Yazdani, D, Yazdani, D, Blanco-Davis, E and Nguyen, TT

A survey of multi-population optimization algorithms for tracking the moving optimum in dynamic environments

Journal of Membrane Computing
https://doi.org/10.1007/s41965-024-00163-y

REVIEW PAPER

A survey of multi-population optimization algorithms for tracking the moving optimum in dynamic environments

Delaram Yazdani[1] · Danial Yazdani[2] · Eduardo Blanco-Davis[1] · Trung Thanh Nguyen[3]

Received: 25 April 2024 / Accepted: 17 June 2024

Abstract
The solution spaces of many real-world optimization problems change over time. Such problems are called dynamic optimization problems (DOPs), which pose unique challenges that necessitate adaptive strategies from optimization algorithms to maintain optimal performance and responsiveness to environmental changes. Tracking the moving optimum (TMO) is an important class of DOPs where the goal is to identify and deploy the best-found solution in each environments Multi-population dynamic optimization algorithms are particularly effective at solving TMOs due to their flexible structures and potential for adaptability. These algorithms are usually complex methods that are built by assembling multiple components, each of which is responsible for addressing a specific challenge or improving the tracking performance in response to changes. This survey provides an in-depth review of multi-population dynamic optimization algorithms, focusing on describing these algorithms as a set of multiple cooperating components, the synergy between these components, and their collective effectiveness and/or efficiency in addressing the challenges of TMOs. Additionally, this survey reviews benchmarking practices within this domain and outlines promising directions for future research.

Keywords Dynamic optimization problems · Tracking the moving optimum · Multi-population optimization algorithms · Benchmarking

1 Introduction

Dynamic optimization problems (DOPs) evolve over time, causing changes in their search spaces [1]. These changes can be influenced by various factors, such as evolutionary processes within the algorithm, variable interactions, spontaneous environmental changes, or time-linkage effects, where current solutions impact future problem states [2, 3]. These problems are common across numerous real-world scenarios, including energy management systems for hybrid electric vehicles [4], dynamic scheduling issues [5, 6], and dynamic trajectory optimization tasks [7–9]. Within these dynamic scenarios, the efficiency of previously optimal solutions can significantly decrease after environmental changes. This requires the application of dynamic optimization algorithms (DOAs) that excel in detecting and adapting to changing optima over time [10–12].

DOPs are categorized by several factors, including whether their search spaces are continuous [13] or discrete [14], aimed at single [15] or multiple objectives [16–18], and whether they focus on tracking optimal solutions or finding solutions robust to upcoming environmental changes [19,

Danial Yazdani, Eduardo Blanco-Davis and Trung Thanh Nguyen are equal contributing to this work.

✉ Trung Thanh Nguyen
T.T.Nguyen@ljmu.ac.uk

Delaram Yazdani
delaram.yazdani@yahoo.com

Danial Yazdani
danial.yazdani@gmail.com

Eduardo Blanco-Davis
E.E.BlancoDavis@ljmu.ac.uk

1 Faculty of Engineering and Technology, Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Liverpool John Moores University, Liverpool L3 3AF, United Kingdom

2 Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo 2007, Australia

3 Faculty of Engineering and Technology, Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Liverpool John Moores University, Liverpool L2 2ER, United Kingdom

Published online: 02 July 2024

20]. While several surveys offer reviews of different DOP types, encapsulating multiple categories within a single review [1, 10, 11, 21], others dedicate their focus to a specific type of DOPs. Numerous surveys have reviewed various types of DOPs, including multiple categories within one study [1, 10, 11, 21]. In contrast, other surveys focus on a specific type of DOP, aiming for a more in-depth exploration and detailed insights, such as those on robust optimization over time (ROOT) [22, 23], multi-objective DOPs [16, 17, 24], combinatorial DOPs [14], and continuous single-objective DOPs [15, 25]. Additionally, some surveys concentrate on algorithmic or benchmarking aspects, with topics covering dynamic benchmark generators [26], self-adaptive methods for solving DOPs [27], and hyper-heuristics for DOPs [28].

