Robust Optimization Over Time by Estimating Robustness of Promising Regions

Danial Yazdani, *Member, IEEE*, Donya Yazdani, Jürgen Branke, *Member, IEEE*, Mohammad Nabi Omidvar, *Senior Member, IEEE*, Amir Hossein Gandomi, *Senior Member, IEEE*, and Xin Yao, *Fellow, IEEE*

***Abstract*—Many real-world optimization problems are dynamic. The field of robust optimization over time (ROOT) deals with dynamic optimization problems in which frequent changes of the deployed solution are undesirable. This can be due to the high cost of switching the deployed solutions, the limitation of the needed resources to deploy such new solutions, and/or the system being intolerant toward frequent changes of the deployed solution. In the considered ROOT problems in this article, the main goal is to find solutions that maximize the average number of environments where they remain acceptable. In the state-of-the-art methods developed to tackle these problems, the decision makers/metrics used to select solutions for deployment mostly make simplifying assumptions about the problem instances. Besides, the current methods all use the population control components, which have been originally designed for tracking the global optimum over time without taking any robustness considerations into account. In this article, a multipopulation ROOT method is proposed with two novel components: 1) a robustness estimation component that estimates robustness of the promising regions and 2) a dual-mode computational resource allocation component to manage subpopulations by taking several factors, including robustness, into account. Our experimental results demonstrate the superiority of the proposed method over other state-of-the-art approaches.**

***Index Terms*—Computational resource allocation (CRA), evolutionary dynamic optimization (EDO), multipopulation, robust optimization over time (ROOT), robustness estimation.**

Manuscript received 13 August 2021; revised 3 January 2022, 10 March 2022, and 5 April 2022; accepted 20 May 2022. Date of publication 7 June 2022; date of current version 31 May 2023. This work was supported in part by the Research Institute of Trustworthy Autonomous Systems, Guangdong Provincial Key Laboratory under Grant 2020B121201001; in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X386; and in part by the Shenzhen Science and Technology Program under Grant KQTD2016112514355531. *(Corresponding author: Xin Yao.)*

Danial Yazdani is with the Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: danial.yazdani@gmail.com).

Donya Yazdani is with the Department of Computer Science, University of Sheffield, Sheffield S1 4DP, U.K. (e-mail: dyazdani1@sheffield.ac.uk).

Jürgen Branke is with the Operational Research and Management Sciences Group, Warwick Business School, University of Warwick, Coventry CV4 7AL, U.K. (e-mail: Juergen.Branke@wbs.ac.uk).

Mohammad Nabi Omidvar is with the School of Computing and Leeds University Business School, University of Leeds, Leeds LS2 9JT, U.K. (e-mail: m.n.omidvar@leeds.ac.uk).

Amir Hossein Gandomi is with the Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: gandomi@uts.edu.au).

Xin Yao is with the Research Institute of Trustworthy Autonomous Systems and the Guangdong Provincial Key Laboratory of Brain inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China, and also with the CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: xiny@sustech.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2022.3180590.

Digital Object Identifier 10.1109/TEVC.2022.3180590

I. Introduction

SEARCH spaces of many real-world optimization problems are dynamic. To tackle optimization problems in dynamic environments, it is important that the optimization algorithms can efficiently react to the environment changes [1], [2]. A dynamic optimization problem (DOP) can be defined as

$$f^{(t)}(\vec{x}) = f\left(\vec{x}, \vec{\alpha}^{(t)}\right) \tag{1}$$

where f is the objective function, $t \in [0, T]$ is the time index, $\vec{x}$ is a solution in the search space, and $\vec{\alpha}$ is a vector of time-dependent control parameters of the objective function. Almost all existing works in the field of DOPs consider problems whose environmental changes occur only at 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

$$\left\{f\left(\vec{x}, \vec{\alpha}^{(k)}\right)\right\}_{k=1}^{T} = \left\{f\left(\vec{x}, \vec{\alpha}^{(1)}\right), f\left(\vec{x}, \vec{\alpha}^{(2)}\right), \ldots, f\left(\vec{x}, \vec{\alpha}^{(T)}\right)\right\}. \tag{2}$$

Some main characteristics of a DOP are defined based on its change severity and frequency. Hence, these two are among the main criteria used for classifying DOPs [2]. The change severity shows how much the morphology of the problem space changes after each environmental change. In the DOP literature, it is mostly assumed that environmental changes are not highly severe and there is a degree of similarity between consecutive environmental states. This is the case for many practical applications [3]–[5]. In real-world DOPs, change frequency is defined based on the duration of the time interval between environmental changes which depends on the nature of the events that cause the environmental changes. In some problems, the duration of this time interval can be very short, which results in higher change frequency (e.g., the short time gaps between changing demands/customers in some dynamic covering location problems [6]). Change frequency is usually

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

very low (i.e., longer time gap between changes) for problems where environmental changes are caused by accidents or faults in parts of the system.

Evolutionary algorithms are commonly used for tackling DOPs [1], [5], [7]. However, since these algorithms are originally designed for solving static optimization problems, they cannot directly be used for DOPs. This is due to the challenges caused by the environmental changes in DOPs, which are: global and local diversity loss, outdated stored fitness values,[1] and limited number of fitness evaluations that can be performed in each environment (i.e., between two consecutive environmental changes)[2] [8]. In order to avoid influences of some factors, such as hardware, compiler, and programming skills, the academic community of evolutionary dynamic optimization (EDO) uses the number of fitness evaluations as the unit of change frequency instead of time in the real-world applications. In many real-world DOPs, the fitness evaluation is time consuming (e.g., in large-scale problems [9], or simulation optimization). Consequently, a limited number of fitness evaluations can usually be performed in each environment. To address the aforementioned challenges of DOPs, EDO algorithms are often created by augmenting evolutionary algorithms with other algorithmic *components* to address the DOPs' challenges stated above [2].

The majority of the existing EDOs tackle DOPs by tracking the moving global optimum after each environmental change [3], [10]. However, tracking is impractical for solving many real-world DOPs because frequent change of the deployed solution is not possible. This can be due to different reasons, such as high *switching cost* [11], [12], limited available resources to deploy new solutions, or system intolerance for frequent changes in the deployed solution [13], [14].

To solve this type of DOPs, Yu *et al.* [13] proposed an approach called robust optimization over time (ROOT). In ROOT, in order to reduce the number of times that the deployed solution is changed, it is reused (i.e., kept deployed) until its quality degrades to an unacceptable level. Therefore, although a deployed solution in an environment is not necessarily the best solution, it must satisfy a quality-based constraint [15]. In this type of ROOT problems, a new solution must be chosen for deployment when the current deployed solution is no longer acceptable after the last environmental change [10]. We wish to maximize the average number of environments where the previously deployed solutions can be reused and kept deployed [15], [16]. Later, two other types of ROOT problems were investigated: 1) time window-based [17] and 2) switching cost-based [12] ROOT problems. In this article, we focus on the first type, i.e., ROOT problems with a quality-based constraint [13]. Unless otherwise stated, we use the term ROOT to refer to this specific type of ROOT throughout this article.

To solve a ROOT problem, not only should an EDO be capable of addressing the challenges of reacting/responding to the environmental changes but also the challenges of estimating the robustness and acceptability of solutions in the forthcoming environments. A desirable solution in ROOT can remain acceptable for a higher number of environments in the future. Nevertheless, *accurate* estimation/prediction of the future acceptability of solutions is very challenging. Depending on how the existing ROOT methods deal with this challenge, they can be categorized into fitness prediction [15], [17] and promising regions' reliability-based approaches [10], [18]. In prediction-based methods, the actual fitness function is altered with a substitute fitness function that considers the predicted fitness values of the candidate solutions in the upcoming environments [15], [17]. It is shown in [18] that using the predicted future fitness values to find robust solutions is too error prone for problem instances generated by the moving peaks benchmark (MPB) [19]. Reliability-based methods, on the other hand, choose the solutions for deployment based on the estimated behavior of the promising regions instead of predicting the future fitness values of solutions [18], [20]. In these methods, some *reliable* promising regions are determined based on the gathered information by a multipopulation method. Thereafter, a solution is chosen for deployment from a reliable promising region based on a strategy, such as picking the best found solution in the reliable promising region with highest fitness[3] [20], or the best found solution in the promising region with the smallest estimated shift severity [18].

Despite the importance of ROOT in many real-world applications and more than a decade from the first time it was introduced [13], very little attention has been given to the field. A major weakness of the existing methods is that they are all tailored for very simple problems. For example, some of their components are specifically developed for low dimensions [16], simple dynamics [18], and/or regular/smooth search space/promising regions [15], [16], [18]. This creates a gap between the academic research and the real-world applications. Moreover, all existing ROOT methods use EDOs [10], [15], [16], which are originally designed for performing tracking the moving global optimum where robustness is not considered in their population management and control components. Despite the significant role of EDOs in the ROOT methods, little attention has been given to design EDO components which are ROOT-specific.

In this article, we propose two components that address the aforementioned shortcomings: 1) a *robustness estimation* component and 2) a *systematic dual-mode computational resource allocation (CRA)* component. By combining the proposed components with those of a multipopulation EDO capable of *tracking multiple moving promising regions*, a new ROOT method is formed. Thanks to the proposed robustness estimation component, the new ROOT method is not dependent on some oversimplified characteristics of the benchmark problems. This component keeps and transfers the historical knowledge about the covered promising regions and estimates their *robustness degrees* accordingly. The estimated robustness degree of each promising region is calculated based on the number of environments that previous promising region's summit positions could be reused/kept as the deployed solution

[1]Also called the *outdated memory* issue in the DOP literature.

