A Species-based Particle Swarm Optimization with Adaptive Population Size and Deactivation of Species for Dynamic Optimization Problems

14

DELARAM YAZDANI, Department of Computer Engineering, Mashhad Branch, Azad University, Iran
DANIAL YAZDANI, Faculty of Engineering & Information Technology, University of Technology Sydney, Australia
DONYA YAZDANI, AI Lab, British Antarctic Survey, United Kingdom
MOHAMMAD NABI OMIDVAR, School of Computing and Leeds University Business School, University of Leeds, United Kingdom
AMIR H. GANDOMI, Faculty of Engineering & Information Technology, University of Technology Sydney, Australia, and University Research and Innovation Center (EKIK), Obuda University, Hungary
XIN YAO, Research Institute of Trustworthy Autonomous Systems (RITAS), and Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, China and The Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, United Kingdom

Population clustering methods, which consider the position and fitness of individuals to form sub-populations in multi-population algorithms, have shown high efficiency in tracking the moving global optimum in dynamic optimization problems. However, most of these methods use a fixed population size, making them inflexible and inefficient when the number of promising regions is unknown. The lack of a functional relationship between the population size and the number of promising regions significantly degrades

Delaram Yazdani and Danial Yazdani contributed equally to this work.

This work was supported by the Research Institute of Trustworthy Autonomous Systems, the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), and Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531).

Authors' addresses: D. Yazdani, Department of Computer Engineering, Mashhad Branch, Azad University, Mashhad, Iran, 9187147578; email: delaram.yazdani@yahoo.com; D. Yazdani, Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo, Australia, 2007; email: danial.yazdani@gmail.com; D. Yazdani, AI Lab, British Antarctic Survey, Cambridge, United Kingdom, CB3 0ET; email: dny.yazdani@gmail.com; M. Nabi Omidvar, School of Computing and Leeds University Business School, University of Leeds, Leeds, United Kingdom, LS2 9JT; email: m.n.omidvar@leeds.ac.uk; A. H. Gandomi, Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo, Australia, 2007, and University Research and Innovation Center (EKIK), Obuda University, Budapest, Hungary, 1034; email: Gandomi@uts.edu.au; X. Yao (corresponding author), Research Institute of Trustworthy Autonomous Systems (RITAS), and Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, 518055, and The Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Birmingham, United Kingdom, B15 2TT; email: xiny@sustech.edu.cn.

2688-3007/2023/12-ART14
https://doi.org/10.1145/3604812

performance and limits an algorithm's agility to respond to dynamic changes. To address this issue, we propose a new species-based particle swarm optimization with adaptive population size and number of sub-populations for solving dynamic optimization problems. The proposed algorithm also benefits from a novel systematic adaptive deactivation component that, unlike the previous deactivation components, adapts the computational resource allocation to the sub-populations by considering various characteristics of both the problem and the sub-populations. We evaluate the performance of our proposed algorithm for the Generalized Moving Peaks Benchmark and compare the results with several peer approaches. The results indicate the superiority of the proposed method.

CCS Concepts: • **Computing methodologies** → **Continuous space search**; • **Theory of computation** → **Bio-inspired optimization;**

Additional Key Words and Phrases: Single-objective dynamic optimization problems, Evolutionary dynamic optimization, Tracking moving global optimum, Particle swarm optimization, Computational resource allocation

ACM Reference format:
Delaram Yazdani, Danial Yazdani, Donya Yazdani, Mohammad Nabi Omidvar, Amir H. Gandomi, and Xin Yao. 2023. A Species-based Particle Swarm Optimization with Adaptive Population Size and Deactivation of Species for Dynamic Optimization Problems. *ACM Trans. Evol. Learn.* 3, 4, Article 14 (December 2023), 25 pages.
https://doi.org/10.1145/3604812

1 INTRODUCTION

Many real-world optimization problems have dynamic search spaces that change over time. A single-objective continuous **dynamic optimization problem (DOP)**[1] can be defined as

$$\begin{aligned} &\text{Maximize}: \ f^{(t)}(\mathbf{x}) = f(\mathbf{x}, \alpha^{(t)}), \ \mathbf{x} = \{x_1, x_2, \ldots, x_d\} \\ &\text{Subject to}: \ \mathbf{x} \in \mathbb{X}: \ \mathbb{X} = \{\mathbf{x} \mid Lb_i \leq x_i \leq Ub_i\}, \ i \in \{1, 2, \ldots, d\}, \\ &\qquad\qquad g_j^{(t)}(\mathbf{x}) \leq 0, \ j \in \{1, 2, \ldots, \hat{g}\}, \\ &\qquad\qquad h_k^{(t)}(\mathbf{x}) = 0, \ k \in \{1, 2, \ldots, \hat{h}\}, \end{aligned} \tag{1}$$

where f is the objective function; $t \in [0, T]$ is the time index; $\mathbf{x}$ is a solution; $\mathbb{X}$ is the d-dimensional search space; Lb_i and Ub_i are lower bound and upper bound of search range in the ith dimension, respectively; α is a vector of time-dependent control parameters of the objective function; g_j is the jth inequality constraint; h_k is the kth equality constraint; and $\hat{g}$ and $\hat{h}$ are the numbers of inequality and equality constraints, respectively. In this article, we focus on box-constrained DOPs; i.e., no additional equality and inequality constraints are considered. The literature of DOP is mostly focused on problems whose environmental changes occur only in discrete time steps, i.e., $t \in \{1, \ldots, T\}$. For a DOP with T environmental states, there is a sequence of T stationary environments. Therefore, $f^{(t)}(\mathbf{x})$ with T environments can be reformulated as

$$\text{Maximize}: f^{(t)}(\mathbf{x}) = \left\{ f(\mathbf{x}, \alpha^{(k)}) \right\}_{k=1}^{T} = \left\{ f(\mathbf{x}, \alpha^{(1)}), f(\mathbf{x}, \alpha^{(2)}), \ldots, f(\mathbf{x}, \alpha^{(T)}) \right\}, \tag{2}$$

where it is usually assumed in the literature that there is a degree of similarity between successive environmental states, which is the case in many real-world DOPs [Branke 2012; Nguyen 2011].

Using evolutionary algorithms and swarm intelligence methods is an effective and popular way to tackle DOPs [Mavrovouniotis et al. 2017; Nguyen et al. 2012]. However, these methods are

[1] In this article, we focus on maximization problems.

originally designed for optimization in static environments and they cannot directly be used to tackle DOPs due to the challenges caused by the environmental changes. Some of these challenges are global and local diversity loss, outdated stored fitness values,[2] and limited available computational resources in each environment [Yazdani et al. 2022]. To cope with DOPs and address their challenges, evolutionary algorithms and swarm intelligence methods are usually combined with some other *components* to form **dynamic optimization algorithms (DOAs).**[3] DOAs are usually complex methods since they need to not only find the global optimum but also track it after environmental changes [Yazdani et al. 2021a].

One of the most efficient and commonly used DOA is the multi-population DOA [Nguyen et al. 2012], where more than one (sub)population is usually used in parallel. Some more advanced and the state-of-the-art multi-population DOAs are the ones whose sub-populations are capable of locating and tracking multiple moving promising regions simultaneously [Blackwell et al. 2008; Nguyen et al. 2016; Yazdani et al. 2022, 2013]. In such multi-population DOAs, a *population division and management component* is responsible to form sub-populations by clustering the individuals [Yazdani et al. 2021a]. Several clustering methods have been used to form sub-populations in multi-population DOAs; some of these methods cluster individuals based on their indices [Blackwell and Branke 2004], and some based on individuals' position and fitness [Luo et al. 2019; Parrott and Li 2006; Yang and Li 2010].

Clustering methods that take the position and fitness of individuals into account have shown high efficiency in tackling DOPs [Yazdani et al. 2021a]. However, their performance is highly sensitive to population size, which is typically fixed [Luo et al. 2018; Parrott and Li 2006; Wang et al. 2012; Yang and Li 2010; Zhang et al. 2019]. In other words, their performance is optimal when the population size is relatively matched to the number of promising regions in the DOPs' landscapes. As such, the performance of these methods is heavily reliant on the choice of population size, and a larger number of promising regions necessitates a larger population size and number of sub-populations.

Despite the considerable number of index-based clustering multi-population DOAs [Blackwell et al. 2008; Yazdani et al. 2022] with sub-population number adaptive to the number of the discovered promising regions, little attention has been given to the design of position- and fitness-based clustering multi-population methods with *adaptive* population size and sub-population number.

Effective management of computational resources between different sub-populations in multi-population DOAs is crucial, given the limited available resources and the different importance/priority of sub-populations [Yazdani et al. 2021a]. However, despite the extensive research in the field of DOPs, little attention has been paid to designing systematic computational resource allocation components that consider various problem characteristics such as temporal and spatial change severity, and sub-population features like their role, task accomplishments, and rank. Currently, most DOAs employ a simple Round-Robin approach to run sub-populations [Yazdani et al. 2021a] in each iteration [Blackwell and Branke 2004; Kordestani et al. 2019; Novoa et al. 2009] or deactivate converged sub-populations [Kamosi et al. 2010].

In the DOP literature, **particle swarm optimization (PSO)** [Kennedy and Eberhart 1995] is the most commonly used optimization method [Yazdani et al. 2021b], and most dynamic frameworks show their best performance when they use PSO as their core optimization component [Li and Yang 2012; Nguyen et al. 2016; Yazdani et al. 2018, 2020, 2021b, 2022].

To address the issues stated above, we propose a species-based PSO method, denoted as SPSO with **adaptive population (AP)** and **adaptive deactivation (AD)** components ($SPSO_{+AP+AD}$).

[2]Also called outdated memory in the literature.

[3]In this article, we use the term *DOA* to refer to both evolutionary and swarm intelligence dynamic optimization.

The proposed method forms sub-populations (species) using a clustering approach that takes into account the fitness and positions of individuals in each iteration. Unlike most existing species-based DOAs [Li et al. 2006; Luo et al. 2016; Parrott and Li 2006] that employ a fixed population size, our approach utilizes adaptive numbers of overall population size and the number of sub-populations, which the proposed AP component adjusts based on the number of discovered promising regions in the search space.

In addition, we propose the AD component that systematically and adaptively allocates the computational resources to the sub-populations. This resource allocation component activates and deactivates species adaptively. This is done by controlling and altering the values of the relevant parameters of the deactivation process based on the role and convergence status of each species, and also DOPs' characteristics including the number of discovered promising regions and shift severity. By using the proposed resource allocation component, the exploitation and exploration capabilities are significantly improved by systematic allocation of the limited available computational resources to the species in each environment.

The rest of this article is organized as follows. Section 2 covers the related work. The proposed method is described in Section 3. Section 4 explains the experiment setup and reports the experimental results and analyses. Finally, Section 5 concludes this article.

2 RELATED WORK

DOAs have been widely used for optimizing DOPs [Cruz et al. 2011; Mavrovouniotis et al. 2017; Nguyen and Yao 2012]. Several classes of DOPs have been investigated in the literature, such as multi-objective DOPs [Raquel and Yao 2013], single-objective DOPs [Yazdani et al. 2021a, b], and robust optimization over time [Fu et al. 2015; Yu et al. 2010]. Designing evolutionary algorithms for solving multi-objective DOPs is a popular sub-field of DOPs that has a rich literature. In multi-objective DOP works, different techniques have been used for tracking the moving **Pareto-optimal-set (PS)** over time, including:

Diversity control [Chen et al. 2017; Ma et al. 2021]. Some multi-objective DOAs use diversity control mechanisms after environmental changes [Deb et al. 2007]. In Ma et al. [2021], a number of solutions are systematically randomized in different regions of the objective space after environmental changes to increase the diversity. A bi-population algorithm is introduced in Chen et al. [2017], where one sub-population is responsible for maintaining diversity and the other one focuses on convergence.