Single-objective unconstrained continuous DOPs with the goal of tracking the moving optimum (TMO) represent a widely studied area with an extensive body of literature. The mechanisms and methods developed for addressing this specific type of DOP can be readily adapted and extended to address various other DOPs, such as ROOT [29] and constrained DOPs [30]. In this survey, we focus on single-objective unconstrained continuous DOPs which can be formulated as follows:

$$f^{(t)}(\mathbf{x}) = f\left(\mathbf{x}, \boldsymbol{\alpha}^{(t)}\right),\ \mathbf{x} = \{x_1, x_2, \cdots, x_d\},\ \mathbf{x} \in \mathbb{X} \tag{1}$$

where $\mathbf{x}$ is a d-dimensional solution within the search space $\mathbb{X}$. The objective function is denoted as f, influenced by a vector of time-dependent control parameters $\boldsymbol{\alpha}$, which can trigger changes in the search space. The time index is $t \in [0, T]$. The TMO literature mainly focuses on scenarios where environmental changes happen at discrete time intervals, that is, $t \in \{1, \dots, T\}$. In the context of a problem featuring T environmental states, it is assumed that there exists a sequence of T stationary environments. In the literature, it is commonly presumed that successive environmental states share a degree of similarity, a characteristic that is often observed in real-world problems [3, 31]. The dynamic objective function $f^{(t)}(\mathbf{x})$ across T environments can be restructured as follows:

$$\begin{aligned}\text{Maximize } :& f^{(t)}(\mathbf{x}) = \left\{f(\mathbf{x}, \boldsymbol{\alpha}^{(k)})\right\}_{k=1}^{T} \\ &= \left\{f(\mathbf{x}, \boldsymbol{\alpha}^{(1)}), f(\mathbf{x}, \boldsymbol{\alpha}^{(2)}), \dots, f(\mathbf{x}, \boldsymbol{\alpha}^{(T)})\right\}.\end{aligned} \tag{2}$$

The main objective in TMO is to find and deploy the optimum solution in each environment t.

The two-part survey in [15, 25] focuses on this specific class of DOPs. DOAs are typically constructed by combining multiple components to address the unique challenges posed by TMOs, thus enhancing and accelerating the optimization process in response to each environmental change. Existing classifications [1, 11, 21] for DOAs are basically performance-based, which can lead to significant overlap, with a single algorithm potentially falling into multiple categories. Additionally, these classifications may not comprehensively cover essential components such as convergence detection and computational resource allocation. To address these issues, the survey in [15] proposes a comprehensive taxonomy for the *components* of DOAs. By utilizing a component-based taxonomy, the survey aims to achieve a clearer categorization of the different components of DOAs, effectively reducing overlaps. These components include:

Population management:
Includes strategies for organizing and managing the population, incorporating multi-population approaches that utilize multiple subpopulations to explore different regions of the search space and maintain exploration capability [13, 32].
Explicit Memory:
Includes storing information from past environments, such as the locations of promising regions, to accelerate the optimization process in new environments by utilizing historical data [33].
Diversity Control:
Concerns strategies to manage the diversity within a population, divided into global diversity control (aiming to maintain exploration capabilities) and local diversity control (aiming to boost exploitation efforts in areas of promise) [34–36].
Convergence Detection:
Refers to methods for determining if a population or subpopulation has converged on a promising area [37, 38].
Environmental Change Detection:
Covers methods for detecting changes in the environment [39, 40].
Static Optimization:
Comprises various optimization algorithms originally designed for static environments [10, 11]. When combined with the components mentioned previously, these optimization techniques are effective at navigating the unique characteristics, requirements, and challenges posed by DOPs.

In [15, 25], the authors review and explain the methods of how each component works. However, discussing each component separately makes it hard to understand how they come together to form DOAs. This approach may prevent readers from seeing how these parts interact as a unified whole. Each DOA involves complex interactions among different components, and understanding these interactions is essential for fully understanding DOAs.

Identifying this critical gap, our survey aims to provide a detailed review of DOAs in their entirety, emphasizing the cohesive interaction among their components for solving TMOs. This approach aims to offer a bird's-eye view, facilitating a deeper understanding of the construction and operational dynamics of DOAs. By reviewing the collective functionality and connection of components as a unified system, we aim to equip researchers with a more comprehensive insight into the architectural structures of DOAs, thus enhancing their understanding of the design and effectiveness of DOAs in tackling dynamic optimization challenges.