[2]Also called limited available computational resources in each environment.

[3]In this article, maximization problems are considered.

until the current environment. The estimated robustness values of the promising regions are used for choosing the next solution for deployment and controlling the subpopulations. A systematic dual-mode robustness-based CRA component for managing the subpopulations is also proposed. Using this component, the proposed method controls the subpopulations based on the estimated robustness of the covered promising regions and the system status.[4]

The organization of the remainder of this article is as follows. Section II covers the related works. The proposed method is described in Section III. Section IV explains the experiment setup, including the used benchmark, performance indicator, comparison algorithms, and parameter settings, and also reports experimental results, comparisons, and analysis. Finally, Section V concludes this article.

II. Related Work

A ROOT method is usually constructed by assembling an EDO and some additional components, such as a decision maker [20] or a transformation of the objective function [15], [17]. The literature of EDOs is not covered in this article due to space limitations and the readers can refer to the two-part survey in [2] and [5]. Instead, we only focus on reviewing ROOT methods and the components specifically designed for them.

A. ROOT Problems

Three types of ROOT problems have been investigated in the literature: 1) ROOT problems based on acceptability of the deployed solution in which the deployed solution is kept as long as it remains acceptable [13]; 2) ROOT problems with time window in which the deployed solution is kept during each time window [17]; and 3) ROOT problems based on acceptability of the deployed solution and switching cost in which the deployed solution is kept until it becomes unacceptable or another solution is found whose fitness is considerably higher than that of the deployed solution, making the benefit of switching outweigh the cost [12]. As stated before, in this article, we focus on the first type of ROOT problems where the main objective is to minimize the number of times when the deployed solution is changed.

Given a DOP $f^{(t)}(\vec{x})$ with T environments, the aim of ROOT is to find a set of deployed solutions $\mathcal{S} = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_l\}$ where $1 \leq l \leq T$. The main objective in ROOT is to minimize l. A deployed solution $\vec{s}_i \in \mathcal{S}$ is considered *robust* if it remains *acceptable* across more than one environment. For a deployed solution $\vec{s}_i$ to be acceptable in the tth environment, $f^{(t)}(\vec{s}_i)$ must be greater than a predefined threshold μ [17]. Otherwise, a new solution must be chosen for deployment. This acceptability evaluation approach has been used to determine the acceptability of solutions in the majority of the works in the literature [11], [18], [20], [21].

[4]In the proposed method, the system can be in two different states: 1) the deployed solution is currently acceptable or a new solution for deployment has already been chosen and 2) the deployed solution is no longer acceptable and a new solution must be chosen for deployment before a deadline.

B. ROOT Methods

Despite the importance of ROOT in tackling many real-world DOPs, this field has not received much attention so far. To the best of our knowledge, there are only three main ROOT methods, which are proposed by Jin *et al.* [15], Fu *et al.* [17], and Yazdani *et al.* [18]. The rest of the works in the field are designed based on these three works [12], [22], [23].

Jin *et al.* [15] proposed the first ROOT method in 2013. This method uses the predicted fitness values of solutions in a predefined number of future environments to estimate their robustness. The main components of this method include a single-population EDO, a database, an approximator, and a predictor. The single population EDO is responsible for gathering data to train the predictor and also the optimization process. The historical data are archived in the database over time, which is used to train the approximation and prediction methods. Jin's method uses a substitute objective function, which is the accumulation of the actual fitness, predicted, and approximated values.

Fu *et al.* [17] proposed a survival time-based ROOT method whose main components are the same as Jin's method, however, the used substitute objective function is different

$$F^{(t)}(\vec{x}) = \begin{cases} 0, & \text{if } f^{(t)}(\vec{x}) < \mu \\ 1 + \max\left\{l' | \forall i \in \{1, 2, \ldots, l'\} : f'^{(t+i)}(\vec{x}) \geq \mu\right\}, & \text{otherwise} \end{cases} \tag{3}$$

where $f'^{(t+i)}$ is the prediction function that predicts the future fitness value of $\vec{x}$ in the $(t+i)$th environment. According to (3), if the fitness value of a solution in the current environment is less than μ, this solution is considered as a *nonrobust* solution. Otherwise, robustness value of this solution will be the number of successive future environments in which its fitness values are predicted to remain above μ.

Several multiobjective ROOT methods have been proposed to find robust solutions based on the substitute objective function in (3) and an additional objective function such as average fitness of the deployed solutions or switching cost. In [24], the algorithm tries to find a Pareto front based on the substitute objective functions of average fitness over a predefined time window and survival time. In [11], a multiobjective approach, called ROOT/SC, is proposed for optimizing two objectives, including maximizing the survival time (3) and minimizing the switching cost. The switching cost is defined as the Euclidean distance between the current deployed solution and a candidate solution. ROOT/SC was modified in [21] by adding the current fitness of candidate solutions as the third objective. Another group of multiobjective ROOT methods is designed to find robust Pareto optimal solutions [25]–[27]. In these methods, the main goal is to find Pareto optimal solutions whose performance is acceptable for the current and upcoming environments.

Yazdani *et al.* [18], [20] proposed a reliability-based ROOT method. Unlike the previous ROOT methods that search for robust solutions based on a substitute objective function, this method works on the search space constructed by the original objective function. The components include a multipopulation EDO capable of locating and tracking multiple moving

promising regions and a decision maker that chooses the solutions for deployment. The multipopulation EDO is responsible to gather information about the promising regions (e.g., shift severity and fitness fluctuation degree). Based on the gathered information, the reliability of each covered promising region is determined for choosing the next solution for deployment. To this end, first, the fitness fluctuation degree of each region covered by the ith subpopulation (pop_i) is calculated after each environmental change as follows:

$$\tau_i^{(t)} = \left| f^{(t-1)}\left(\vec{\mathbf{g}}_i^{*(t-1)}\right) - f^{(t)}\left(\vec{\mathbf{g}}_i^{*(t-1)}\right) \right| \tag{4}$$

where $\tau_i^{(t)}$ is the fitness fluctuation in the tth environmental change, and $\vec{\mathbf{g}}_i^{*(t-1)}$ is the best found position by pop_i in the $t-1$th environment. Thereafter, the average values of τ_i in the past environments ($\bar{\tau}_i$) are used for determining the reliability of the region using

$$\rho_i = \begin{cases} 1, \text{ if } f^{(t)}\left(\vec{\mathbf{g}}_i^{*(t)}\right) - \bar{\tau}_i \geq \mu \\ 0, \text{ otherwise.} \end{cases} \tag{5}$$

If the best found position $\vec{\mathbf{g}}_i^{*(t)}$ is reliable, it means that it is expected that its worst possible future fitness value will remain above the threshold μ for at least another environment. In the tth environment, if the previous deployed solution becomes unacceptable, the best found positions in the reliable promising regions (i.e., $\rho_i = 1$) are considered as a set of *candidate solutions* $\mathcal{C}$ for the next deployed solution. In [20], the solution j in $\mathcal{C}$ is picked for deployment using

$$j = \underset{i \in \mathcal{C}}{\text{argmax}} \left(f\left(\vec{\mathbf{g}}_i^{*(t)}\right) - \bar{\tau}_i \right) \tag{6}$$

where the solution from $\mathcal{C}$ that has the highest worst estimated future fitness value is chosen for deployment. Another strategy, used in [18], to choose a solution for deployment is as follows:

$$j = \underset{i \in \mathcal{C}}{\text{argmin}} \left(\frac{s'_i}{s'_{\max}} + \frac{h'_i}{h'_{\max}} \right) \tag{7}$$

where s'_i and h'_i are the estimated shift and height severity values of the promising region covered by the ith subpopulation, respectively, and $s'_{\max}$ and $h'_{\max}$ are the largest s' and h' values among reliable promising regions, respectively.

A major shortcoming of the reviewed methods in this section is that they are all tailored for very simple problems. For example, some of their components are designed for low numbers of dimensions, regular/smooth search space/promising regions, and/or simple dynamics. As described before, both Jin's and Fu's methods use approximation and prediction components to estimate the solutions' future fitness values. On the one hand, it is shown in [10] that using such approximation and prediction methods to estimate the future fitness values of solutions can be error prone. On the other hand, Yazdani's method depends on the accuracy of the estimated fitness fluctuations in (4). Until now, this method has only been tested on MPB whose promising regions are regular/smooth, without ill-conditioning, fully separable, and symmetric peaks [28], with fixed peak relocation length over time and homogeneous dynamics [2]. However, our investigations indicate that the effectiveness of determining reliability of the promising regions based on the estimated fitness fluctuations deteriorates where the problems are more challenging.

The last issue concerns the EDOs used in every ROOT methods to perform the optimization in dynamic environments. All existing ROOT methods use EDOs, which are originally designed for tracking the moving global optimum (Jin's and Fu's methods use a single-population EDO presented in [29], and Yazdani's method uses the multipopulation EDO from [30]). Despite the significant role of EDOs in the ROOT methods, little attention has been given into designing some components of EDOs, which take ROOT's considerations into account.

Algorithm 1: Procedure of the ROOT Method Constructed by Assembling the Proposed Components and Those of Multipopulation EDO

```
1  Initialize pop_1;
2  repeat
3      Choose sub-populations for execution by the proposed resource allocation
         (Alg. 3);
4      foreach chosen pop_i do
5          pop_i ← Optimizer(pop_i);
6      Execute population management and diversity control components;
7      if Environment has changed then
8          Estimate robustness of the covered promising regions (Alg. 2);
9          Execute change reaction components;
10     if a solution need to be deployed then
11         Choose the best found solution in the promising regions with the
             highest estimated robustness;
12 until stopping criterion is met;
```

III. Robustness Estimation and Computational Resource Allocation for ROOT

In this section, to address the shortcomings of the existing ROOT methods, we propose two new components.

