.93(0.04)	15.06(0.62)
	5000	0.29(0.01)	0.50(0.03)	0.42(0.04)	1.99(0.18)	1.01(0.04)	0.43(0.04)	0.76(0.10)	0.53(0.04)	1.67(0.17)	0.41(0.03)	0.70(0.06)	13.38(0.55)



(a) Convergence behaviour of ACF_{PSO} and mPSO on f_1.



(b) Convergence behaviour of ACF_{PSO} and mPSO on f_2.



(c) Convergence behaviour of ACF_{PSO} and mPSO on f_3.

Fig. S-1. Current error of ACF_{PSO} and mPSO as two competitive algorithms with the same core optimizer for 20 environments on the benchmark problems with different peak functions/shapes from (S-1), (S-2) and (S-3) and the default settings for the rest of the parameters. All sub-figures are illustrated according to the average of the current errors for 31 independent runs.

mance of ACF* on the MPB with different number of peaks and report the obtained results in Table S-V. By comparing the reported results by $\text{ACF}^*_{\text{PSO}}$ and ACF_{PSO} on the standard

TABLE S-V
OBTAINED RESULTS (AVERAGE OFFLINE ERROR BY (17) AND STANDARD ERROR IN PARENTHESIS) BY $\text{ACF}^*_{\text{PSO}}$ ON THE GENERATED PROBLEMS BY THE MPB WITH DIFFERENT NUMBERS OF PEAKS m.

m	offline error
10	0.80(0.02)
25	1.42(0.05)
50	1.79(0.04)
100	1.92(0.04)
200	1.98(0.03)

MPB (f_2) in Tables S-V and S-I, it can be observed that the former results are inferior. The reason is that the $\text{ACF}^*_{\text{PSO}}$ uses the re-evaluation mechanism to detect changes, which consumes a considerable amount of fitness evaluations in each iteration.

REFERENCES

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

[2] R. W. Morrison and K. A. D. Jong, "A test problem generator for non-stationary environments," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, vol. 3, 1999, pp. 2047–2053.

[3] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*. Springer-Natural Computing Series, 2004.

[4] J. J. Grefenstette, "Evolvability in dynamic fitness landscapes: a genetic algorithm approach," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, vol. 3, 1999, pp. 2031–2038.

[5] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, University of Birmingham, 2011.

[6] H. Richter, "Detecting change in dynamic fitness landscapes," in *Congress on Evolutionary Computation*. IEEE, 2009, pp. 1613–1620.

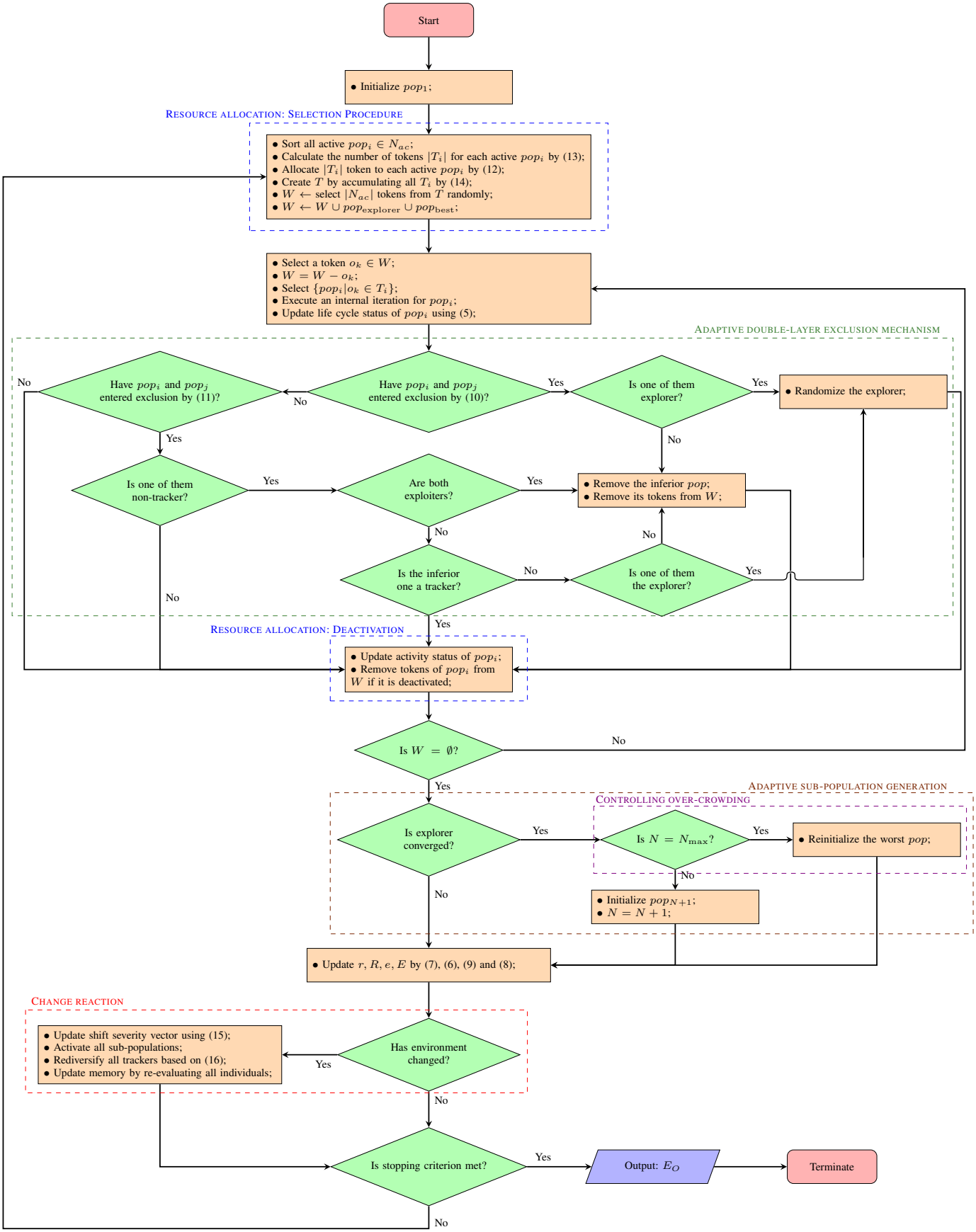


Fig. S-1. Flow chart of the proposed framework.