Memory [Goh and Tan 2008; Peng et al. 2015]. The main purpose of these techniques is to enhance the optimization process in the current environment using historical information. In Goh and Tan [2008], randomly chosen individuals are added to the memory before each environmental change. The archived solutions are then used to update the non-dominated set in the future environments. In Peng et al. [2015], non-dominated solutions in each environment are archived into memory, which are used to replace the worst individuals after each environmental change.

Prediction [Jiang et al. 2017; Muruganantham et al. 2015; Xu et al. 2017]. In these algorithms, some prediction methods are used to estimate the PS/Pareto front after environmental changes when they exhibit some predictability characteristics. For example, a linear Kalman filter is used in Muruganantham et al. [2015] to predict the PS in the new environment. In Jiang et al. [2017], a domain adaptation approach is used to build a predictor that learns from populations in past environments. Differential prediction and Cauchy mutation are employed in Xu et al. [2017] for initializing the population after environmental changes.

Note that the mechanisms and components used for developing dynamic multi-objective optimization algorithms are different from those used in designing DOAs for tracking the moving global optimum in single-objective DOPs. Since this work focuses on single-objective DOPs, providing a comprehensive review of dynamic multi-objective algorithms is outside the scope of this article. Surveys on this topic that can be referred to include Azzouz et al. [2017], Jiang et al. [2022] and Raquel and Yao [2013].

Most algorithms developed for single-objective DOPs focus on **tracking the moving optimum (TMO)** [Yazdani et al. 2021a]. The majority of the state-of-the-art algorithms in this field use multiple sub-populations. In these algorithms, the main concerns are to address global and local diversity loss and effectively control the sub-populations over time [Yazdani et al. 2021a]. However, TMO is not practical for some real-world DOPs in which frequent change of the deployed solution is undesirable/impossible. Such DOPs are denoted as robust optimization over time whose main goal is to find solutions for deployment whose quality remains acceptable after several environmental changes [Jin et al. 2013]. In the rest of this section, we focus on reviewing related works in TMO for single-objective continuous DOP literature, which matches the focus of this article. For brevity, from now on, we use the terms *DOAs* and *DOPs* to refer to dynamic single-objective continuous algorithms and problems, respectively.

The state-of-the-art DOAs are usually complex algorithms constructed by assembling several components to address the challenges of DOPs including diversity loss and limited available computational resources in each environment [Yazdani et al. 2021a]. Two important components of DOAs are population division and management, and computational resource allocation. The former usually divides the population into sub-populations and manages them in order to provide a foundation for addressing the diversity loss issue and accelerating the optimization process after environmental changes. The latter, on the other hand, controls the consumption of computational resources (i.e., the number of fitness evaluations) by sub-populations. In this section, we review these two components.

2.1 Population Division and Management

Multi-population DOAs are both the most effective and flexible methods to tackle DOPs [Li et al. 2015; Yazdani et al. 2021a]. The population is divided into several sub-populations in such DOAs where the number of sub-populations is a parameter that can be set either adaptively or by the user [Yazdani et al. 2022]. In multi-population DOAs, different clustering approaches have been applied to form sub-populations [Yazdani et al. 2021a]. It is worth mentioning that dividing the population into sub-populations is also vastly used in speciation and niching methods in the field of multimodal optimization [Darwen and Yao 1996, 1997; Lu and Yao 2005]. In fact, many population clustering methods used in DOAs originally belonged to the field of multimodal optimization [Li et al. 2016; Luo et al. 2021; Parrott and Li 2006].

Generally, clustering approaches used in DOAs include index-based clustering, which clusters individuals based on their indices [Blackwell and Branke 2004]; position-based clustering, which clusters individuals based on their positions [Halder et al. 2011]; and position- and fitness-based clustering, which clusters individuals according to both their fitness and position [Liu et al. 2019; Luo et al. 2019, 2021; Parrott and Li 2006].

There is also a difference in the frequency of population clustering in different DOAs. Clustering can be performed frequently or only at specific points in time. In DOAs based on index clustering, the population is divided at the beginning of optimization or after a sub-population has been generated, and individual memberships in sub-populations are permanent throughout the optimization process [Blackwell and Branke 2006; Blackwell et al. 2008]. In contrast, DOAs that use clustering methods based on positions/fitness values may re-cluster individuals every

iteration [Parrott and Li 2006], or at specific points in time, such as after environmental changes [Yang and Li 2010].

A multi-population DOA may use fixed or variable population sizes and sub-population numbers, depending on the clustering approach and other mechanisms employed. For example, in Blackwell and Branke [2006], a fixed number of sub-populations with fixed individual memberships is used, resulting in a fixed overall population size throughout the optimization process. Similarly, in most DOAs that use fitness/position-based clustering methods, the population size is fixed [Luo et al. 2018; Zhang et al. 2019]. The most flexible multi-population DOAs use both varying population sizes and varying numbers of sub-populations, which are adapted to the number of promising regions discovered during optimization [Blackwell et al. 2008; Yazdani et al. 2022].

It should be noted that the concept of varying population size has been utilized in various subfields of evolutionary computation, including global optimization [Arabas et al. 1994; Coelho and de Oliveira 2008] and multi-objective optimization [Zhang and Mahfouf 2009] in static environments. In these works, the primary objectives of varying the population size are typically to tune the population size and regulate exploration and exploitation capabilities and diversity. In contrast, in the field of tracking the global optimum in single-objective DOPs, the purpose of adaptively changing the population size is to adjust the number of sub-populations to track multiple promising regions over time, whose number is unknown and variable. This approach was first introduced in Blackwell et al. [2008]. Nearly all current DOAs that adaptively change both the population size and the number of sub-populations employ index-based clustering methods [Yazdani et al. 2021a].

2.2 Computational Resource Allocation

Multi-population DOAs use different computational resource allocation policies for distributing computational resources between multiple sub-populations. The main commonly used computational resource allocation approaches are *Round Robin* (which allocates the computational resources equally to different sub-populations) [Blackwell and Branke 2006], methods based on *deactivating converged sub-populations* [Kamosi et al. 2010], and methods performing *performance-based sub-population selection* that allocate most of the computational resources to the sub-populations with higher performance [du Plessis and Engelbrecht 2013; Yang et al. 2017; Yazdani et al. 2022].

In DOAs with a deactivation-based component, to avoid wastage of the computational resources caused by performing unnecessary exploitation (i.e., over-exploitation), the sub-populations that are converged to a local optimum get deactivated. For almost all available methods used to identify the sub-populations to be deactivated, constant parameters are used (e.g., a predefined value for deactivation radius [Yazdani et al. 2013]) and the deactivated sub-populations will not become activated until the next environmental change.

The idea of deactivation has been proposed in Kamosi et al. [2010], which was originally inspired by the removal of converged sub-populations in Li and Yang [2009] to avoid *over-exploitation*. This idea has also been used in several DOAs [Amo et al. 2010; Novoa-Hernández et al. 2010]. In Yazdani et al. [2013], the best sub-population is not involved in the deactivation method since its output directly affects the performance of the DOA. Therefore, this sub-population continues to perform exploitation to improve the quality of the best found solution in each environment.

It is worth mentioning that the concept of computational resource allocation is also used in cooperative coevolutionary approaches in large-scale optimization [Omidvar et al. 2021a, b]. However, the approaches used in this field differ from those used in DOAs. In cooperative coevolutionary approaches, the systematic resource allocation methods usually take the contributions of subfunctions on the overall fitness value into account [Kazimipour et al. 2019; Yang et al. 2017]. However, in the systematic resource allocation components used in DOAs, in addition to considering dynamic

environments, several other factors including the role of sub-populations, their performance, and their convergence status are also taken into account [Yazdani et al. 2021a].

2.3 Motivation

As mentioned above, almost all DOAs whose population size and number of sub-populations adaptively vary over time are those that use index-based clustering. This is because controlling sub-populations and adjusting the population size are relatively straightforward and do not negatively impact other tasks of the algorithms such as tracking and coverage. Conversely, most species-based DOAs that divide the population into species/sub-populations using position- and fitness-based clustering approaches have a fixed population size. This is because using population control components to vary the population size can interfere with other parts of the algorithm, resulting in the deterioration of other tasks such as tracking multiple moving promising regions.

However, using a fixed population size and dividing individuals into species can be inflexible and deficient when the number of promising regions is unknown or changes over time, which is often the case in real-world DOPs. Therefore, population size must be tuned for different DOPs; otherwise, the algorithm may be deficient, as the number of species may be significantly fewer or greater than the number of promising regions in the landscape. Hence, species-based DOAs must adapt the number of species to the number of discovered promising regions and adjust the overall population size accordingly. To achieve this, a proper set of population control components must be used to avoid any undesirable interference with other parts of the algorithm and minimize negative impacts on other tasks, such as coverage and tracking of promising regions.

As described in Kamosi et al. [2010] and Yazdani et al. [2013], in order to manage the computational resource consumption by species, the *non-best tracker* species doing unnecessary exploitation (i.e., over-exploitation) should be stopped. In DOPs, tracker species are responsible for covering and tracking promising regions. To fulfill the tracking task, each tracker species needs to only get close enough to the summit of the covered promising region. However, further exploitation by tracker species will be useless and only wastes the computational resources. All existing processes of identifying the species to be deactivated are fixed/non-adaptive, and a deactivated species will not be activated until the next environmental change. Therefore, the relevant parameters to the deactivation process must be tuned for each DOP based on some characteristics, such as the number of promising regions (which is related to the number of species) and change frequency. Optimal tuning of these parameters can be impossible for DOPs whose characteristics change over time, e.g., the number of promising regions and the change frequency [Yazdani et al. 2022]. In addition, in the existing deactivation methods, only the convergence status of each species is considered, and the problem properties (e.g., the change frequency) and the status of other species are not taken into account. Therefore, the deactivation mechanism should be adaptive to the learned characteristics of the problems (e.g., the number of discovered promising regions) and all species' status.

The next section describes the proposed method that addresses the above-mentioned considerations.

3 ADAPTIVE SPECIES-BASED PSO

The proposed adaptive species-based PSO has the following novel components:

- A new species-based population division and management component that uses fitness- and position-based clustering to form species with adaptive population size and number of species.
- A new adaptive deactivation-based computational resource allocation component.

ALGORITHM 1: Pseudo code of the population clustering component [Qu et al. 2012] used in the proposed algorithm

Input: List of individuals $\mathcal{L}$ (including positions, indices, and fitness values) and species size n

Output: Species information (it includes all $\mathcal{S}_{j\in\{1,\ldots,N\}}$)

```
1 j = 0;
2 while L ≠ ∅ do
3     j = j + 1;
4     S_j = ∅;
5     seed_j ← the best individual ∈ L;
6     Calculate the Euclidean distances between seed_j and other individuals ∈ L;
7     S_j ← {seed_j ∪ the closest n − 1 individuals to seed_j}; // S_j is the jth species.
8     L = L \ S_j;
```

In the rest of this section, we explain the proposed method in more detail. We first propose an adaptive population division and management approach that adapts the number of species to the number of discovered promising regions. We then propose a systematic computational resource allocation component that adaptively controls the deactivation/activation process based on all species' current status and their task achievements. Finally, we describe the change reaction procedure.

3.1 Adaptive Species-based Population Division and Management Component

To divide the population into species, we adopt the clustering technique from Qu et al. [2012], which works based on the fitness and position of individuals. The species size is an input parameter of this clustering method, and the number of individuals in each species is fixed and equal among all species. This method does not need any additional problem-dependent parameters, such as the species radius in Parrott and Li [2006]. This clustering method works as follows.