1) A *robustness estimation* component that uses an explicit archive to keep and transfer historical knowledge about the covered promising regions for estimating their *robustness degrees*. The estimated robustness values of the promising regions are used for choosing the next solution for deployment and controlling the subpopulations.
2) A *systematic dual-mode robustness-based CRA* component that picks the subpopulations to run in each iteration in order to manage the consumption of the fitness evaluations. This is done according to several factors, including the system status, the estimated robustness of each covered promising region, and roles and task achievements of subpopulations.

We integrate the proposed components into a state-of-the-art multipopulation EDO to construct a ROOT algorithm. Algorithm 1 shows a high level procedure of the resulting ROOT method. As shown in this pseudocode, the proposed robustness estimation component is executed after each environmental change (line 8). The proposed resource allocation is executed at the beginning of each iteration (line 3) to pick the subpopulations to be run in the current iteration.

The high-level components of a multipopulation EDO can be classified into change reaction (line 9), optimization

(line 5), and population management and diversity controlling components (line 6), which usually include the mechanisms used for dividing the population into subpopulations, removing/randomizing redundant individuals/subpopulations, generating new subpopulations, and increasing/maintaining global diversity.[5] Finally, a decision maker is used (line 11) to choose a solution for deployment.

In the rest of this section, we describe the following.

1) Suitable multipopulation EDOs that can be equipped with the proposed components to form ROOT algorithms.
2) The proposed robustness estimation component.
3) The proposed CRA.

A. Compatible Multipopulation EDO

In ROOT, the main responsibility of the multipopulation EDO is not to find the global optimum but to locate and track multiple moving promising regions. Each subpopulation tracks and covers one promising region. The main purpose of tracking the promising regions for tackling ROOT problems is that the solutions around their summits are more likely to remain acceptable after environmental changes [14], [18], [31]. Using the proposed robustness estimation component's explicit archive, the historical information of the best found positions by each subpopulation in the previous environments is retrieved and used to estimate the robustness of each covered region.

Although many multipopulation EDOs have been developed for tracking the moving global optimum, not all of them are efficient at performing tracking multiple moving promising regions. For example, many state-of-the-art multipopulation EDOs, such as those that form the subpopulations by clustering the individuals based on their fitness and position [32], are defective for this purpose as they may lose track of the inferior (based on fitness) promising regions. Besides, those multipopulation EDOs, whose number of subpopulations and overall population size do not adapt to the discovered promising regions, are not suitable for constructing a ROOT method. Such EDOs are incapable of tracking multiple moving promising regions effectively in the problems whose number of promising regions is larger than the number of the EDO's subpopulations. In such problems, several promising regions cannot be covered due to the limited number of subpopulations.

A suitable class of multipopulation EDOs for performing a *stable and reliable* tracking of multiple moving promising regions over time are those whose number of subpopulations adapts to the number of discovered promising regions, where the membership of individuals in each subpopulation is fixed and determined based on the individuals' indices [8], [30], [33]. These multipopulation EDOs usually start with one subpopulation (Algorithm 1, line 1) and once it has converged to a promising region, a new subpopulation is initialized (Algorithm 1, line 6).

[5]The readers are referred to [2] for a detailed review of the population management and diversity controlling components used in multipopulation EDOs.

B. Proposed Robustness Estimation Component

The first component that is triggered right after an environmental change is the proposed robustness estimation whose main responsibility is to estimate the degree of robustness of the covered promising regions. Estimated robustness values are used for choosing solutions for deployment and also in the population control by the proposed resource allocation. The pseudocode of the proposed robustness estimation component is shown in Algorithm 2.

Each pop_i is equipped with an *explicit memory* $\mathcal{M}_i$, which is a *circular queue* of size $\mathfrak{m}_{\max}$. After each environmental change, the best found position by pop_i in the last environment ($\vec{\mathbf{g}}_i^{*(t-1)}$) is archived in $\mathcal{M}_i$ (line 1). $\mathfrak{m}_i$ is the number of archived solutions in $\mathcal{M}_i$. In the case in which the explicit archive is full (i.e., $\mathfrak{m}_i = \mathfrak{m}_{\max}$) the oldest archived solution is removed, then $\vec{\mathbf{g}}_i^{*(t-1)}$ will be archived.

After updating the explicit archive of all subpopulations, a robustness value γ is calculated for each covered promising region (lines 7–9). To this end, for each pop_i, γ_i is first reset to zero. Thereafter, the acceptability of the archived solutions in $\mathcal{M}_i$ in the current environment t is evaluated. We first evaluate $f^{(t)}(\mathcal{M}_{i,1})$. If $f^{(t)}(\mathcal{M}_{i,1})$ is acceptable [i.e., $f^{(t)}(\mathcal{M}_{i,1}) > \mu$], we increment γ_i by one. Then, we repeat this step for $\mathcal{M}_{i,2}$, which contains $\vec{\mathbf{g}}_i^{*(t-2)}$. This step is repeated until we either reach an archived solution, which is unacceptable or have evaluated all $\mathfrak{m}_i$ archived solutions in $\mathcal{M}_i$.

The calculation of γ is costly (i.e., it consumes fitness evaluations), in particular, when the number of discovered promising regions is large. To avoid wasting the computational resources for calculating γ, archived solutions are reevaluated *one-by-one* from the most recent to the oldest, and once an unacceptable archived solution is detected, the reevaluation process stops. Furthermore, after detecting an unacceptable archived solution, this solution and all older ones are removed from the explicit archive. Actually, when a historical best found solution in a promising region is not accepted in the current environment, it cuts off the chain of robustness in the successive environments. Consequently, we take no account of the older archived solutions in the robustness estimation. We remove the older archived solutions once the chain of robustness is cut off by an unacceptable archived solution (line 12). This *pruning mechanism* helps to reduce the burden of fitness evaluation consumption caused by using the robustness estimation component.

According to Algorithm 2, $\gamma_i \in \{0, 1, \ldots, \mathfrak{m}_i\}$. The value of γ_i indicates the number of successive environments for which if any of the previous best found positions by pop_i were deployed, they would have remained acceptable until the current environment. Besides, considering the removal of the unacceptable archived solutions in line 12, all existing archived best found solutions in the tth environment were acceptable from the environment that they were added to the archive until the current environment, i.e.,

$$f^{(t-j)}\left(\vec{\mathbf{g}}_i^{*(t-k-1)}\right) \geq \mu | j, k \in \{0, 1, \ldots, \gamma_i\} \wedge k \geq j. \quad (8)$$

For a pop_i, larger values of γ_i show that the promising region covered by this subpopulation had some characteristics during

Algorithm 2: Estimating the Robustness of pop_i (γ_i) and Managing the Explicit Archive $\mathcal{M}_i$

Input: $\mathcal{M}_i$ and $\vec{\mathbf{g}}_i^{*(t-1)}$.
Output: $\mathcal{M}_i$ and γ_i.

1 $\mathcal{M}_i \leftarrow \texttt{Push}(\vec{\mathbf{g}}_i^{*(t-1)})$;
2 $\mathfrak{m}_i \leftarrow \texttt{Update}(\mathfrak{m}_i)$;
3 $\gamma_i \leftarrow 0$;
4 $j \leftarrow t$;
5 **repeat**
6 $j \leftarrow j-1$;
7 $\vec{y} \leftarrow \texttt{Retrieve}(\vec{\mathbf{g}}_i^{*(j)} \in \mathcal{M}_i)$;
8 **if** $f^{(t)}(\vec{y}) \geq \mu$ **then**
9 $\gamma_i \leftarrow \gamma_i + 1$;
10 **until** $\gamma_i = \mathfrak{m}_i \vee f^{(t)}(\vec{y}) < \mu$;
11 **if** $\gamma_i < \mathfrak{m}_i$ **then**
12 $\texttt{Remove}(\{\mathbf{g}_i^{*(t-k)} \in \mathcal{M}_i | k > \gamma_i\})$;
13 $\mathfrak{m}_i \leftarrow \gamma_i$;

the recent γ_i environments that resulted in acceptability of the best found solutions over a larger numbers of the environmental changes. Larger values of γ_i can be due to several morphological and dynamical characteristics, such as smaller shift severity, smaller fitness fluctuation, and/or wider shape of the promising region. Therefore, larger values of γ_i indicate higher likelihood of robustness to environmental changes.

C. Decision-Making Process

A decision maker is responsible to choose a solution for deployment based on γ values calculated by the robustness estimation component (Algorithm 1, line 11). Indeed, γ values, which indicate robustness of the covered promising regions from past to the current environment, form the *historical knowledge* that is used by the decision maker for choosing solutions for deployment. These solutions are chosen from the promising regions with more reliable and suitable dynamical and morphological characteristics, which are likely robust to the upcoming environmental changes. To identify such promising regions, we use the gathered historical knowledge, i.e., γ values, to estimate their future robustness. We describe the relation between γ values, some dynamical and morphological characteristics of promising regions, and likelihood of future robustness in Section S-I in the supplementary material.

In the proposed ROOT method, the best found position in the promising region with the largest γ value is chosen for deployment. The purpose of choosing such a solution is to maximize the survival time, which is the number of successive environmental changes that the deployed solution can remain acceptable. Applying Algorithm 2 for the promising region covered by pop_i, the output indicates that during the last γ_i environments, a best found position in this region, which could be successfully deployed, is still reusable until the end of the current environment. Larger γ values for a promising region demonstrate that it has some dynamical and morphological characteristics, which make it more suitable for choosing solutions for deployment that are likely robust to the upcoming environmental changes.

Note that in the proposed robustness estimation component, the archived solutions are used to form the historical knowledge to estimate the future robustness of the promising regions and they are not candidates for deployment. Indeed, similar to [18], the proposed method focuses on robustness of promising regions, which differs from those ROOT methods proposed in [15] and [16], which focus on the robustness of all candidate solutions. This allows us to avoid the complexities and issues of using approximation and prediction components [10], [18], which are necessary for estimating the robustness of candidate solutions used in [15] and [16].

D. Proposed Computational Resource Allocation Component

In most existing multipopulation EDOs, a simple round robin/parallel method is used to allocate computational resources to subpopulations in each iteration [2]. Knowing that subpopulations do not necessarily share the same priority [2], [8], equal allocation of resources to all subpopulations is inefficient [2].

Herein, we propose a new CRA component, which works based on the estimated robustness of the covered promising regions, the role of the subpopulations, subpopulations' progress in their tasks, their age, and the current system status. The proposed resource allocation component uses three thresholds to identify the role of the subpopulations and measuring subpopulations' progress in their tasks.

1) r_{conv}: When the spatial size of a subpopulation is less than r_{conv}, it is assumed that is has converged to a promising region. Otherwise, it has not yet converged to any promising region and is still performing exploration. Such a mechanism is commonly used to identify converged subpopulations in EDOs [2], [34]. In this article, the spatial size λ_i of a subpopulation pop_i is defined as the Euclidean distance of the farthest pair of individuals [35], which is calculated by

$$\lambda_i = \max_{\vec{x}_j, \vec{x}_k \in \text{pop}_i} \|\vec{x}_j - \vec{x}_k\|. \tag{9}$$

2) r_{cover}: When the spatial size of a subpopulation drops below r_{cover}, we assume that it has converged to the promising region's *summit*. Note that $r_{\text{cover}} < r_{\text{conv}}$ and if $r_{\text{cover}} < \lambda_i < r_{\text{conv}}$, we assume that although pop_i has converged to the promising region, it is still climbing the promising region to reach the summit.
3) r_{min}: This threshold has the smallest value out of the three, i.e., $r_{\text{min}} < r_{\text{cover}} < r_{\text{conv}}$. When the spatial size of a subpopulation drops below r_{min}, it is assumed that the individuals of pop_i have collapsed on the summit of a promising region. In this circumstance, it is assumed that the tracking task has been fulfilled. Many EDOs use r_{min} to deactivate collapsed subpopulations [9], [36].

Algorithm 3 shows the details of the proposed resource allocation component. In each iteration, it determines which subpopulations are allowed to run and use computational resources (Algorithm 1, line 3). As can be seen in Algorithm 3, the proposed resource allocation component is composed of two operational modes, which are selected based on the current

Algorithm 3: Selecting Subpopulations to Run in the Current Iteration

Input: t, $f^{(t)}(\vec{\mathbf{s}})$, and γ and λ of all sub-populations.
Output: $\mathcal{L}$.

```
1  L ← ∅; // Create an empty set L
2  if t = 1 then // For the first environment
3      foreach {pop_i | λ_i > r_min} do
4          L ← L ∪ i;
5  else
6      γ_max = max_{∀ pop_i}(γ_i);
7      if f^(t)(s⃗) < μ ∧ {∃ pop_i | r_min < λ_i ∧ γ_i = γ_max} then
8          Mode ← Quick recovery;
9      else
10         Mode ← Normal;
11     if Mode = Normal then
12         foreach {pop_i | r_min < λ_i ≤ r_cover} do
13             Calculate p_i using (10);
14             if U[0, 1] ≤ p_i then
15                 L ← L ∪ i;
16         foreach {pop_i | λ_i > r_cover} do
17             L ← L ∪ i;
18     else if Mode = Quick recovery then
19         foreach {pop_i | r_min < λ_i ∧ γ_i = γ_max} do
20             L ← L ∪ i;
```

system status: 1) normal (lines 11–17) and 2) quick recovery (lines 18–20).

1) *Normal Mode:* In the environments in which the previously deployed solution is still acceptable/reusable, or the new solution for deployment has been chosen.
2) *Quick Recovery Mode:* When the previously deployed solution is no longer acceptable and a new solution must be chosen for deployment.

In the following, we describe these two modes.

1) Normal Mode: In the normal mode, the proposed resource allocation prioritizes the following subpopulations.

a) Subpopulations with larger γ values: These subpopulations are very important since they are tracking the promising regions that likely contain high quality robust solutions. Consequently, by allocating more computational resources to such subpopulations, their exploitation capability will be accelerated and improved.

b) Subpopulations with unfinished exploration/exploitation tasks: The proposed resource allocation component also considers relatively young subpopulations whose γ values are small, but they are important for locating and covering promising regions. A promising region can be considered properly covered when a subpopulation exploits around its summit and tracks it. The last generated subpopulation is the one that is responsible for the vital task of performing exploration in the multipopulation method. This subpopulation fulfills its task once it has converged to a promising region. Besides the explorer subpopulation, the ones that have lately converged to the promising regions, but still have not got close to the summit, are also prioritized by the proposed resource allocation component. The main task of each of these subpopulations is to exploit the promising region and getting close to its summit. Assigning computational resources to such subpopulations to fulfill their current tasks is crucial for providing more accurate information for the robustness estimation component. Besides, the lack of prioritizing such subpopulations may even result in losing the coverage of the newly discovered promising regions.

Allocating computational resources to converged subpopulations is a waste of limited resources. For this reason, resource allocation component stops optimizing subpopulations with lost local diversity. This happens irrespective of the resource allocation component mode, environment number, and γ values. In the resource allocation component, when the spatial size of a subpopulation becomes smaller than $r_{\min}$, it does not get selected until its spatial size is increased by the local diversity control component after each environmental change [2], [10].

In the first environment, the round robin method is used to allocate an identical amount of computational resources to all subpopulations (whose spatial size is larger than $r_{\min}$) in each iteration since in the first environment, $\gamma = 0$ for all subpopulations (Algorithm 3, lines 2–4). After the first environmental change, in each environment where either the previous deployed solution is still acceptable or a new solution has been chosen for deployment, the resource allocation component in normal mode is activated. Below, we describe this process.

In all iterations, the resource allocation component selects the subpopulations with unfinished exploration/exploitation tasks to run in the current iteration. The spatial size of the the last generated subpopulation is always larger than r_{conv} since right after its spatial size falls under the threshold, a new subpopulation is initialized. Therefore, the resource allocation component selects any subpopulation whose spatial size is larger than r_{conv}. Besides, the subpopulations that have recently converged to the basin of attraction of a promising region and are not yet close to the summit are selected. These subpopulations are the ones whose spatial sizes are less than r_{conv} but more than a second threshold r_{cover}. When the spatial size of a subpopulation falls under r_{cover}, we assume that it has converged to the promising region's summit and fulfilled its previous task. Since $r_{\text{cover}} < r_{\text{conv}}$, the proposed resource allocation component's normal mode selects all subpopulations with spatial sizes larger than r_{cover}, which are assumed to be not finished with their exploration/exploitation tasks, to run in the current iteration.

The main tasks of the subpopulations with spatial sizes less than r_{cover} include tracking the optimum of the promising region and providing a history of the optimum position in each environment for the robustness estimation component. It is expected that after initial environments and due to prioritizing the subpopulations that have not converged to the summits yet, most existing subpopulations will become tracker subpopulations whose spatial sizes are less than r_{cover}. The reason is that the number of the promising regions in the DOPs considered in this article—and in all existing works—is not excessively high to hinder performance [2]. Consequently, it is expected that after a while, most of the promising regions will be covered by subpopulations that are residing around their summits. Among these subpopulations, the proposed resource

allocation component prioritizes those with larger γ values using a probability-based selection process. To this end, at the beginning of each iteration, the resource allocation component assigns a probability p_i to each pop_i, which satisfies $r_{\min} < \lambda_i \leq r_{\text{cover}}$, i.e., the subpopulations participating in the selection process are the ones whose spatial sizes are larger than $r_{\min}$ and smaller than r_{cover}. This probability is calculated as follows:

$$p_i = \frac{\gamma_i}{\max_{\left\{\text{pop}_j \,|\, r_{\min} < \lambda_j < r_{\text{cover}}\right\}} \gamma_j}. \tag{10}$$

Afterward, a random number is generated in $[0, 1]$ with uniform distribution for each of these pop_i and if this number is less than p_i, then pop_i will be selected by the resource allocation component to execute an *internal iteration*. In other words, after determining the probability value for each subpopulation using (10), a selection process is *independently* run for each subpopulation, which decides whether it is chosen to run in the current iteration or not. Since the probability of selecting the subpopulation with the highest robustness value among the participating subpopulations in the selection process is 1.0 using (10), it will definitely be selected. Other participating subpopulations also are selected with probability p_i. Thus, at least one subpopulation and at most all participating subpopulations might be chosen. According to this probability-based selection process, the superior subpopulations, i.e., the ones with larger robustness values, have more chance to be selected in each iteration. Note that although the inferior subpopulations are less likely to be selected, they still have a chance to be selected and perform tracking. Moreover, the more frequent the superior subpopulations are selected, the quicker they will be omitted in the selection process due to loss of diversity (once their spatial sizes have shrunk to less than $r_{\min}$). Consequently, by excluding the subpopulations with larger γ values, the selection probabilities for inferior subpopulations increase.