First, all individuals form a list called $\mathcal{L}$. Then, the following steps are repeated until $\mathcal{L}$ becomes empty:

(1) The best individual in $\mathcal{L}$ is chosen as a species *seed*.
(2) The Euclidean distance between the best individual and all other individuals in $\mathcal{L}$ is calculated.
(3) The best individual and its $n-1$ closest individuals form a species.
(4) The species members are removed from $\mathcal{L}$.

n is the species size (i.e., the number of individuals in each species). The clustering process is performed at every iteration and updates the species according to the individuals' positions and fitness values. Algorithm 1 shows the pseudo code of the population clustering component. It is worth mentioning that the Euclidean distances calculated in the clustering process of each iteration will be (re-)used for determining the spatial size of species (for convergence detection purposes) in that iteration. Each species seed is used as the local attractor position (i.e., Gbest in PSO) by its species members.

In the proposed algorithm, species are classified into *tracker* and *non-tracker* species based on their spatial size. A Tracker is a species that has converged to a promising region and is tracking its optimum (i.e., the summit). Since the trackers are those which have converged, their diversity is relatively small. Herein, we use the *spatial size* $\mathfrak{s}$ of a species to evaluate its convergence status, which is calculated by

$$\mathfrak{s}_i = \max_{\mathcal{I}_j \in \mathcal{S}_i} \|\mathbf{g}_i^* - \mathbf{p}_j\|, \tag{3}$$

where $\mathcal{S}_i$ is the ith species, $\mathfrak{s}_i$ is the spatial size of the ith species, $\mathbf{g}_i^*$ is the personal best position of the seed in the ith species, $\mathcal{I}_j$ is the jth individual of the ith species, and $\mathbf{p}_j$ is the personal best position of $\mathcal{I}_j$.

In the beginning, species usually have high diversities and are referred to as non-tracker species. Non-tracker species are responsible for exploration and discovering promising regions. Once a non-tracker species has converged to a promising region, it becomes a tracker species. To determine whether a species is a tracker or non-tracker, we compare its spatial size with a predefined threshold r_{track}. The ith species is a tracker if $\mathfrak{s}_i \leq r_{\text{track}}$; otherwise, it is a non-tracker.

A tracker species is responsible for tracking the optimum of its covered promising region. Each tracker species must address the local diversity loss issue to maintain its exploitation capability at an appropriate level. An effective way to address this issue is to increase the local diversity of a species after each environmental change based on the optimum's estimated shift severity [Yazdani et al. 2018]. Therefore, r_{track} must always be greater than the increased spatial size of a species to address the local diversity loss. Consequently, a tracker species will not become a non-tracker by increasing its local diversity after each environmental change. This is important in the proposed algorithm since tracker species provide some important information to estimate the number of discovered promising regions and shift severity. In the proposed method, the value of r_{track} is defined adaptively based on the estimated shift severity over time.

As described in Section 2.3, the population size and the number of species should be adapted to the number of discovered promising regions. On the one hand, when the number of individuals/species is more than enough to track the discovered promising regions and search for possible undiscovered ones, redundant individuals/species must be removed. On the other hand, new individuals/species need to be born when all species have converged, or when the possibility of existing uncovered promising regions is high.

To increase the population size, we inject $\mathfrak{m}$ randomly initialized individuals into the population when all existing species have converged. A species is considered converged to a promising region if its spatial size is smaller than a threshold r_{generate}. When all species have converged, the algorithm loses its exploration capability and cannot discover any possible uncovered promising region. In addition, due to the converged species, the fitness and position of all members in each species are relatively close to each other; therefore, the species membership is unlikely to change by the clustering component. Injecting new random individuals will result in increasing global diversity and exploration capability of the algorithm. After generating $\mathfrak{m}$ new individuals, the clustering process is performed on all individuals (old and new) to update species. Updating the membership of species may result in migration of some individuals from a promising region to another, which further increases the global diversity.

In addition, similar to the majority of multi-population DOAs, we need to use an anti-*overcrowding* component. Overcrowding usually happens where more than one species converges to a promising region. In such a circumstance, depending on the population clustering component used, the promising region would be covered either by more than one species or by a too large species (when the converged species merge). Such additional species or individuals are *redundant* and waste the computational resources and deteriorate global diversity. Considering the population clustering approach used in the proposed algorithm in this article, overcrowding would happen when more than one species converges to a promising region. To address this issue, we use the standard *exclusion* mechanism [Blackwell and Branke 2006] for removing redundant species. The exclusion component considers two species to have converged to a promising region if the Euclidean distance between their seeds is less than r_{excl}. In such a circumstance, individuals of the species with inferior seed are removed.

Using the above-mentioned components for managing the population and controlling species, the proposed algorithm is capable of tracking multiple moving promising regions and adapting the number of species to the number of discovered promising regions. When additional species are needed to cover possible undiscovered promising regions, the proposed method increases the population size by injecting $\mathfrak{m}$ new individuals into the population, which in turn increases the number of species. Furthermore, the number of species is reduced when there are redundant species. In fact, it is expected that the redundant species will eventually converge to a covered promising region and be removed by the exclusion component. Thus, the increase and decrease of the population size are done by injecting $\mathfrak{m}$ new individuals and an exclusion component, respectively. The applied adaptive population-increasing/decreasing mechanisms enable the proposed algorithm to adapt its number of species to the number of discovered promising regions, which is beneficial in solving DOPs with an unknown number of promising regions and those whose number of promising regions changes over time.

A shortcoming of the proposed adaptive population generation method is that in solving DOPs with very large numbers of promising regions, this component may generate too many individuals/species. Such a circumstance results in shortage of computational resources for species. To address this issue, similar to Yazdani et al. [2022], the largest number of species N is bounded to a threshold $N_{\max}$. Using this approach, when $N = N_{\max}$, an anti-convergence mechanism [Blackwell and Branke 2006] will be activated. Therefore, when the number of species N is equal to $N_{\max}$ and all of them have converged, instead of generating $\mathfrak{m}$ new individuals, individuals of the species with the worst best found position are randomly re-initialized (i.e., their positions are randomized and their personal best positions are reset to their new positions). In fact, in a circumstance where the number of species has reached $N_{\max}$, by sacrificing the coverage of the promising region with the lowest fitness, the algorithm increases global diversity, which in turn results in maintaining the exploration capability.

3.2 Computational Resource Allocation Component

In this subsection, we propose a systematic computational resource allocation component that controls the activation/deactivation process of species. At the beginning of each environment, all species are active, and they run during each iteration. Each tracker species continues exploitation until it gets close enough to the summit of its covered promising region. A tracker species is considered close to the summit if its spatial size is less than a deactivation radius r_{a}. The tracker species i will be deactivated if $\mathfrak{s}_i \leq r_{\mathrm{a}}$. It is important to note that deactivated species hibernate and do not run during iterations; i.e., they do not consume computational resources. Deactivating such tracker species allows the algorithm to save some computational resources that can be used for (1) the non-tracker species to converge to a promising region, (2) tracker species that are still trying to get close to the summit of their covered promising region to fulfill the tracking task, and (3) the best tracker species that is performing exploitation around the best found position in the current environment.

The value of r_{a} plays a major role in all deactivation-based computational resource allocation components, including the proposed one, as the activity statuses of species are determined by comparing their spatial sizes to r_{a} [Yazdani et al. 2021a]. To the best of our knowledge, in all existing deactivation components, r_{a} has a fixed value over time. The optimal value of this parameter depends on different characteristics of DOPs, such as change frequency, and also the number of species. Consequently, using a fixed value for r_{a} results in the necessity of re-tuning the value of this parameter for various problems and circumstances in order to achieve the best performance. In DOPs where computational resources are limited in each environment, such as those with high change frequency, setting r_{a} to small values may render the deactivation component ineffective.

In such a circumstance, due to limited available computational resources in each environment, species cannot converge enough until their spatial sizes become smaller than r_a. Consequently, most species may not be deactivated in each environment. On the other hand, if there are adequate available computational resources in each environment, when r_a is set to large values, the resource allocation mechanism may hinder the tracking capability by early deactivation of species. Another consideration is that r_a cannot be tuned for most DOPs whose number of promising regions and/or change frequency change over time.

To address the above considerations regarding the r_a setting, we propose an adaptive method to adjust the values of r_a to the current status of the tracker species. At the beginning of each environment, r_a is set to a relatively large value r_a^{max}. Once the spatial size of all existing tracker species becomes less than r_a (i.e., they have been deactivated), r_a will be decreased. Then, all tracker species whose spatial size is larger than the new value of r_a will be activated again until their spatial size is constricted to smaller values than the updated r_a value. This procedure ensures that all tracker species can progress in their tracking tasks at each step. In cases where the available computational resources are very limited, the algorithm first tries to allocate only some computational resources to the tracker species, enough to perform a relatively acceptable exploitation/tracking. If the environment does not change, the algorithm tries to allocate additional computational resources to the tracker species by decreasing the value of r_a, which improves their exploitation/tracking capability.

In the proposed resource allocation approach, if a non-tracker species becomes a tracker (i.e., an uncovered promising region may have been discovered), the value of r_a will be reset to its initial value, i.e., r_a^{max}. By doing so, all currently deactivated tracker species will remain inactive since their spatial size is less than the initial value of r_a, and the new tracker species will benefit from more computational resources to perform exploitation. Note that active tracker species may become deactivated if their spatial size was larger than the previous r_a value but smaller than r_a^{max}. Once the new tracker's spatial size becomes smaller than the current r_a value, the proposed approach constricts the r_a value. Therefore, the algorithm prioritizes the new tracker species to ensure proper coverage of the newly discovered promising region.

The value of r_a decreases gradually from r_a^{max} until it reaches a lower bound value r_a^{min}. The value of r_a^{min} represents an appropriate level of exploitation for tracking a local moving optimum where further exploitation would be unnecessary. Therefore, to avoid over-exploitation, after r_a has reached its minimum value r_a^{min} in an environment, once the spatial size of a tracker species becomes less than r_a, it will not be activated until the next environment (i.e., until its spatial size is increased again to address the local diversity loss). Note that the tracker species with the best seed will always remain active since exploiting around the best found position has a significant impact on the algorithm's performance. Therefore, after all species trackers' spatial sizes become less than r_a^{min}, the remainder of computational resources will be used for (1) improving the exploitation around the best found position and (2) improving exploration by allocating computational resources to non-tracker species, which improves the capability of discovering uncovered promising regions.

The initial/maximum value of r_a, i.e., r_a^{max}, should be determined according to the estimated shift severity. The reason is that to address local diversity loss, the spatial size of species needs to be increased after each environmental change based on the estimated shift severity [Yazdani et al. 2021a]. Therefore, to control the activation status of the recently re-diversified tracker species at the beginning of each environment, the initial value of r_a, i.e., r_a^{max}, should be smaller than the estimated shift severity. We define $r_a^{max} = \rho \cdot \hat{s}$, where $\rho \in (0, 1)$ and $\hat{s}$ is the estimated shift severity. Furthermore, the value of r_a^{min} is set to $\mu \cdot \sqrt{d}$, where μ is a positive constant and d is the problem dimensionality. If the spatial size of all tracker species is smaller than r_a, then r_a is

updated by

$$r_{\mathrm{a}} = r_{\mathrm{a}}^{\min} + (r_{\mathrm{a}}^{\max} - r_{\mathrm{a}}^{\min}) \cdot \beta, \tag{4}$$

where β is a non-negative value parameter that is responsible for constricting r_{a}. At the beginning of each environment and when the value of r_{a} is reset to $r_{\mathrm{a}}^{\max}$, β is set to one. To constrict the value of r_{a}, we reduce the value of β by multiplying it by a constant $\gamma \in (0, 1)$, i.e., $\beta = \beta \cdot \gamma$. The value of β is reduced when r_{a} needs to be constricted, i.e., when the spatial sizes of all tracker species are smaller than the current r_{a} value. Using this procedure, the value of r_{a} will be $\in (r_{\mathrm{a}}^{\min}, r_{\mathrm{a}}^{\max})$.

3.3 Change Reaction Component

Our proposed algorithm is a reaction-based method [Nguyen et al. 2012] that needs to know the occurrence of the environmental changes in order to trigger some response components. Like many real-world DOPs with *visible* environmental changes [Branke and Schmeck 2003] where the algorithms are informed about the environmental changes by other parts of the system (e.g., sensors and agents), we assume that all tested algorithms in this article, including the proposed method, are informed about changes. It should be mentioned that the proposed algorithm can also use a simple reevaluation-based change detection component to detect environmental changes [Richter 2009; Yazdani et al. 2022].[4]

Once an environmental change happens, our proposed algorithm performs the following steps to cope with the new environment:

- First, the algorithm tries to estimate the shift severity. To this end, we use the Euclidean distances between the successive best found positions in all promising regions in the previous environment. Estimated shift severity in the tth environment $\hat{s}^{(t)}$ is obtained by averaging the Euclidean distances between the best found positions (seeds) by tracker species at the end of the $(t-1)$th and $(t-2)$th environments:

$$\hat{s}^{(t)} = \frac{1}{|\mathcal{T}|} \cdot \sum_{i \in \mathcal{T}} \left\| \mathbf{g}_i^{\star(t-1)} - \mathbf{g}_i^{\star(t-2)} \right\|, \tag{5}$$

  where $\mathcal{T}$ is the set of tracker species that were trackers at the end of both the $(t-1)$th and $(t-2)$th environments, and $\mathbf{g}_i^{\star(t-1)}$ and $\mathbf{g}_i^{\star(t-2)}$ are the best found positions by the ith tracker species at the *end* of the $(t-1)$th and $(t-2)$th environments, respectively.
- Second, to address the local diversity loss in tracker species i, we randomize all $m-1$ non-seed members, $\mathbf{x}_j$, around the seed position $\mathbf{g}_i^{\star(t-1)}$ by

$$\mathbf{x}_j = \mathbf{g}_i^{\star(t-1)} + \left(\frac{\mathfrak{n}}{\|\mathfrak{n}\|} \cdot \hat{s}^{(t)} \cdot r \right), \tag{6}$$

  where $\mathfrak{n}$ is a d-dimensional vector of random numbers drawn from Gaussian distribution $\mathcal{N}(0, 1)$, $\|\mathfrak{n}\|$ is the $L2-$norm of $\mathfrak{n}$, $\frac{\mathfrak{n}}{\|\mathfrak{n}\|}$ generates a random direction, and r is a random number generated with uniform distribution in $(0, 1)$. Using Equation (6), members are uniformly distributed inside a hyper-ball whose radius and center are $\hat{s}^{(t)}$ and $\mathbf{g}_i^{\star(t-1)}$, respectively. Therefore, the local diversity of species is increased based on the estimated shift severity. Note that there is a possibility that an individual is relocated to a position out of boundaries by Equation (6) (this may happen if a promising region's summit is extremely close to the

[4]The performance of the proposed method equipped with a change detection component for solving DOPs in which the changes need to be detected is investigated in Section S.3 of the supplementary document.

boundaries). To address this issue, the absorbing bound handing technique [Helwig et al. 2012] is used to push back the individual inside the search range.
- Finally, to address the outdated memory issue, all stored solutions are reevaluated.

3.4 Procedure of the Proposed Method

Algorithm 2 shows the workflow of the species-based PSO with the proposed adaptive population division and management and adaptive deactivation-based computational resource allocation components ($\text{SPSO}_{+\text{AP}+\text{AD}}$). This algorithm starts with a set of randomized individuals (line 1). At the beginning of each iteration, if all species had converged in the previous iteration (i.e., $f_{\text{generate}} = 1$ whose value is determined in lines 21 and 22), the global diversity is increased by randomizing/initializing $\mathfrak{m}$ individuals. If $N < N_{\max}$, $\mathfrak{m}$ new individuals are initialized (line 6), which also increases the number of species N in order to cover a higher numbers of promising regions. Otherwise, to avoid generating too many species while maintaining the global diversity (which increases the exploration capability), the individuals of the species with the worst $\mathbf{g}^\star$ are randomized (line 8). Then, in lines 10 and 11, the species are formed by clustering (by invoking Algorithm 1) the individuals. Note that the Euclidean distances calculated in the species formation process are used in the following procedures, such as determining spatial sizes and exclusion. Thereafter, the species will be classified into tracker and non-tracker ones (lines 14–20). The exclusion component is executed for all species in lines 23–25. Afterward, the computational resource allocation determines the species to be run in the current iteration. It first sets the deactivation radius parameter r_{a} in lines 26–30. Then, the resource allocation determines the activation/deactivation status of all species in lines 31–35. The PSO optimization process is performed for the active species in line 39. At the end of each iteration, the change reaction is done if an environmental change has happened (lines 40–50). The MATLAB (R2021a) source code of $\text{SPSO}_{+\text{AP}+\text{AD}}$ can be found in Yazdani [2023].

4 EXPERIMENTS AND ANALYSIS

In this section, we first explain the experimental design. We then investigate the effectiveness of the proposed population division and management component and the computational resource allocation component. Finally, we compare the performance of $\text{SPSO}_{+\text{AP}+\text{AD}}$ and several peer algorithms.

4.1 Experimental Design

4.1.1 Benchmark Function. In most work in the field of single-objective DOPs, the **moving peaks benchmark (MPB)** [Branke 1999] is used for examining the performance of the DOAs in experiments. The MPB generates landscapes consisting of multiple promising regions (peaks) whose widths, heights, and locations change over time. Peaks generated by this benchmark generator[5] are unimodal, smooth, regular, and symmetric, which make them easy to optimize. To address this shortcoming, many researchers used the **generalized dynamic benchmark generator (GDBG)** [Li et al. 2013] instead of MPB. GDBG is a dynamic version of the commonly used composition-based functions [Liang et al. 2005] used in the field of global optimization in static environments. Using some functions such as Ackley and Rastrigin, the promising regions in GDBG are not as easy to optimize as those used in MPB. However, GDBG is not as flexible and controllable as MPB in generating problem instances with various characteristics, which is why MPB remained the most popular benchmark in the field. The **Generalized Moving Peaks Benchmark (GMPB)** [Yazdani et al. 2021, 2022] is a state-of-the-art DOP benchmark generator in which the shortcomings of MPB and GDBG have been addressed. Unlike MPB, the promising

[5]The majority of work in the field used MPB with conical peaks [Branke and Schmeck 2003].

ALGORITHM 2: Pseudo code of SPSO$_{+AD+AP}$

Input: Fitness function, parameter settings of PSO and SPSO$_{+AP+AD}$

Output: The best found solution when a new solution needs to be deployed

```
1  Initialize the initial individuals;
2  f_generate = 0;
3  repeat
4      if f_generate = 1 then // If all species have converged
5          if N < N_max then // The number of species N is smaller than N_max
6              Randomly initialize m individuals; // Increasing the population size.
7          else // when N = N_max and the population size has reached the maximum threshold.
8              Randomly re-initialize individuals of the species with the worst g^★; // Increasing global diversity by anti-convergence
9          f_generate = 0;
10     L ← individuals; // All individuals including the possible newly generated ones in Line 6.
11     S_{i∈{1,...,N}} ← Algorithm 1(L, n); // Form species by executing Algorithm 1 on L.
12     if number of species has changed in comparison to the last iteration then
13         Update d_boa = (Ub−Lb)/∜n (see Table 3), and then update r_generate and r_excl;
14     foreach Species S_i do
15         Determine s_i using (3);
16         T = ∅;
17         if s_i ≤ r_track then
18             Tag S_i as a tracker and T = T ∪ i;
19         else
20             Tag S_i as a non-tracker;
21     if for all species S_i, s_i ≤ r_generate then
22         f_generate = 1;
23     foreach pair of species S_i and S_j (i ≠ j) do // Exclusion component for removing redundant species.
24         if ‖g*_i − g*_j‖ ≤ r_excl then
25             Remove the inferior species; // Individuals of the species with inferior g* are removed.
       // Determining activation status of species by adaptive deactivation (Lines 26 to 35).
26     if there is a new tracker then
27         r_a = r_a^max; // Reset r_a to the upper bound value if at least a non-tracker species has become tracker in the last
           iteration.
28     if {∄S_{i∈T} : s_i > r_a} then // If spatial sizes of all tracker species are smaller than r_a.
29         β = β · γ;
30         Constrict r_a using Equation (4);
31     Activate all species;
32     foreach Species S_i do
33         if s_i ≤ r_a then
34             Deactivate S_i;
35     Keep the best species activated;
36     if a new solution needs to be deployed then
37         Return the best found solution in the current environment; // Output in solving online dynamic optimization problems.
38     foreach active species S_i do
39         S_i = PSO(S_i); // Run an internal iteration of PSO for the species S_i
40     if environment has changed then
41         Estimate shift severity ŝ^(t) using Equation (5); // From the second environmental change.
42         foreach tracker species S_i do
43             Keep g*_i as an individual;
44             Increase local diversity using Equation (6);
45             Perform absorbing bound handing technique [Helwig et al. 2012] for individuals located out of boundaries;
46         Re-evaluate all individuals; // This addresses the outdated memory.
47         r_a^max = ρ · ŝ^(t);
48         r_a = r_a^max;
49         β = 1;
50         r_track = ŝ^(t);
51 until stopping criterion is met;
```

regions of GMPB are complex and challenging to optimize. In addition, unlike GDBG, GMPB is highly configurable and controllable. Additionally, problem instances generated by GMPB pose some characteristics that were not present in previous benchmarks in the field, such as varying symmetry, condition number, and variable interaction.

Table 1. Comparison Algorithms

Algorithm	Ref.	Optimizer
ACF_{PSO}	Yazdani et al. [2022]	PSO [Eberhart and Shi 2001]
psfNBC	Luo et al. [2018]	PSO [Eberhart and Shi 2001]
mDE	Yazdani et al. [2020]	†DE/best/2/bin [Mendes and Mohais 2005]
ImQSO	Kordestani et al. [2019]	PSO [Eberhart and Shi 2001]
CPSO	Yang and Li [2010]	PSO [Eberhart and Shi 2001]
FTMPSO	Yazdani et al. [2013]	PSO [Eberhart and Shi 2001]
CDE	Du Plessis and Engelbrecht [2012]	DE/best/2/bin [Mendes and Mohais 2005]
mCMAES	Yazdani et al. [2020]	CMAES [Hansen and Ostermeier 2001]

† Differential evolution (DE) [Das and Suganthan 2010].

GMPB is capable of generating landscapes with a controllable number of promising regions with a variety of *changing* and configurable characteristics, including symmetry, roughness, modality, irregularity, condition number, and variable interaction. The user can adjust several parameters in GMPB to generate problem instances with a vast variety of morphological and dynamical characteristics and levels of difficulty. In this article, we use two scenarios of GMPB: (1) Scenario 1, in which the variable structure is fully non-separable, and (2) Scenario 2, where the variable structure is partially separable, which is made by composing several sub-functions. Scenario 2 poses some additional characteristics including modularity, heterogeneity, and imbalance [Omidvar et al. 2015]. A detailed description of GMPB is provided in Section S.1 of the supplementary document.