2) Quick Recovery Mode: When the algorithm responses to an environmental change, if the deployed solution is no longer acceptable, the proposed resource allocation component switches to the quick recovery mode. This mode is designed to ensure that the full potential of the algorithm in finding a better solution for deployment is used, in particular, when the change frequency is high or there is a deadline (temporal constraint) for deploying a new solution [3]. As stated in Section III-C, the best found solution in the promising regions with the highest γ value is chosen for deployment. To this end, we first identify the highest value of robustness among all subpopulations as $\gamma_{\max} = \max_{\{\forall \text{pop}_i\}}(\gamma_i)$. Then, we identify the subpopulations whose $\gamma_i = \gamma_{\max}$ and compare their best found solutions. The one with the highest fitness value is chosen for deployment (see Algorithm 4, lines 24 and 25).

In this mode, *only* the subpopulations whose $\gamma_i = \gamma_{\max}$ run in all iterations while all other subpopulations are hibernated. The resource allocation component remains in this mode until:

1) the spatial sizes of the running subpopulations become less than $r_{\min}$ (i.e., sufficient exploitation has been performed);
2) the computational budget has run out before the deadline.

When one of the aforementioned conditions is met, the resource allocation component will switch back to the normal mode immediately.

Algorithm 4: Instantiation of the Proposed ROOT Method

```
1  Initialize pop_1;
2  repeat
       // Executing the proposed computational resource
          allocation component
3      Obtain list L using Alg. 3;
       // Execute the optimization component of EDO
4      foreach pop_i ∈ L do
5          pop_i ← Optimizer(pop_i);
       // Execute population management and diversity
          control components of the EDO
6      foreach {pop_i & pop_j | i ≠ j} do // Exclusion component
7          if ||x*_i − x*_j|| < φ_excl then
8              Remove the inferior sub-population;
9      foreach pop_i do // Spatial size calculation
10         Update λ_i by Eq. (9);
11     if {∄ pop_i | λ_i > r_conv} then
12         Initialize a new sub-population;
13     if Environment has changed then
14         foreach pop_i do
15             Age_i = Age_i + 1;
16             Calculate ŝ_i by Eq. (S-3);
               // Executing the proposed robustness
                  estimation component
17             foreach {pop_i | Age_i > 1} do
18                 Calculate γ_i by Alg. 2;
           // Executing change reaction components of the
              EDO
19         foreach {pop_i | λ_i < r_conv} do
20             Re-diversify by Eq. (S-2);
21         foreach pop_i do
22             Update stored fitness values in the new environment;
23     if a solution need to be deployed then
24         γ_max = max_{∀ pop_i}(γ_i);
25         Deploy x*_i from {pop_i | i = argmax_{pop_j | γ_j = γ_max}(f^(t)(x*_j))};
26 until stopping criterion is met;
```

E. Detailed Description of an Instantiation of the Proposed ROOT Method

Algorithm 4 shows how the proposed robustness estimation and resource allocation components and those of a multipopulation EDO are assembled to form an instantiation of the proposed ROOT method. Herein, we choose a simple, yet very efficient, multipopulation EDO framework from [9] and [10], which is described in Section S-II in the supplementary material. A complexity analysis of the instantiation of the proposed ROOT method shown in Algorithm 4 is provided in Section S-III in the supplementary material.

IV. Experiments and Analysis

In this section, we first describe the experimental design. Then, we investigate the effectiveness of the proposed robustness estimation and CRA components. We finally compare the performance of a multipopulation EDO equipped with the proposed components and several peer ROOT algorithms.

A. Experimental Design

1) Benchmark Generator: The experiments in this article are based on various problem instances generated by the generalized MPB (GMPB) [28], [37]. GMPB is a benchmark generator, which is capable of generating landscapes with a controllable number of promising regions with a variety of parametric and changeable characteristics, including symmetry, condition number, irregularity, roughness, modality, and variable interaction. GMPB has several parameters (see Table I) that can be set by the user to generate problem instances with a vast variety of morphological and dynamical characteristics and difficulty levels. Detailed information of the GMPB used in this article is provided in Section S-IV in the supplementary material. The MATLAB source code of this benchmark is available from [38].

2) Performance Indicator: The focus of this article is on solving ROOT problems in which the main goal is to find solutions for deployment in order to maximize the average number of environments that the deployed solutions remain acceptable. It is worth to mention that *none* of the algorithms examined in this article, including the proposed one, are designed to maximize the fitness of deployed solutions or minimize the switching cost, which are related to other classes of ROOT problems or can be considered as other objectives [11], [12], [17]. Considering the focus of this article, we compare the performance of the algorithms according to the *average survival time* [17], which is the most commonly used performance indicator in the field [16], [18].

3) Algorithms: In the experiments, we use the multipopulation framework from [10] for all multipopulation-based methods, including the proposed one. We also use PSO with a constriction factor [39] as the optimization component in the multipopulation framework. To evaluate the effectiveness of proposed resource allocation and robustness estimation components, we add them to mPSO. This approach is denoted as $\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$ in the experiments.

Besides $\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$, we also use three other multipopulation comparison algorithms, which are: mPSO, mPSO_{b}, and mPSO_{sh}. mPSO is a tracking the moving global optimum method that is adapted to tackle ROOTs in which the best found solution (in terms of fitness) is chosen for the next solution for deployment. mPSO_{Rb} uses the ROOT decision maker from [20], where the best found solution among the *candidate reliable promising regions* is chosen for deployment. mPSO_{Rsh} applies (7) (for which estimated shift and height severity values are needed) to choose the next solution for deployment [18].

We also choose the fitness prediction-based ROOT methods from [17] (denoted as PbM_{F}) and [15] (denoted as PbM_{J}) as comparison algorithms. For the experiments, we assume that these methods have access to the previous environmental parameters; thus, they do not need any approximation method. It should be noted that the fitness evaluations used for training the predictor in the previous environments are not counted toward the overall computational cost. This means that the results obtained by these algorithms are not affected by the approximation error and can therefore be taken as an upper bound for what the algorithms are capable of achieving in practice.

TABLE I
PARAMETER SETTINGS OF THE GMPB. THE DEFAULT PARAMETER VALUES ARE HIGHLIGHTED WHERE SEVERAL VALUES ARE USED IN OUR EXPERIMENTS. THE MATLAB SOURCE CODE OF THE USED GMPB CAN BE ACCESSED FROM [38]

Parameter	Symbol	Value(s)
Acceptability ROOT threshold	μ	40,45,50
Dimension	d	5,10
Numbers of promising regions	m	10,25,50,100
Shift severity	$\tilde{s}_i$ [†]	$\mathcal{U}[1,3]$,$\mathcal{U}[1,5]$,$\mathcal{U}[1,7]$ [‡]
Change frequency	ϑ	250 ,500,1000,2500
Computational budget [⋆]	δ	$\lfloor \frac{\vartheta}{3} \rfloor$, $\lfloor \frac{\vartheta}{2} \rfloor$
Height severity	$\tilde{h}_i$	$\mathcal{U}[1,15]$
Width severity	$\tilde{w}_i$	$\mathcal{U}[0.1,1.5]$
Irregularity parameter τ severity	$\tilde{\tau}$	0.05
Irregularity parameter η severity	$\tilde{\eta}$	2
Angle severity	$\tilde{\theta}$	$\pi/9$
Search range	$[Lb, Ub]^d$	$[-50, 50]^d$
Number of environments	T	100

[†] The value of a given parameter X in the ith promising region is shown by X_i. In the case a parameter value is the same for all the promising regions or it is a temporal parameter, no subscript is used.
[‡] $\mathcal{U}[a, b]$ denotes a uniform distribution used to uniformly draw random numbers in the range $[a, b]$.
[⋆] It indicates the deadline for deploying a new solution when the previous one is no longer acceptable, which can be found in many real-world DOPs [3].

In this article, we focus on DOPs with *visible* environmental changes where the optimization algorithms are informed about environmental changes by other parts of the system, such as sensors and agents, similar to many real-world DOPs [2]. Therefore, the examined algorithms in this article do not use any change detection component. If needed, a simple reevaluation-based change detection component [40] could be added to the algorithms.

4) Parameter Settings: Table I shows the parameter settings of GMPB. The experiments are done on the problem instances with various numbers of promising regions (m), acceptability threshold (μ), and dimensions (d). The parameter settings chosen in Table I are commonly used in the ROOT and EDO literature. Different parameter settings result in problem instances with different difficulty levels. In addition, different parameter settings generate problem instances with different maximal robustness values for both problems and promising regions. By increasing μ, the regions containing the robust solutions shrink, the maximal possible survival time decreases, and finding robust solutions becomes more challenging [16]. The higher the number of promising regions m, the easier to find robust solutions. This is due to the fact that in landscapes with larger numbers of the promising regions, the size/number of areas containing robust solutions with better qualities increases [18]. In addition, in problems with larger m values, the promising regions are likely to overlap and support the deployed solution to remain acceptable for further environments. Finally, by increasing the dimension, the problem instances become more challenging.