4.1.2 Performance Evaluation. In order to measure the performance of the DOAs, we use *offline-error* [Branke and Schmeck 2003] (E_O), which is the most commonly used performance indicator in the literature [Yazdani et al. 2021b]. E_O calculates the average error of the best found position over all fitness evaluations using the following equation:

$$E_O = \frac{1}{T\vartheta} \sum_{t=1}^{T} \sum_{c=1}^{\vartheta} \left(f^{(t)} \left(\mathbf{x}^{\star(t)} \right) - f^{(t)} \left(\mathbf{x}^{*((t-1)\vartheta+c)} \right) \right), \tag{7}$$

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

4.1.3 Comparison Algorithms. A set of eight DOAs are selected for comparison, which are listed in Table 1. These DOAs represent various types of methods from different perspectives, including different optimizers such as PSO, DE, and CMA-ES, and different population division and management components. For example, ImQSO and CDE are multi-population DOAs that use a fixed number of sub-populations and population size. ACF_{PSO}, FTMPSO, mDE, and mCMAES use index-based clustering, varying population size, and adaptive numbers of sub-populations. CPSO and psfNBC use clustering approaches that form species/sub-populations based on the fitness values and positions of their individuals. Although these two methods have varying numbers of species/sub-populations, their population sizes are fixed. Moreover, FTMPSO, mDE, and mCMAES use conventional deactivation components with a fixed deactivation condition. Finally, ACF_{PSO} benefits from an advanced computational resource allocation method.

In order to have a fair comparison, some modifications have been made to the algorithms. First, the procedure for detecting changes has been removed from all methods. Like many real-world cases where environmental changes are visible [Yazdani et al. 2021a], we assume that

Table 2. Parameter Settings for Scenarios 1 and 2 of GMPB

Parameter	Symbol	Value(s) in Scenario 1	Value(s) in Scenario 2
Number of sub-functions	q	1	5
Sub-function dimensionality	d_i	{5}	{4, 2, 2, 1, 1}
Shift severity	$\tilde{s}$	1, 2, 5	$\mathcal{U}[1,3]^\dagger$,$\mathcal{U}[1,5]^\dagger$
Numbers of promising regions	m	10, 25, 50, 100, 200	$^\diamond$ $\mathcal{U}[5,15]^\dagger$,$\mathcal{U}[15,35]^\dagger$
Evaluations between changes	ϑ	500, 1,000, 2,500, 5,000	2,500, 5,000
Dimension	d	5	10
Height severity	$\tilde{h}$	7	$\mathcal{U}[5,9]^\dagger$
Width severity	$\tilde{w}$	1	$\mathcal{U}[0.5,1.5]^\dagger$
Weight of sub-function i	ω_i	1	$\mathcal{U}[0.5,3]^\dagger$
Angle severity	$\tilde{\theta}$	$\pi/9$	$\mathcal{U}[\pi/12,\pi/6]^\dagger$
Irregularity parameter τ severity	$\tilde{\tau}$	0.2	$\mathcal{U}[0.025,0.075]^\dagger$
Irregularity parameter η severity	$\tilde{\eta}$	2	$\mathcal{U}[1,3]^\dagger$
Search range	$[Lb, Ub]^d$	$[-100,100]^d$	$[-50,50]^d$
Height range	$[h_{\min}, h_{\max}]$	$[30,70]$	
Width range	$[w_{\min}, w_{\max}]^d$	$[1,12]^d$	
Angle range	$[\theta_{\min}, \theta_{\max}]$	$[-\pi,\pi]$	
Irregularity parameter τ range	$[\tau_{\min}, \tau_{\max}]$	$[-1,1]$	
Irregularity parameter η range	$[\eta_{\min}, \eta_{\max}]$	$[-20,20]$	
Number of environments	T	100	

†For each sub-function in Scenario 2, the values defined by $\mathcal{U}[a, b]$ are generated randomly in $[a, b]$ with uniform distribution.

$^\diamond$This defines the number of promising regions in each sub-function independently. The total number of promising regions in Scenario 2 is equal to the product of the number of promising regions in all sub-functions.

the algorithms are informed about the environmental changes [Nguyen 2011]. Thus, they do not need to detect changes. Second, for all PSO-based algorithms, we use PSO with constriction factor [Eberhart and Shi 2001]. Finally, all methods that require knowledge about the shift severity adopt the shift severity estimation method from Yazdani et al. [2018]. Note that the estimated shift severity for the first two environments is temporarily set to one in all the experiments, as shift severity cannot be estimated before the second environmental change [Yazdani et al. 2018].

4.1.4 Parameter Settings. Table 2 shows the parameter settings for Scenarios 1 and 2 of GMPB. The experiments are done on the problem instances with various number of promising regions, shift severity values, change frequencies, dimensions, and variable interaction structures.

In SPSO$_{+AP+AD}$, we use constriction factor PSO [Eberhart and Shi 2001] as the optimization component. We use the values suggested in Eberhart and Shi [2001] for the parameter settings of the PSO in SPSO$_{+AP+AD}$. For SPSO$_{+AP+AD}$, r_{track} is set to the estimated shift severity $\hat{s}$ based on the provided descriptions in Section 3.1. The remaining parameters of SPSO$_{+AP+AD}$, namely r_{generate}, $\mathfrak{m}$, ρ, γ, μ, r_{excl}, and $N_{\max}$, are set based on a sensitivity analysis we conducted. The detailed results and analysis of this study can be found in Section S.2 of the supplementary document. Our sensitivity analysis shows that there are multiple combinations of parameter settings that result in the best performance of SPSO$_{+AP+AD}$, indicating that the proposed algorithm is not highly sensitive to the values of these parameters. A summary of the parameter settings for SPSO$_{+AP+AD}$ is shown in Table 3.

We use the parameter settings suggested in the original references for the comparison algorithms, as our investigations have shown that these settings yield their best performance.

Table 3. $SPSO_{+AP+AD}$ Parameter Settings

Method	Parameter	Value	Reference
PSO	χ	0.729843788	Eberhart and Shi [2001]
	C_1, C_2	2.05	Eberhart and Shi [2001]
	Neighborhood topology	global star	Eberhart and Shi [2001]
	Population size*	5	Blackwell et al. [2008]; Yazdani et al. [2020]
$SPSO_{+AP+AD}$	r_{track}	$\hat{s}$**	Descriptions in Section 3.1
	ρ	0.7	Sensitivity analysis in Figure S.2
	γ	0.1	Sensitivity analysis in Figure S.2
	μ	0.2†	Sensitivity analysis in Figure S.2
	r_{generate}	$0.3 \times d_{\text{boa}}$‡	Sensitivity analysis in Table S.1
	$\mathfrak{m}$	5	Sensitivity analysis in Table S.1
	r_{excl}	$0.5 \times d_{\text{boa}}$‡	Sensitivity analysis in Table S.2
	$N_{\max}$	30	Sensitivity analysis in Table S.3

*This parameter represents the size of each species. Note that the overall population size is adaptive, with an initial value of 50 in our experiments.
**$\hat{s}$ is temporarily set to one for the first two environments in all the experiments. $\hat{s}$ value is updated by Equation (5) right after the second environmental change.
†μ value is used for determining $r_a^{\min}$, where $r_a^{\min} = \mu \cdot \sqrt{d}$.
‡$d_{\text{boa}} = \frac{\text{Ub}-\text{Lb}}{\sqrt[d]{N}}$ [Blackwell and Branke 2006; Blackwell et al. 2008], where Ub and Lb are the upper and lower bounds of the search space, respectively, and N is the number of sub-populations/species.

4.2 Experimental Results

All experiments in this section are done 31 times with different random seed values and the average E_O (and standard error in parenthesis) reported in tables. In each row of the tables, the symbols +, -, and ≈ indicate that the compared algorithm is statistically significantly better than $SPSO_{+AP+AD}$, statistically significantly worse than $SPSO_{+AP+AD}$, and statistically equivalent to $SPSO_{+AP+AD}$, respectively, based on the Wilcoxon rank-sum test with Holm–Bonferroni p-value correction and $\alpha = 0.05$. We highlight the best result in each problem instance and those that are statistically equivalent to it.

4.2.1 Effect of the Proposed Components on the Performance. In this section, we investigate the effectiveness of the proposed population division and management (AP) and computational resource allocation (AD) components. To this end, we compare the performance of $SPSO_{+AP+AD}$ with three methods:

- SPSO: This is a species-based PSO with a fixed population size. To form species, this method uses the same clustering approach as the one used in the proposed method. Simple Round Robin is used for SPSO as the computational resource allocation component. We also use anti-convergence and exclusion for maintaining the global diversity and exploration capability of this method [Blackwell and Branke 2006; Blackwell et al. 2008]. Following Blackwell et al. [2008], we set the population size to 50 for this method.
- $SPSO_{+AP}$: This is similar to $SPSO_{+AP+AD}$ but without the proposed AD (instead, the traditional Round Robin method is used for allocating computational resources to species).
- $SPSO_{+AP+TD}$: This is similar to $SPSO_{+AP+AD}$ but it uses the ***traditional* deactivation (TD)** component [Kamosi et al. 2010] instead of AD.

The experiments in this section are done on GMPB Scenario 1 with $m = \{25, 50, 100\}$, $\vartheta = 5{,}000$, and $\tilde{s} = 1$, and the average E_O (and standard error in parenthesis) obtained by SPSO, $SPSO_{+AP}$, $SPSO_{+AP+TD}$, and $SPSO_{+AP+AD}$ are reported in Table 4.

Table 4. Investigating the Effect of the Proposed Adaptive Population Division and Management and Adaptive Deactivation Computational Resource Allocation Components by Comparing the Obtained Average E_O (and Standard Error in Parenthesis) by $SPSO_{+AP+AD}$ with SPSO, $SPSO_{+AP}$, and $SPSO_{+AP+TD}$ in Problem Instances Generated by GMPB Scenario 1 with $m = \{25, 50, 100\}$, $\vartheta = 5{,}000$, and $\tilde{s} = 1$

m	Algorithms			
	$SPSO_{+AP+AD}$	SPSO	$SPSO_{+AP}$	$SPSO_{+AP+TD}$
25	3.86(0.11)	5.26(0.14)−	4.35(0.08)−	4.09(0.08)−
50	4.07(0.07)	5.41(0.11)−	4.47(0.08)−	4.40(0.05)−
100	4.12(0.08)	5.43(0.14)−	4.61(0.10)−	4.55(0.08)−

The highlighted entries are significantly better using Wilcoxon rank-sum test with Holm–Bonferroni p-value adjustment ($\alpha = 0.05$).

Comparing the results obtained by SPSO and $SPSO_{+AP}$ demonstrates the effectiveness of the proposed adaptive population division and management component. Unlike SPSO, whose number of species is fixed over time for all problem instances, $SPSO_{+AP}$ systematically adapts its number of species to the number of discovered promising regions. As SPSO has a fixed number of species, it suffers from (1) wastage of computational resources by redundant species when the number of promising regions is smaller than the number of species (i.e., $N > m$) and (2) inadequate coverage of promising regions when their number is larger than the number of species (i.e., $N < m$). These shortcomings have been addressed in $SPSO_{+AP}$ by the proposed species generation/removal method.

The next observation is obtained by comparing the results of $SPSO_{+AP}$, which uses Round Robin without deactivating any species, and $SPSO_{+AP+TD}$, which uses traditional deactivation component. As can be seen in Table 4, thanks to the traditional deactivation component, the results obtained by $SPSO_{+AP+TD}$ are slightly superior to those obtained by $SPSO_{+AP}$, in particular in problem instances with fewer number of promising regions (i.e., smaller m). However, the performance gap between $SPSO_{+AP+TD}$ and $SPSO_{+AP}$ diminishes when solving problem instances with a larger number of promising regions (i.e., larger m), where more species are involved. This is because in the traditional deactivation component, the deactivation radius, which is usually set to a very small value, is fixed. In such cases, species typically do not receive enough computational resources to converge in each environment, and as a result, their spatial size does not shrink enough to be deactivated. This leads to the ineffectiveness of the deactivation process in such circumstances.

According to the reported results in Table 4, $SPSO_{+AP+AD}$ outperforms other methods thanks to both proposed components. Comparing the results of $SPSO_{+AP}$ and $SPSO_{+AP+AD}$ demonstrates the effectiveness of using the proposed computational resource allocation component. In addition, as a result of the systematic control of the deactivation radius used in the proposed adaptive deactivation component, the performance of $SPSO_{+AP+AD}$ is superior to that of $SPSO_{+AP+TD}$, which uses fixed deactivation radius. Moreover, unlike the traditional deactivation component, the proposed adaptive deactivation component maintains its effectiveness in problem instances with larger numbers of the promising regions. The reasons behind the improvements that resulted by using the proposed adaptive deactivation component in $SPSO_{+AP+AD}$ are:

- Improving the exploitation around the best found position
- Improving the tracking performance by allowing all tracker species to perform tracking task according to the limitation degree of the computational resources
- Improving the tracking performance by allowing new tracker species to converge to the summit

Table 5. Results Obtained (Average E_O and Standard Error in Parenthesis) by the Algorithms on the Problem Instances Generated by GMPB Scenario 1 with Different Number of Promising Regions $m = \{10, 25, 50, 100, 200\}$, $\vartheta = 5{,}000$, and $\tilde{s} = 1$

m	Algorithms								
	$SPSO_{+AP+AD}$	ACF_{PSO}	FTMPSO	mDE	ImQSO	CPSO	CDE	psfNBC	mCMAES
10	3.47(0.09)	4.67(0.17)−	5.82(0.13)−	5.83(0.13)−	5.41(0.17)−	9.27(0.20)−	10.30(0.54)−	4.59(0.12)−	3.96(0.16)−
25	3.86(0.11)	4.78(0.15)−	5.69(0.81)−	5.85(0.10)−	6.37(0.18)−	8.80(0.15)−	11.92(0.01)−	5.93(0.13)−	4.53(0.16)−
50	4.07(0.07)	4.80(0.10)−	5.77(0.09)−	6.08(0.09)−	6.58(0.16)−	8.27(0.13)−	9.91(0.45)−	6.30(0.14)−	4.61(0.13)−
100	4.12(0.08)	4.70(0.11)−	5.68(0.08)−	6.37(0.11)−	6.68(0.15)−	7.87(0.13)−	9.52(0.38)−	6.40(0.15)−	4.80(0.13)−
200	4.26(0.06)	4.61(0.11)−	5.67(0.08)−	6.26(0.11)−	6.57(0.12)−	7.69(0.15)−	10.22(0.65)−	6.37(0.14)−	4.70(0.12)−

The highlighted entries are significantly better using Wilcoxon rank-sum test with Holm–Bonferroni p-value adjustment ($\alpha = 0.05$).

Table 6. Results Obtained (Average E_O and Standard Error in Parenthesis) by the Algorithms on the Problem Instances Generated by GMPB Scenario 1 with Different Shift Severity Values $\tilde{s} = \{1, 2, 5\}$, $m = 10$, and $\vartheta = 5{,}000$

$\tilde{s}$	Algorithms								
	$SPSO_{+AP+AD}$	ACF_{PSO}	FTMPSO	mDE	ImQSO	CPSO	CDE	psfNBC	mCMAES
1	3.47(0.09)	4.67(0.17)−	5.82(0.13)−	5.83(0.13)−	5.41(0.17)−	9.27(0.20)−	10.30(0.54)−	4.59(0.12)−	3.96(0.16)−
2	5.23(0.15)	5.72(0.17)−	7.28(0.17)−	8.24(0.17)−	7.24(0.16)−	12.43(0.24)−	15.72(0.84)−	6.32(0.13)−	5.29(0.21)≈
5	9.11(0.19)	9.08(0.21)≈	10.45(0.20)−	13.36(0.18)−	11.42(0.20)−	18.92(0.31)−	29.98(1.39)−	13.64(0.20)−	13.54(0.36)−

The highlighted entries are significantly better using Wilcoxon rank-sum test with Holm–Bonferroni p-value adjustment ($\alpha = 0.05$). Multiple highlighted entries in each row are statistically similar.

- Improving the coverage performance by maintaining activation of newly generated and non-tracker species

4.2.2 Comparison with Peer Methods. In this section, the performance of $SPSO_{+AP+AD}$ is compared with that of the peer algorithms from Table 1 on a number of problem instances generated by GMPB Scenario 1 and Scenario 2. Obtained average E_O results and standard errors are reported in Tables 5, 6, 7, and 8.

Table 5 shows the results obtained by the algorithms on problem instances generated by GMPB Scenario 1 with different numbers of the promising regions. As shown in this table, $SPSO_{+AP+AD}$ ranks first among all algorithms. According to this table, problem instances with larger numbers of the promising regions are often more challenging for the algorithms. This is more noticeable in the results obtained by the adaptive multi-population DOAs that increase the number of sub-populations with the rise in the number of promising regions. Therefore, due to large population size values, each sub-population receives a smaller amount of the computational resources in each environment. According to the reported results, $SPSO_{+AP+AD}$ can maintain its high efficiency for the problem instances with larger numbers of the promising regions.

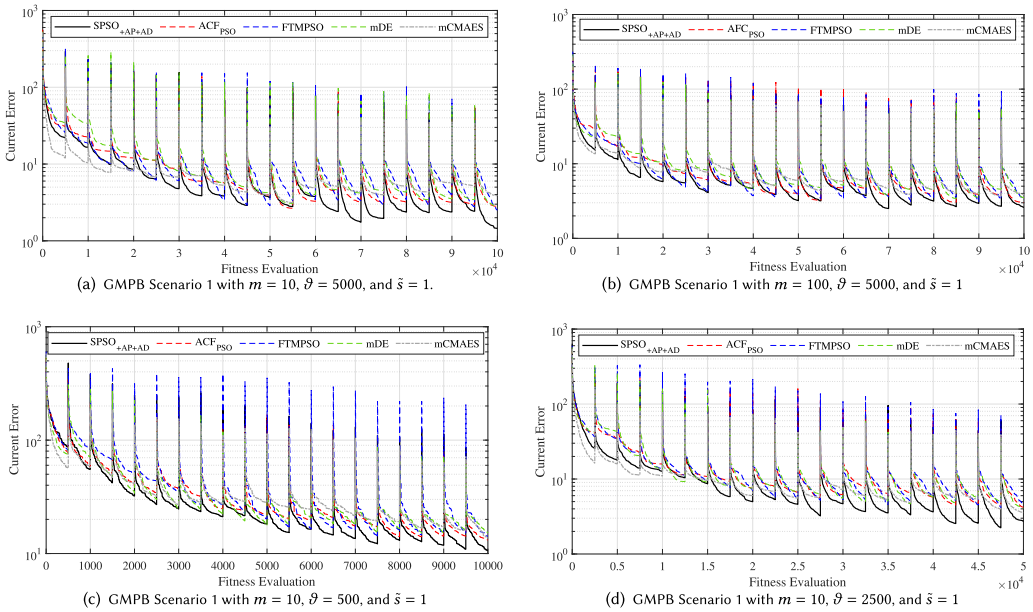
Table 6 compares the results obtained by the algorithms on problem instances generated by GMPB Scenario 1 with different degrees of shift severity. Based on the reported results in this table, problem instances with larger values of shift severity pose a greater challenge for the algorithms since the tracking needs more computational resources. The reason is that the summits of the promising regions relocate to farther positions after each environmental change when the shift severity value is large. Moreover, the fitness drop values after each environmental change are large due to the large summit relocation lengths, which result in E_O deterioration. As shown in this table, $SPSO_{+AP+AD}$ ranks first among all algorithms in the problem instances with different shift severity values.

Table 7 shows the results obtained by the algorithms on problem instances generated by GMPB Scenario 1 with different change frequency values. According to this table, the higher the change

Table 7. Results Obtained (Average E_O and Standard Error in Parenthesis) by the Algorithms on the Problem Instances Generated by GMPB Scenario 1 with Different Change Frequencies $\vartheta = \{500, 1{,}000, 2{,}500, 5{,}000\}$, $m = 10$, and $\tilde{s} = 1$

ϑ	Algorithms								
	$SPSO_{+AP+AD}$	ACF_{PSO}	FTMPSO	mDE	ImQSO	CPSO	CDE	psfNBC	mCMAES
500	13.49(0.38)	18.46(0.78)−	20.26(0.51)−	18.99(0.47)−	21.07(0.38)−	48.16(1.48)−	66.28(6.27)−	104.2(15.12)−	17.04(0.93)−
1000	8.11(0.27)	11.62(0.43)−	13.55(0.32)−	12.43(0.22)−	12.30(0.37)−	28.23(0.79)−	40.47(4.28)−	19.39(0.71)−	9.56(0.53)−
2500	4.77(0.17)	6.79(0.25)−	8.37(0.18)−	7.49(0.17)−	7.70(0.22)−	14.52(0.31)−	17.27(1.02)−	7.21(0.14)−	5.65(0.28)−
5000	3.47(0.09)	4.67(0.17)−	5.82(0.13)−	5.83(0.13)−	5.41(0.17)−	9.27(0.20)−	10.30(0.54)−	4.59(0.12)−	3.96(0.16)−

The highlighted entries are significantly better using Wilcoxon rank-sum test with Holm–Bonferroni p-value adjustment ($\alpha = 0.05$).



(a) GMPB Scenario 1 with $m = 10$, $\vartheta = 5000$, and $\tilde{s} = 1$.
(b) GMPB Scenario 1 with $m = 100$, $\vartheta = 5000$, and $\tilde{s} = 1$
(c) GMPB Scenario 1 with $m = 10$, $\vartheta = 500$, and $\tilde{s} = 1$
(d) GMPB Scenario 1 with $m = 10$, $\vartheta = 2500$, and $\tilde{s} = 1$

Fig. 1. Current error plots of $SPSO_{+AP+AD}$, ACF_{PSO}, FTMPSO, mDE, and mCMAES on four problem instances generated by GMPB Scenario 1 with different configurations and for 20 environments. All sub-figures are illustrated according to the average of the current errors for 31 independent runs.

frequency, the more challenging the problem becomes due to the very limited computational resources available in each environment. Note that lower ϑ values indicate higher change frequencies. As can be seen in this table, $SPSO_{+AP+AD}$ also outperforms other methods in the problem instances with higher change frequencies. This superiority demonstrates the high impact of the proposed resource allocation component in efficiently managing the consumption of the very limited computational resources by each species in the problem instances with high change frequency values.

Figure 1 illustrates the superiority of the change reaction capability and convergence speed of the proposed algorithm in comparison to peer DOAs[6] on problem instances generated by GMPB Scenario 1 with various numbers of promising regions and change frequencies.

Table 8 reports the results obtained by the DOAs on GMPB Scenario 2 with four different configurations. As shown in Table 2, GMPB Scenario 2 is modular and constructed by the composition of

[6]For the sake of plot clarity, we have selected the top five DOAs from Tables 5, 6, and 7 to be displayed in Figure 1.