We use the parameter settings extracted from the sensitivity analysis in [10] for our multipopulation EDO framework and also for PSO in the multipopulation methods. Besides, the parameter settings of the proposed components are taken

TABLE II
PARAMETER SETTINGS OF THE COMPONENTS OF $\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$. THE MOST RIGHT COLUMN INDICATES WHETHER THE PARAMETER SETTINGS ARE TAKEN FROM THE ORIGINAL REFERENCES OR FROM THE SENSITIVITY ANALYSIS RESULTS REPORTED IN SECTION S-V IN THE SUPPLEMENTARY MATERIAL. NOTE THAT VALUES OF OTHER PARAMETERS OF THE PROPOSED COMPONENTS, γ_i AND $\gamma_{\max}$, ARE CALCULATED IN EACH ITERATION BY ALGORITHM 2 AND LINE 24 OF ALGORITHM 4, RESPECTIVELY

Method	Parameter	Value	Reference
PSO	χ	0.729843788	[10]
	C_1, C_2	2.05	[10]
	Neighbourhood topology	global star	[10]
	Sub-population size	5	[10]
mEDO framework*	φ_{excl}	$0.5\frac{Ub-Lb}{\sqrt[d]{p}}$ †	[9], [41]
	r_{conv}	$0.5\frac{Ub-Lb}{\sqrt[d]{p}}$	[9], [41] and Table S-II
Proposed components	$\text{m}_{\max}$	9	Table S-I
	r_{cover}	5	Fig. S-4
	$r_{\min}$	0.75	Fig. S-4

* The multi-population EDO framework used in this paper is described in Section S-II of the supplementary document.
† Where p is the number of sub-populations.

from the sensitivity analysis reported in Section V in the supplementary material. Parameter settings of $\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$ are summarized in Table II. For the parameter settings of the comparison algorithms, the values suggested in their original references are used. Our investigations also indicate that these algorithms show their best efficiency with those suggested settings.

B. Experimental Results

The statistical results provided in this section are based on 31 independent runs. For statistical analysis, we use multiple comparison tests using the Wilcoxon rank-sum test with Holm–Bonferroni p-value correction and $\alpha = 0.05$.

1) Effect of the Proposed Components on Performance: In this section, we investigate the effectiveness of the proposed robustness estimation and CRA components. To this end, we compare the performance of mPSO, mPSO with robustness estimation component ($\text{mPSO}_{+\text{RE}}$), and mPSO with both proposed robustness estimation and CRA components ($\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$). Fig. 1(a) compares the average survival time over time by these three methods on GMPB with the default parameter settings from Table I. We also compare the average percentage of the previously deployed solutions that are reused (i.e., remaining acceptable) by these methods in each environment in Fig. 1(b). Comparing the performance of mPSO and $\text{mPSO}_{+\text{RE}}$ in these plots based on average survival time and the average percentage of acceptability of the previously deployed solutions demonstrates the effectiveness of this component in choosing more robust solutions for deployment.

As can be seen in Figs. 1(a) and (b), adding the proposed CRA component further improves the performance of the algorithm. This improvement is a result of systematic control of the computational resource consumption by each subpopulation according to their estimated robustness value (provided by the proposed robustness estimation component), role, task achievement, and current system status. To further investigate

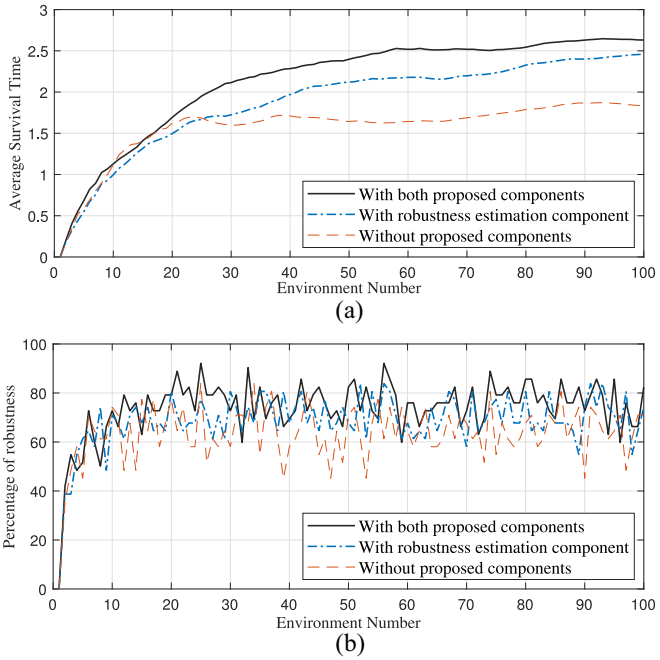

Fig. 1. Effects of the proposed robustness estimation and computation resource allocation components on the performance of mPSO for GMPB with the default parameter settings from Table I. This figure compares the efficiency of simple mPSO, mPSO with robustness estimation component ($\text{mPSO}_{+\text{RE}}$), and mPSO with both proposed robustness estimation and computation resource allocation components ($\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$). The plots are obtained by averaging the results of 31 independent runs. (a) Average survival time over time plot. (b) Average percentage of the previously deployed solutions which are reused (i.e., remaining acceptable) in each environment. For the ith environment, the value plotted in this figure is obtained by $[(\sum_{b=1}^{\hat{b}} a_{i,b})/\hat{b}] \times 100$ where $\hat{b}$ is the total number of runs and $a_{i,b} \in \{0, 1\}$ shows the acceptability of the last deployed solution in the ith environment of bth run. $a_{i,b} = 1$ indicates that the last deployed solution is still acceptable and reused in the ith environment, and it is zero otherwise.

the effectiveness of the proposed resource allocation, we compare the performance of $\text{mPSO}_{+\text{RE}}$ and $\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$ on the problem instances generated by GMPB with different numbers of the promising regions and the default settings for the remaining parameters. The results are compared in Fig. 2. As can be seen in this figure, by adding the proposed resource allocation component, the performance is improved, in particular, in the problem instances with larger numbers of the promising regions. In such problems, larger numbers of subpopulations are generated to cover the promising regions. Thus, the role of resource allocation becomes more vital since larger numbers of the fitness evaluations are needed in each iteration. $\text{mPSO}_{+\text{RE}}$ suffers from a shortage of the available computational resources before the deployment time. However, for the $\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$, this challenge is ameliorated by systematic allocation of the computational resources to the subpopulations with higher robustness and taking the system status into account.

As described in Section III-D, besides prioritizing the subpopulations with larger γ values, the proposed CRA component also prioritizes the subpopulations, which are performing exploration or recently have converged to a promising region and are moving toward the summit. This systematic

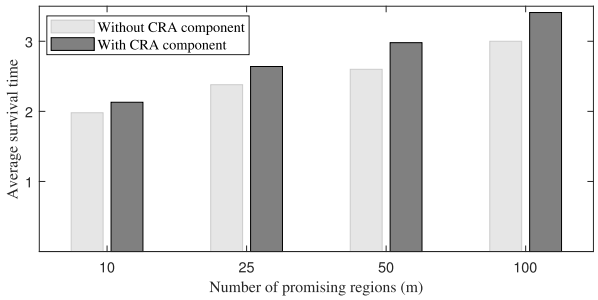

Fig. 2. Investigating the effect of the proposed CRA by comparing the obtained average survival time by $\text{mPSO}_{+\text{RE}}$ and $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$ in problem instances generated by GMPB with different numbers of promising regions (m) and the default parameter settings from Table I for the rest of the parameters.

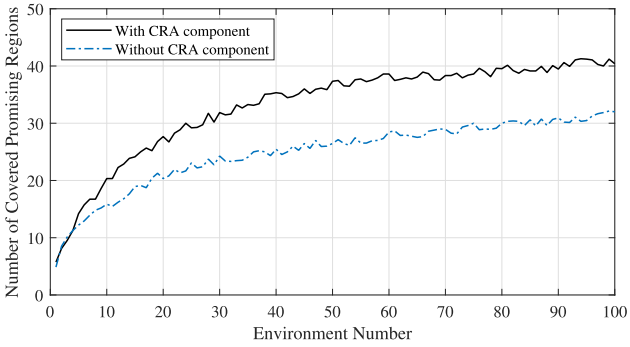

Fig. 3. Investigating the effect of the proposed CRA by comparing the ability of finding and covering/tracking promising regions in $\text{mPSO}_{+\text{RE}}$ and $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$ over 100 environments of GMPB with 50 promising regions ($m = 50$) and the default parameter settings from Table I for the rest of the parameters. The plots are obtained by averaging the results of 31 independent runs. Note that usually some smaller promising regions are covered by larger ones, thus, the number of visible promising regions is usually less than m. Besides, by changing the size and location of promising regions, the number of visible promising regions changes over time [28].

prioritizing approach considerably improves the abilities of exploration, exploitation, and tracking, which results in discovering and covering larger numbers of promising regions. Fig. 3 shows the effect of using the proposed resource allocation in the ability of the algorithm in finding and covering/tracking promising regions over time. Note that a promising region is considered covered if there is a subpopulation whose individuals reside in the basin of attraction of the promising region. As can be seen in this plot, thanks to the proposed resource allocation component, $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$ covers larger numbers of promising regions in comparison to $\text{mPSO}_{+\text{RE}}$. By covering larger numbers of promising regions in $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$, the possibility of missing promising regions, which may contain solutions with higher robustness, decreases that in turn results in improving the performance of the algorithm in finding better robust solutions.