Table 8. Results Obtained (Average E_O and Standard Error in Parenthesis) by the Algorithms on GMPB Scenario 2 with Various Parameter Settings from Table 2

GMPB Scenario 2 Parameters			Algorithms								
$s^{\dagger}$	$m^{\dagger}$	ϑ	$SPSO_{+AP+AD}$	ACF_{PSO}	FTMPSO	mDE	ImQSO	CPSO	CDE	psfNBC	mCMAES
$\mathcal{U}[1, 3]^*$	$\mathcal{U}[5, 15]$	5,000	16.85(0.91)	18.55(0.84)−	25.41(1.19)−	22.02(1.02)−	48.95(2.50)−	18.32(0.98)−	33.67(1.79)−	19.70(0.78)−	25.17(1.21)−
$\mathcal{U}[1, 5]$	$\mathcal{U}[5, 15]$	5,000	18.69(1.05)	20.31(0.91)−	28.04(1.32)−	24.69(1.24)−	52.08(2.76)−	21.06(1.12)−	39.71(2.24)−	23.92(1.24)−	30.59(1.70)−
$\mathcal{U}[1, 3]$	$\mathcal{U}[15, 35]$	5,000	16.08(0.86)	17.61(0.76)−	24.37(1.10)−	21.40(0.95)−	40.52(2.04)−	16.24(0.76)≈	29.98(1.73)−	19.96(0.89)−	23.39(1.08)−
$\mathcal{U}[1, 3]$	$\mathcal{U}[5, 15]$	2,500	23.19(1.07)	23.38(1.09)≈	30.23(1.39)−	28.08(1.21)−	48.97(2.41)−	23.52(1.20)≈	39.18(2.12)−	25.48(1.26)−	30.25(1.56)−

The highlighted entries are significantly better using Wilcoxon rank-sum test with Holm–Bonferroni p-value adjustment ($\alpha = 0.05$). Multiple highlighted entries in each row are statistically similar.

†Values are defined randomly and independently for each sub-function of GMPB Scenario 2.

$^{*}\mathcal{U}[a, b]$generates a random number with uniform distribution in $[a, b]$.

five heterogeneous sub-functions. The modular property of GMPB Scenario 2 results in an exponential increase in the number of promising regions [Yazdani et al. 2022], resulting in significant intensification of the multimodality and roughness of the search space. Consequently, GMPB Scenario 2 is more challenging than Scenario 1.

Results in Table 8 show that increasing the shift severity values makes it more difficult to solve GMPB Scenario 2 due to longer displacement distances of promising regions. In addition, Table 8 compares the results obtained on GMPB Scenario 2 with m set to $\mathcal{U}[5, 15]$ and $\mathcal{U}[15, 35]$. As stated before, the number of promising regions in GMPB Scenario 2 is equal to the product of the number of promising regions in all sub-functions. Therefore, the number of promising regions is very high in both cases where m is set to either $\mathcal{U}[5, 15]$ or $\mathcal{U}[15, 35]$. The results obtained by the DOAs on GMPB Scenario 2 with $\mathcal{U}[15, 35]$ promising regions for each sub-function are better than those obtained on problem instances with $m = \mathcal{U}[5, 15]$. This is because there are significantly more promising regions when $m = \mathcal{U}[15, 35]$ for each sub-function, which in turn increases the likelihood of covering high-quality promising regions by DOAs [Yazdani et al. 2022]. Finally, the worst results are obtained on GMPB Scenario 2 with a higher change frequency ($\vartheta = 2{,}500$), as it reduces the available computational resources (i.e., function evaluations) in each environment, thereby decreasing the exploration and exploitation capabilities of the DOAs. Overall, $SPSO_{+AP+AD}$ obtained the best results in solving the four problem instances generated by GMPB Scenario 2.

5 CONCLUSION

In this article, we proposed an adaptive species-based PSO called $SPSO_{+AP+AD}$ with two novel components: adaptive population division and management, and adaptive deactivation-based computational resource allocation. $SPSO_{+AP+AD}$ forms sub-populations (species) using a clustering method that considers both fitness and positions of individuals in each iteration. Unlike most existing species-based dynamic optimization algorithms that have a fixed population size, $SPSO_{+AP+AD}$ adapts the overall population size and the number of sub-populations according to the discovered promising regions. We also introduced a novel systematic adaptive deactivation component to adaptively allocated computational resources to the sub-populations based on the role and convergence status of the sub-populations, as well as the characteristics of dynamic optimization problems including the number of discovered promising regions and shift severity. By applying the proposed resource allocation component, exploitation and exploration capabilities are substantially enhanced. We then proved the efficiency of the proposed algorithm on the Generalized Moving Peaks Benchmark with different numbers of promising regions, shift severity degrees, dimensions, and change frequencies in comparison to those of several peer algorithms. The results demonstrated the effectiveness and accuracy of the proposed algorithm.

Similar to the majority of the state-of-the-art multi-population DOAs [Yazdani et al. 2021a, b], the proposed method in this article also uses several parameters that need to be set by the user.

A future research direction is to design self-adaptive [Novoa-Hernández et al. 2016] and auto-parameter tuning mechanisms [Huang et al. 2020] for some parameters of the proposed method, in particular those that belong to the proposed adaptive deactivation and adaptive population handling components. Using such techniques is also vital in solving many real-world DOPs that possess heterogeneous dynamical behavior over time [Yazdani et al. 2021a]. In such problems, using fixed values for many parameters would be inefficient since their optimal values change over time following changes in dynamical characteristics such as change intensity and frequency.

Most work in the DOP literature, including this article, focuses on problems whose search ranges are identical in all dimensions, and problems with various search ranges in different dimensions have rarely been investigated. In fact, most DOAs, in particular multi-population ones, use components that work based on Euclidean distances that may not be effective for solving DOPs with various search ranges in different dimensions. Investigating the effectiveness of the existing algorithms/components in solving such DOPs and developing algorithms/components for tackling these problems is an important future direction.

Like most work in the DOP literature, we examined the effectiveness of the proposed method in solving artificial benchmark problems. An important area of future work will be adapting the proposed method for solving a real-world DOP. An important group of real-world DOPs is the dynamic covering location problem [Ankrah et al. 2019; Plastria 2002]. In these problems, the aim is to relocate a finite set of facilities to serve demands/customers whose numbers and locations change over time. To react to changes in demands/customers, the dynamic algorithms need to track the optimal positions of the facilities.

REFERENCES

Ignacio G. Del Amo, David A. Pelta, and Juan R. González. 2010. Using heuristic rules to enhance a multiswarm PSO for dynamic environments. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–8.

Reginald Ankrah, Benjamin Lacroix, John McCall, Andrew Hardwick, and Anthony Conway. 2019. Introducing the dynamic customer location-allocation problem. In *IEEE Congress on Evolutionary Computation*. IEEE, 3157–3164.

Jaroslaw Arabas, Zbigniew Michalewicz, and Jan Mulawka. 1994. GAVaPS-a genetic algorithm with varying population size. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. IEEE, 73–78.

Radhia Azzouz, Slim Bechikh, and Lamjed Ben Said. 2017. Dynamic multi-objective optimization using evolutionary algorithms: A survey. In *Recent Advances in Evolutionary Multi-objective Optimization*. Springer, 31–70.

Tim Blackwell and Juergen Branke. 2004. Multi-swarm optimization in dynamic environments. In *Applications of Evolutionary Computing*, Günther R. Raidl et al. (Ed.), Vol. 3005. Lecture Notes in Computer Science, 489–500.

Tim Blackwell and Juergen Branke. 2006. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation* 10, 4 (2006), 459–472.

Tim Blackwell, Juergen Branke, and Xiaodong Li. 2008. Particle swarms for dynamic optimization problems. In *Swarm Intelligence: Introduction and Applications*, Christian Blum and Daniel Merkle (Eds.). Springer Lecture Notes in Computer Science, 193–217.

Juergen Branke. 1999. Memory enhanced evolutionary algorithms for changing optimization problems. In *IEEE Congress on Evolutionary Computation*, Vol. 3. IEEE, 1875–1882.

Juergen Branke. 2012. *Evolutionary Optimization in Dynamic Environments*, Vol. 3. Springer Science & Business Media.

Juergen Branke and Hartmut Schmeck. 2003. Designing evolutionary algorithms for dynamic optimization problems. In *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui (Eds.). Springer Natural Computing Series, 239–262.

Renzhi Chen, Ke Li, and Xin Yao. 2017. Dynamic multiobjectives optimization with a changing number of objectives. *IEEE Transactions on Evolutionary Computation* 22, 1 (2017), 157–171.

André L. V. Coelho and Daniel G. de Oliveira. 2008. Dynamically tuning the population size in particle swarm optimization. In *Proceedings of the 2008 ACM Symposium on Applied Computing*. 1782–1787.

Carlos Cruz, Juan R. González, and David A. Pelta. 2011. Optimization in dynamic environments: A survey on problems, methods and measures. *Soft Computing* 15, 7 (2011), 1427–1448.

Paul Darwen and Xin Yao. 1996. Every niching method has its niche: Fitness sharing and implicit sharing compared. In *International Conference on Parallel Problem Solving from Nature*. Springer, 398–407.

Paul J. Darwen and Xin Yao. 1997. Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation* 1, 2 (1997), 101–108.

Swagatam Das and Ponnuthurai Nagaratnam Suganthan. 2010. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation* 15, 1 (2010), 4–31.

Kalyanmoy Deb, Udaya Bhaskara Rao N., and Sindhya Karthik. 2007. Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In *International Conference on Evolutionary Multi-criterion Optimization*. Springer, 803–817.

Mathys C. Du Plessis and Andries P. Engelbrecht. 2012. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research* 218, 1 (2012), 7–20.

Mathys C. du Plessis and Andries P. Engelbrecht. 2013. Self-adaptive differential evolution for dynamic environments with fluctuating numbers of optima. In *Metaheuristics for Dynamic Optimization*, Enrique Alba et al. (Ed.). Springer, Berlin, 117–145.

Russell Eberhart and Y. Shi. 2001. Comparing inertia weights and constriction factors in particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, Vol. 1. IEEE, 84–88.

Haobo Fu, Bernhard Sendhoff, Ke Tang, and Xin Yao. 2015. Robust optimization over time: Problem difficulties and benchmark problems. *IEEE Transactions on Evolutionary Computation* 19, 5 (2015), 731–745.

Chi-Keong Goh and Kay Chen Tan. 2008. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 13, 1 (2008), 103–127.

Udit Halder, Dipankar Maity, Preetam Dasgupta, and Swagatam Das. 2011. Self-adaptive cluster-based differential evolution with an external archive for dynamic optimization problems. In *Swarm, Evolutionary, and Memetic Computing*, Bijaya Ketan Panigrahi et al. (Eds). Springer, Berlin, 19–26.

Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9, 2 (2001), 159–195.

Sabine Helwig, Juergen Branke, and Sanaz Mostaghim. 2012. Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 17, 2 (2012), 259–271.

Changwu Huang, Yuanxiang Li, and Xin Yao. 2020. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation* 24, 2 (2020), 201–216.

Min Jiang, Zhongqiang Huang, Liming Qiu, Wenzhen Huang, and Gary G. Yen. 2017. Transfer learning-based dynamic multiobjective optimization algorithms. *IEEE Transactions on Evolutionary Computation* 22, 4 (2017), 501–514.

Shouyong Jiang, Juan Zou, Shengxiang Yang, and Xin Yao. 2022. Evolutionary dynamic multi-objective optimisation: A survey. *Comput. Surveys* 55, 4 (2022), 1–47.