2) Comparison With Peer Algorithms: In this section, we compare the results obtained by $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$ and the peer algorithms described in Section IV-A3 in solving problem instances generated by GMPB with different dimensions d, numbers of promising regions m, and acceptability threshold values μ. The average survival time (and standard error) obtained by the algorithms is reported in Table III, where

TABLE III
AVERAGE SURVIVAL TIME (AND STANDARD ERROR) ON GMPB WITH DIFFERENT d, μ, AND m VALUES. THE HIGHLIGHTED ENTRIES ARE SIGNIFICANTLY BETTER USING WILCOXON RANK-SUM TEST WITH HOLM–BONFERRONI p-VALUE ADJUSTMENT ($\alpha = 0.05$)

d	μ	m	Algorithms					
			$\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$	mPSO	mPSO_{Rb}	mPSO_{Rsh}	PbM_{F}	PbM_{J}
5	40	10	2.14(0.25)	1.80(0.30)	1.90(0.33)	1.88(0.30)	0.14(0.03)	0.66(0.15)
		25	2.63(0.21)	1.83(0.18)	2.23(0.22)	2.21(0.24)	0.26(0.04)	1.10(0.24)
		50	2.99(0.16)	2.01(0.11)	2.35(0.19)	2.36(0.20)	0.33(0.05)	1.08(0.18)
		100	3.40(0.21)	2.42(0.26)	2.71(0.23)	2.65(0.26)	0.32(0.03)	1.03(0.17)
	45	10	1.41(0.12)	0.98(0.11)	1.01(0.11)	1.09(0.12)	0.07(0.02)	0.45(0.12)
		25	1.70(0.12)	1.23(0.15)	1.35(0.16)	1.40(0.16)	0.09(0.03)	0.63(0.13)
		50	1.90(0.10)	1.30(0.11)	1.48(0.14)	1.55(0.13)	0.07(0.01)	0.75(0.10)
		100	2.15(0.15)	1.52(0.16)	1.78(0.16)	1.74(0.15)	0.17(0.02)	0.71(0.11)
	50	10	0.85(0.09)	0.54(0.08)	0.62(0.09)	0.57(0.08)	0.01(0.00)	0.21(0.07)
		25	1.10(0.09)	0.70(0.10)	0.89(0.11)	0.85(0.10)	0.06(0.02)	0.29(0.07)
		50	1.27(0.07)	0.86(0.06)	1.02(0.09)	0.99(0.07)	0.06(0.02)	0.41(0.05)
		100	1.46(0.10)	0.98(0.11)	1.18(0.13)	1.15(0.10)	0.09(0.03)	0.35(0.09)
10	40	10	1.06(0.08)	0.82(0.09)	0.83(0.09)	0.83(0.09)	0.07(0.02)	0.38(0.07)
		25	1.35(0.12)	1.03(0.11)	1.04(0.11)	1.01(0.10)	0.07(0.03)	0.45(0.11)
		50	1.27(0.13)	0.92(0.15)	0.94(0.15)	0.96(0.15)	0.07(0.02)	0.41(0.14)
		100	1.60(0.18)	0.90(0.14)	0.98(0.14)	0.97(0.14)	0.11(0.03)	0.56(0.16)
	45	10	0.65(0.06)	0.42(0.05)	0.47(0.06)	0.46(0.05)	0.02(0.01)	0.21(0.05)
		25	0.80(0.08)	0.51(0.07)	0.56(0.07)	0.56(0.07)	0.02(0.01)	0.25(0.07)
		50	0.82(0.09)	0.49(0.10)	0.52(0.10)	0.53(0.10)	0.06(0.01)	0.25(0.10)
		100	0.91(0.08)	0.47(0.05)	0.47(0.05)	0.47(0.05)	0.06(0.01)	0.33(0.09)
	50	10	0.30(0.02)	0.18(0.02)	0.19(0.02)	0.18(0.02)	0.02(0.01)	0.10(0.02)
		25	0.37(0.04)	0.23(0.03)	0.25(0.04)	0.26(0.03)	0.00(0.00)	0.12(0.03)
		50	0.40(0.06)	0.19(0.05)	0.21(0.05)	0.20(0.05)	0.00(0.00)	0.10(0.06)
		100	0.38(0.05)	0.18(0.03)	0.18(0.03)	0.18(0.03)	0.00(0.00)	0.13(0.04)

the best results are highlighted according to the performed statistical analysis. The reported results in this table clearly demonstrate that $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$ performs significantly better than all comparison algorithms in all cases, thanks to the proposed robustness estimation and CRA components. The robustness estimation component used in $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$ does not rely on any estimated parameter values related to the dynamical characteristics of the promising regions, such as fitness fluctuation (used in both mPSO_{Rsh} and mPSO_{Rb}), and/or shift and height severity values (used in mPSO_{Rsh}). As a result, $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$ does not suffer from inaccuracy in estimating these values, especially, in the DOPs with random and/or heterogeneous dynamics [2] where estimating these parameter values is error prone. Besides, additional morphological and dynamical characteristics of the promising regions are also implicitly considered in the calculation of the estimated robustness values in $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$, which are not taken into consideration for choosing solutions in mPSO_{Rsh} and mPSO_{Rb}.

The proposed CRA component is another major feature of $\text{mPSO}_{+\text{RE}}^{+\text{CRA}}$. The used resource allocation methods in mPSO_{Rsh}, mPSO_{Rb}, and mPSO are designed for tracking the moving global optimum [10], and they do not consider any attribute related to the robustness of promising regions in controlling subpopulations. On the other hand, the proposed resource allocation is particularly designed for tackling ROOT problems, which controls the subpopulations based on several factors, including the robustness of the promising regions

and the acceptability of the deployed solution. Besides, the proposed resource allocation takes the roles of subpopulations, their convergence status, and their task achievements into consideration.

To further investigate the performance of $\text{mPSO}^{+\text{CRA}}_{+\text{RE}}$ in problem instances with different characteristics, we carry out additional experiments on the problem instances generated by GMPB with different acceptability ROOT threshold values, change frequencies, shift severity degrees, and computational budget values. The results are reported in Section S-VI in the supplementary material. Moreover, to show the independence of the proposed components with respect to the optimization component used, we compare the performance of the multipopulation methods when they use differential evolution (DE) [42] as the optimization component in Section S-VII in the supplementary material. The results indicate the superiority of the proposed ROOT method when DE is used as the optimization component.

V. Conclusion

In this article, we have presented two new components—robustness estimation and dual-mode CRA—for ROOT methods. The robustness estimation component is responsible for estimating the robustness degree of the covered promising regions, while the systematic dual-mode robustness-based CRA component controls the subpopulations. These two components and those of a multipopulation EDO, which is capable of tracking multiple moving promising regions, are assembled to form a new ROOT method. Unlike the existing components for determining robust solutions, the performance of the proposed robustness estimation component does not rely on the oversimple characteristics of the benchmark problems, such as fixed relocation length of the promising regions, unimodality and smoothness of the promising regions, and/or low dimensionality of the problem. Moreover, using the proposed dual-mode CRA component, the proposed ROOT method is the first one that takes robustness and the system status into account for controlling the subpopulations. The proposed ROOT method and a set of peer methods have been used for maximizing the average survival time of the deployed solutions in 48 different problem instances with various characteristics generated by GMPB. The experimental results have shown the superiority of the proposed method in almost all test cases.

In our algorithm, we monitor the spatial size of subpopulations to determine their role and also progress in carrying out their tasks, where we used a simple widely used method [35] for calculating the spatial size of subpopulations. Designing a more advanced and systematic spatial size monitoring method that takes some problem characteristics, such as ill-conditioning and asymmetry into account, is a potential future research direction.

In the proposed robustness estimation component, the value of $\mathfrak{m}_{\max}$ should be set considering the scope of robustness in the problem. In this article, the value of $\mathfrak{m}_{\max}$ is fixed over time, which may not be efficient for solving *heterogeneous* DOPs [2] in which dynamical behavior changes over time. In such problems, the scope of robustness significantly changes over time following the changes in the dynamical characteristics, such as change frequency and severity. A potential future work will be designing parameter adaptation mechanisms [43] for adapting the value of $\mathfrak{m}_{\max}$ to the changing scope of robustness over time in heterogeneous DOPs.

There are many real-world problems whose search spaces contain multiple moving promising regions and the multipopulation-based methods, such as our proposed method, are efficient in solving them. A potential future work will be solving a real-world ROOT problem. An example of real-world ROOT problems is crowd monitoring and management [44], which is a dynamic covering location problem [45]. In this problem, the locations of security agent units are changed over time based on the status of the crowd. However, frequently changing the locations of security agent units is undesirable as it disturbs the monitoring task. Consequently, in this problem, we seek solutions (i.e., the locations of the security agent units) that can remain acceptable for a longer time.

References

[1] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 6, pp. 1–24, Oct. 2012.

[2] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decades—Part A," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 609–629, Aug. 2021.

[3] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, Dept. School Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2011.

[4] J. Branke, *Evolutionary Optimization in Dynamic Environments*, vol. 3. New York, NY, USA: Springer, 2012.

[5] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decades—Part B," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 630–650, Aug. 2021.

[6] R. Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia, and M. Goh, "Covering problems in facility location: A review," *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 368–407, 2012.

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

[8] D. Yazdani, R. Cheng, C. He, and J. Branke, "Adaptive control of subpopulations in evolutionary dynamic optimization," *IEEE Trans. Cybern.*, early access, Dec. 7, 2020, doi: 0.1109/TCYB.2020.3036100.

[9] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 1–15, Feb. 2020.

[10] D. Yazdani, "Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost," Ph.D. dissertation, Dept. Doctor Philos., Liverpool John Moores Univ., Liverpool, U.K., 2018.

[11] Y. Huang, Y. Ding, K. Hao, and Y. Jin, "A multi-objective approach to robust optimization over time considering switching cost," *Inf. Sci.*, vols. 394–395, pp. 183–197, Jul. 2017.

[12] D. Yazdani, J. Branke, M. N. Omidvar, T. T. Nguyen, and X. Yao, "Changing or keeping solutions in dynamic optimization problems with switching costs," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 1095–1102.

[13] X. Yu, Y. Jin, K. Tang, and X. Yao, "Robust optimization over time—A new perspective on dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–6.

[14] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Characterizing environmental changes in robust optimization over time," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

[15] Y. Jin, K. Tang, X. Yu, B. Sendhoff, and X. Yao, "A framework for finding robust optimal solutions over time," *Memetic Comput.*, vol. 5, no. 1, pp. 3–18, 2013.