Yaochu Jin, Ke Tang, Xin Yu, Bernhard Sendhoff, and Xin Yao. 2013. A framework for finding robust optimal solutions over time. *Memetic Computing* 5, 1 (2013), 3–18.

Masoud Kamosi, Ali B. Hashemi, and Mohammad Reza Meybodi. 2010. A hibernating multi-swarm optimization algorithm for dynamic environments. In *Nature and Biologically Inspired Computing*. IEEE, 363–369.

Borhan Kazimipour, Mohammad Nabi Omidvar, A. Kai Qin, Xiaodong Li, and Xin Yao. 2019. Bandit-based cooperative coevolution for tackling contribution imbalance in large-scale optimization problems. *Applied Soft Computing* 76 (2019), 265–281.

James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of International Conference on Neural Networks (ICNN'95)*, Vol. 4. IEEE, 1942–1948.

Javidan Kazemi Kordestani, Mohammad Reza Meybodi, and Amir Masoud Rahmani. 2019. A note on the exclusion operator in multi-swarm PSO algorithms for dynamic environments. *Connection Science* 32, 3 (2019), 1–25.

Changhe Li, Michalis Mavrovouniotis, Shengxiang Yang, and Xin Yao. 2013. *Benchmark Generator for the IEEE WCCI-2014 Competition on Evolutionary Computation for Dynamic Optimization Problems*. Technical Report. De Montfort University.

Changhe Li, Trung Thanh Nguyen, Ming Yang, Shengxiang Yang, and Sanyou Zeng. 2015. Multi-population methods in unconstrained continuous dynamic environments: The challenges. *Information Sciences* 296 (2015), 95–118.

Changhe Li and Shengxiang Yang. 2009. A clustering particle swarm optimizer for dynamic optimization. In *IEEE Congress on Evolutionary Computation*. IEEE, 439–446.

Changhe Li and Shengxiang Yang. 2012. A general framework of multipopulation methods with clustering in undetectable dynamic environments. *IEEE Transactions on Evolutionary Computation* 16, 4 (2012), 556–577.

Xiaodong Li, Jürgen Branke, and Tim Blackwell. 2006. Particle swarm with speciation and adaptation in a dynamic environment. In *Conference on Genetic and Evolutionary Computation*. ACM, 51–58.

Xiaodong Li, Michael G. Epitropakis, Kalyanmoy Deb, and Andries Engelbrecht. 2016. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation* 21, 4 (2016), 518–538.

Jing Liang, Ponnuthurai N. Suganthan, and Kalayanmoy Deb. 2005. Novel composition test functions for numerical global optimization. In *Swarm Intelligence Symposium*. IEEE, 68–75.

Xiao-Fang Liu, Yu-Ren Zhou, Xue Yu, and Ying Lin. 2019. Dual-archive-based particle swarm optimization for dynamic optimization. *Applied Soft Computing* 85 (2019), 105876.

Qiang Lu and Xin Yao. 2005. Clustering and learning gaussian distribution for continuous optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35, 2 (2005), 195–204.

Wenjian Luo, Xin Lin, Jiajia Zahng, and Mike Preuss. 2021. A survey of nearest-better clustering in swarm and evolutionary computation. In *IEEE Congress on Evolutionary Computation*, 1961–1967.

Wenjian Luo, Juan Sun, Chenyang Bu, and Houjun Liang. 2016. Species-based particle swarm optimizer enhanced by memory for dynamic optimization. *Applied Soft Computing* 47 (2016), 130–140.

Wenjian Luo, Juan Sun, Chenyang Bu, and Ruikang Yi. 2018. Identifying species for particle swarm optimization under dynamic environments. In *Symposium Series on Computational Intelligence (SSCI'18)*. IEEE, 1921–1928.

Wenjian Luo, Ruikang Yi, Bin Yang, and Peilan Xu. 2019. Surrogate-assisted evolutionary framework for data-driven dynamic optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence* 3, 2 (2019), 137–150.

Xuemin Ma, Jingming Yang, Hao Sun, Ziyu Hu, and Lixin Wei. 2021. Multiregional co-evolutionary algorithm for dynamic multiobjective optimization. *Information Sciences* 545 (2021), 1–24.

Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. 2017. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation* 33 (2017), 1–17.

Rui Mendes and Arvind S. Mohais. 2005. DynDE: A differential evolution for dynamic optimization problems. In *IEEE Congress on Evolutionary Computation*, Vol. 3. IEEE, 2808–2815.

Arrchana Muruganantham, Kay Chen Tan, and Prahlad Vadakkepat. 2015. Evolutionary dynamic multiobjective optimization via Kalman filter prediction. *IEEE Transactions on cybernetics* 46, 12 (2015), 2862–2873.

Changhe Li Trung Thanh Nguyen, Ming Yang, Michalis Mavrovouniotis, and Shengxiang Yang. 2016. An adaptive multipopulation framework for locating and tracking multiple optima. *IEEE Transactions on Evolutionary Computation* 20, 4 (2016), 590–605.

Trung Thanh Nguyen. 2011. *Continuous Dynamic Optimisation using Evolutionary Algorithms*. Ph.D. Dissertation. University of Birmingham.

Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. 2012. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation* 6 (2012), 1–24.

Trung Thanh Nguyen and Xin Yao. 2012. Continuous dynamic constrained optimization-the challenges. *IEEE Transactions on Evolutionary Computation* 16, 6 (2012), 769–786.

Pavel Novoa, David A. Pelta, Carlos Cruz, and Ignacio García del Amo. 2009. Controlling particle trajectories in a multi-swarm approach for dynamic optimization problems. In *Methods and Models in Artificial and Natural Computation. A Homage to Professor Mira's Scientific Legacy*, José Mira et al. (Eds). Springer, Berlin, 285–294.

Pavel Novoa-Hernández, Carlos Cruz Corona, and David A. Pelta. 2016. Self-adaptation in dynamic environments—A survey and open issues. *International Journal of Bio-Inspired Computation* 8, 1 (2016), 1–13.

Pavel Novoa-Hernández, David A. Pelta, and Carlos Cruz Corona. 2010. *Improvement Strategies for Multi-swarm PSO in Dynamic Environments*. Springer, Berlin, 371–383.

Mohammad Nabi Omidvar, Xiaodong Li, and Ke Tang. 2015. Designing benchmark problems for large-scale continuous optimization. *Information Sciences* 316 (2015), 419–436.

Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. 2021a. A review of population-based metaheuristics for large-scale black-box global optimization-Part I. *IEEE Transactions on Evolutionary Computation* 26, 5 (2021), 802–822.

Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. 2021b. A review of population-based metaheuristics for large-scale black-box global optimization-Part II. *IEEE Transactions on Evolutionary Computation* 26, 5 (2021), 823–843.

Daniel Parrott and Xiaodong Li. 2006. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation* 10, 4 (2006), 440–458.

Zhou Peng, Jinhua Zheng, Juan Zou, and Min Liu. 2015. Novel prediction and memory strategies for dynamic multiobjective optimization. *Soft Computing* 19, 9 (2015), 2633–2653.

Frank Plastria. 2002. Continuous covering location problems. *Facility Location: Applications and Theory* 1 (2002), 37–79.

Bo-Yang Qu, Ponnuthurai Nagaratnam Suganthan, and Jane-Jing Liang. 2012. Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Transactions on Evolutionary Computation* 16, 5 (2012), 601–614.

Carlo Raquel and Xin Yao. 2013. Dynamic multi-objective optimization: A survey of the state-of-the-art. In *Evolutionary Computation for Dynamic Optimization Problems*. Springer, 85–106.

Hendrik Richter. 2009. Detecting change in dynamic fitness landscapes. In *IEEE Congress on Evolutionary Computation*. IEEE, 1613–1620.

Hongfeng Wang, Shengxiang Yang, W. H. Ip, and Dingwei Wang. 2012. A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems. *International Journal of Systems Science* 43, 7 (2012), 1268–1283.

Biao Xu, Yong Zhang, Dunwei Gong, Yinan Guo, and Miao Rong. 2017. Environment sensitivity-based cooperative co-evolutionary algorithms for dynamic multi-objective optimization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15, 6 (2017), 1877–1890.

Ming Yang, Mohammad Nabi Omidvar, Changhe Li, Xiaodong Li, Zhihua Cai, Borhan Kazimipour, and Xin Yao. 2017. Efficient resource allocation in cooperative co-evolution for large-scale global optimization. *IEEE Transactions on Evolutionary Computation* 21, 4 (2017), 493–505.

Shengxiang Yang and Changhe Li. 2010. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation* 14, 6 (2010), 959–974.

Delaram Yazdani. 2023. SPSO+AP+AD: A species-based PSO for dynamic optimization problems [Source Code]. https://codeocean.com/capsule/1977861/tree (2023). https://doi.org/10.24433/CO.3072519.v1

Danial Yazdani, Juergen Branke, Mohammad Nabi Omidvar, Xiaodong Li, Changhe Li, Michalis Mavrovouniotis, Trung Thanh Nguyen, Shengxiang Yang, and Xin Yao. 2021. IEEE CEC 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark. *arXiv: 2106.06174* (2021).

Danial Yazdani, Ran Cheng, Cheng He, and Juergen Branke. 2022. Adaptive control of subpopulations in evolutionary dynamic optimization. *IEEE Transactions on Cybernetics* 52, 7 (2022), 6476–6489.

Danial Yazdani, Ran Cheng, Donya Yazdani, Jürgen Branke, Yaochu Jin, and Xin Yao. 2021a. A survey of evolutionary continuous dynamic optimization over two decades – Part A. *IEEE Transactions on Evolutionary Computation* 25, 4 (2021), 609–629.

Danial Yazdani, Ran Cheng, Donya Yazdani, Jürgen Branke, Yaochu Jin, and Xin Yao. 2021b. A survey of evolutionary continuous dynamic optimization over two decades – Part B. *IEEE Transactions on Evolutionary Computation* 25, 4 (2021), 630–650.

Danial Yazdani, Babak Nasiri, Alireza Sepas-Moghaddam, and Mohammad Reza Meybodi. 2013. A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing* 13, 4 (2013), 2144–2158.

Danial Yazdani, Trung Thanh Nguyen, and Jürgen Branke. 2018. Robust optimization over time by learning problem space characteristics. *IEEE Transactions on Evolutionary Computation* 23, 1 (2018), 143–155.

Danial Yazdani, Mohammad Nabi Omidvar, Juergen Branke, Trung Thanh Nguyen, and Xin Yao. 2020. Scaling up dynamic optimization problems: A divide-and-conquer approach. *IEEE Transactions on Evolutionary Computation* 24, 1 (2020), 1–15.

Danial Yazdani, Mohammad Nabi Omidvar, Ran Cheng, Jürgen Branke, Trung Thanh Nguyen, and Xin Yao. 2022. Benchmarking continuous dynamic optimization: Survey and generalized test suite. *IEEE Transactions on Cybernetics* 52, 5 (2022), 3380–3393.

Xin Yu, Yaochu Jin, Ke Tang, and Xin Yao. 2010. Robust optimization over time-a new perspective on dynamic optimization problems. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–6.

Qian Zhang and Mahdi Mahfouf. 2009. A modified PSO with a dynamically varying population and its application to the multi-objective optimal design of alloy steels. In *IEEE Congress on Evolutionary Computation*. IEEE, 3241–3248.

Weiwei Zhang, Weizheng Zhang, Gary G. Yen, and HongLei Jing. 2019. A cluster-based clonal selection algorithm for optimization in dynamic environment. *Swarm and Evolutionary Computation* 50 (2019), 100454.

Received 27 February 2022; revised 4 June 2023; accepted 7 June 2023