[16] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Robust optimization over time: Problem difficulties and benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 731–745, Oct. 2015.
[17] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Finding robust solutions to dynamic optimization problems," in *Proc. Eur. Conf. Appl. Evol. Comput.*, 2013, pp. 616–625.
[18] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 143–155, Feb. 2019.
[19] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, 1999, pp. 1875–1882.
[20] D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, "A new multi-swarm particle swarm optimization for robust optimization over time," in *Applications of Evolutionary Computation*, G. Squillero and K. Sim, Eds. Cham, Switzerland: Springer Int., 2017, pp. 99–109.
[21] Y. Huang, Y. Jin, and K. Hao, "Decision-making and multi-objectivization for cost sensitive robust optimization over time," *Knowl. Based Syst.*, vol. 199, Jul. 2020, Art. no. 105857.
[22] M. Fox, S. Yang, and F. Caraffini, "An experimental study of prediction methods in robust optimization over time," in *Proc. Congr. Evol. Comput.*, 2020, pp. 1–7.
[23] P. Novoa-Hernández, D. A. Pelta, and C. C. Corona, "Approximation models in robust optimization over time—An experimental study," in *Proc. Congr. Evol. Comput.*, 2018, pp. 1–6.
[24] Y.-N. Guo, M. Chen, H. Fu, and Y. Liu, "Find robust solutions over time by two-layer multi-objective optimization method," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2014, pp. 1528–1535.
[25] M. Chen, Y. Guo, H. Liu, and C. Wang, "The evolutionary algorithm to find robust Pareto-optimal solutions over time," *Math. Problems Eng.*, vol. 2015, Apr. 2015, Art. no. 814210.
[26] Y. Guo, H. Yang, M. Chen, J. Cheng, and D. Gong, "Ensemble prediction-based dynamic robust multi-objective optimization methods," *Swarm Evol. Comput.*, vol. 48, pp. 156–171, Aug. 2019.
[27] Y. Guo, H. Yang, M. Chen, D. Gong, and S. Cheng, "Grid-based dynamic robust multi-objective brain storm optimization algorithm," *Soft Comput.*, vol. 24, no. 10, pp. 7395–7415, 2020.
[28] D. Yazdani, M. N. Omidvar, R. Cheng, J. Branke, T. T. Nguyen, and X. Yao, "Benchmarking continuous dynamic optimization: Survey and generalized test suite," *IEEE Trans. Cybern.*, vol. 52, no. 5, pp. 3380–3393, May 2022, doi: 10.1109/TCYB.2020.3011828.
[29] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: Detection and response to dynamic systems," in *Proc. Congr. Evol. Comput.*, vol. 2, 2002, pp. 1666–1670.
[30] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, "A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2144–2158, 2013.
[31] L. Adam and X. Yao, "A simple yet effective approach to robust optimization over time," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, 2019, pp. 680–688.
[32] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 959–974, Dec. 2010.
[33] T. Blackwell, *Particle Swarm Optimization in Dynamic Environments*. Berlin, Germany: Springer, 2007, pp. 29–49.
[34] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, Aug. 2006.
[35] K. Trojanowski, "Properties of quantum particles in multi-swarms for dynamic optimization," *Fundamenta Informaticae*, vol. 95, nos. 2–3, pp. 349–380, 2009.
[36] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A hibernating multi-swarm optimization algorithm for dynamic environments," in *Proc. 2nd World Congr. Nat. Biol. Inspired Comput.*, 2010, pp. 363–369.
[37] D. Yazdani *et al.*, "IEEE CEC 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark," 2021, *arXiv:2106.06174*.
[38] D. Yazdani. "Generalized Moving Peaks Benchmark for Robust Optimization Over Time (MATLAB Source Code)." 2021. [Online]. Available: https://bitbucket.org/public-codes-danial-yazdani/gmpb-for-root/src/main/ (Accessed: Dec. 6, 2021).
[39] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, 2001, pp. 84–88.
[40] H. Richter, "Detecting change in dynamic fitness landscapes," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1613–1620.
[41] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," in *Swarm Intelligence: Introduction and Applications* (Lecture Notes in Computer Science), C. Blum and D. Merkle, Eds. Berlin, Germany: Springer, 2008, pp. 193–217.
[42] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[43] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, Apr. 2020.
[44] C. Martella, J. Li, C. Conrado, and A. Vermeeren, "On current crowd management practices and the need for increased situation awareness, prediction, and intervention," *Safety Sci.*, vol. 91, pp. 381–393, Jan. 2017.
[45] F. Plastria, *Covering Location Problems*. New York, NY, USA: Springer-Verlag, 2002, pp. 37–79.

Danial Yazdani (Member, IEEE) received the Ph.D. degree in computer science from Liverpool John Moores University, Liverpool, U.K., in 2018.

He is currently a Research Fellow with the Data Science Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia. Prior to that, he was a Research Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His main research interests include evolutionary algorithms, dynamic optimization problems, large-scale optimization, and covering location problems.

Dr. Yazdani was a recipient of the Best Thesis Award from the Faculty of Engineering and Technology, Liverpool John Moores University, and the SUSTech Presidential Outstanding Postdoctoral Award from the Southern University of Science and Technology. He is a member of the IEEE Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, and the IEEE Task Force on Large-Scale Global Optimization.

Donya Yazdani received the Ph.D. degree in computer science from the University of Sheffield, Sheffield, U.K., in 2020, with a thesis on the time complexity analysis of artificial immune systems for combinatorial optimization.

She is currently a Visiting Researcher with the University of Sheffield. Prior to that, she was a Lecturer with the Department of Computer Science, Aberystwyth University, Aberystwyth, U.K. Her current research interests include theoretical analysis of evolutionary algorithms, dynamic optimization problems, and combinatorial optimization.

Jürgen Branke (Member, IEEE) received the Ph.D. degree from the University of Karlsruhe, Karlsruhe, Germany, in 2000.

He is a Professor of Operational Research and Systems with the Warwick Business School, University of Warwick, Coventry, U.K. He has been an active Researcher in the area of evolutionary optimization since 1994 and has published more than 200 papers in international peer-reviewed journals and conferences and a book on *Evolutionary Optimization in Dynamic Environments*. Besides dynamically changing environments, his research interests include multiobjective optimization, handling of uncertainty in optimization, simulation-based optimization, and the design of complex systems.

Prof. Branke is an Editor of *ACM Transactions on Evolutionary Learning and Optimization*, an Area Editor of the *Journal of Heuristics* and the *Journal on Multi-Criteria Decision Analysis*, as well as an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *Evolutionary Computation Journal*.

Mohammad Nabi Omidvar (Senior Member, IEEE) received the first bachelor's degree (First Class Hons.) in computer science, the second bachelor's degree in applied mathematics, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2010, 2014, and 2016, respectively.

He is currently an Assistant Professor of Artificial Intelligence in Financial Services affiliated with Leeds University Business School and School of Computing, University of Leeds, Leeds, U.K., and the Chair of IEEE Computational Intelligence Taskforce on Large-Scale Global Optimization. Prior to that, he was a Research Fellow with the School of Computer Science, University of Birmingham, Birmingham, U.K. His current research interests include large-scale global optimization, high-dimensional machine learning, and AI for financial services.

Dr. Omidvar is the winner of IEEE CEC Large-Scale Global Optimization Competition in 2019, and a recipient of the IEEE TRANSACTION ON EVOLUTIONARY COMPUTATION Outstanding Paper Award for his research on large-scale global optimization in 2017, the Australian Postgraduate Award in 2010, and the Best Computer Science Honours Thesis Award from the School of Computer Science and IT, RMIT University.

Amir Hossein Gandomi (Senior Member, IEEE) received the Ph.D. degree in engineering from the University of Akron, Akron, OH, USA, in 2015.

He is a Professor of Data Science and an ARC DECRA Fellow with the Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), Ultimo, NSW, Australia. Prior to joining UTS, he was an Assistant Professor with the Stevens Institute of Technology, Hoboken, NJ, USA, and a Distinguished Research Fellow with the BEACON Center, Michigan State University, East Lansing, MI, USA. He has published over 300 journal papers and 12 books which collectively have been cited over 27 000 times (H-index = 77). His research interests are global optimization and (big) data analytics using machine learning and evolutionary computations in particular.

Prof. Gandomi has received multiple prestigious awards for his research excellence and impact, such as the 2022 Walter L. Huber Prize. He has been named as one of the most influential scientific minds and Highly Cited Researcher (top 1% publications and 0.1% researchers) for five consecutive years from 2017 to 2021. He also ranked 17th in GP bibliography among more than 12 000 researchers. He has served as an associate editor, editor, and guest editor in several prestigious journals, such as an Associate Editor of IEEE TRANSACTIONS ON BIG DATA and IEEE INTERNET OF THINGS JOURNAL. He is active in delivering keynotes and invited talks.

Xin Yao (Fellow, IEEE) received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 1990.

He is a Chair Professor of Computer Science with the Southern University of Science and Technology, Shenzhen, China, and a part-time Professor of Computer Science with the University of Birmingham, Birmingham, U.K. He was a Distinguished Lecturer of the IEEE Computational Intelligence Society (CIS). His major research interests include evolutionary computation, ensemble learning, and their applications to software engineering.

Prof. Yao received a prestigious Royal Society Wolfson Research Merit Award in 2012, the IEEE CIS Evolutionary Computation Pioneer Award in 2013, and the 2020 IEEE Frank Rosenblatt Award. His paper on evolving artificial neural networks won the 2001 IEEE Donald G. Fink Prize Paper Award. He also won the 2010, 2016, and 2017 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Awards, the 2011 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award, and many other best paper awards. He was the President of IEEE CIS from 2014 to 2015 and the Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008.