Based on the classification outlined in existing surveys [10, 11, 15] in the field, DOAs for TMO can be categorized into single-, bi-, and multi-population algorithms. Among these, multi-population algorithms stand out as the current state-of-the-art in navigating the complexities of TMO in dynamic environments. While this survey focuses on multi-population algorithms, it is important to distinguish these from co-evolutionary and divide-and-conquer strategies. Multi-population algorithms explore different regions of the search space with several subpopulations that may interact through information exchange. Co-evolutionary algorithms also use multiple populations but emphasize their interactions through competition or cooperation, crucial for problems with interdependent components [41]. Thus, multi-population algorithms can be seen as a subset of co-evolutionary approaches when such interactions are present. In contrast, divide-and-conquer strategies solve a problem by breaking it into smaller, independent subproblems and combining their solutions [42].

Multi-population algorithms are proven to be effective and widely utilized for solving TMOs. Their prominence can be attributed to their inherent flexibility in incorporating multiple diverse components, their compatibility with various optimization strategies, and their capacity to enhance or preserve diversity by managing multiple populations simultaneously [13]. Given these compelling attributes, this survey narrows its focus to multi-population DOAs (mDOAs), on explaining how these advanced algorithms benefit from their complex structures and diverse components to address TMO challenges.

In this survey, we explain the mDOAs by describing their components and illustrating how these components are assembled to build an mDOA framework. Our analysis utilizes the component classification proposed in [15], with Fig. 1 serving as a visual guide. For a more in-depth understanding of these classifications, we refer readers to [15]. To improve readability, we categorize the explanation of algorithms based on two distinct factors rather than introducing a new taxonomy. These factors ensure that algorithms within one category do not overlap with those in another. The two factors considered are the nature of the subpopulations and the population size. In Sects. 2 and 3, we categorize the algorithms according to whether their subpopulations shows homogeneity (i.e., are identical to each other) or heterogeneity, and by whether the population size remains constant or varies over time.

Moreover, this survey reviews the benchmarking methods used for evaluating the performance of mDOAs. We describe the commonly used and state-of-the-art dynamic benchmark generators and performance indicators. Concluding our survey, we explore potential avenues for future research, aiming to inspire continued advancements in the field.

The rest of this survey paper is organized as follows: Sect. 4 reviews dynamic benchmark generators commonly used in the TMO literature, alongside well-known performance indicators. It also explores the applications of mDOAs in real-world problems. Section 5 outlines potential future research directions in the field. Finally, Sect. 6 concludes this survey.

2 mDOAs with homogeneous subpopulations

Homogeneous subpopulations imply that all subpopulations are identical to each other [15]. This uniformity means that they all use the same optimizer and parameter settings. Additionally, they share identical tasks and roles. Furthermore, all subpopulations have an equal number of members. Homogeneous subpopulations are often utilized in multi-population DOAs that use the indices of individuals to form subpopulations [13, 43] and in those that employ clustering methods where the number of individuals is predefined as an input parameter [44–46]. The population size of mDOAs can either remain constant or be adjusted by replacing or generating subpopulations that resemble the existing ones. In the following, mDOAs with homogeneous subpopulations are described in detail. For further clarification, we categorize them based on whether the overall population size is modified or remains unchanged during the process.

2.1 Homogeneous mDOAs with constant population size

Some mDOAs employ index-based clustering methods to create subpopulations, leading to homogeneity among these subpopulations. Blackwell and Branke [47] introduce two mDOAs: mCPSO and mQSO, where the population divided into subpopulations based on the index of the individuals, and both the number of subpopulations and their sizes are predetermined. To enhance global diversity, these mDOAs utilize an exclusion method, randomizing redundant subpopulations when the Euclidean distance between the best-found positions of any two subpopulations falls below a predefined threshold r_{excl}. Meanwhile, local diversity is preserved over time in mCPSO by employing charged particles [35], and in

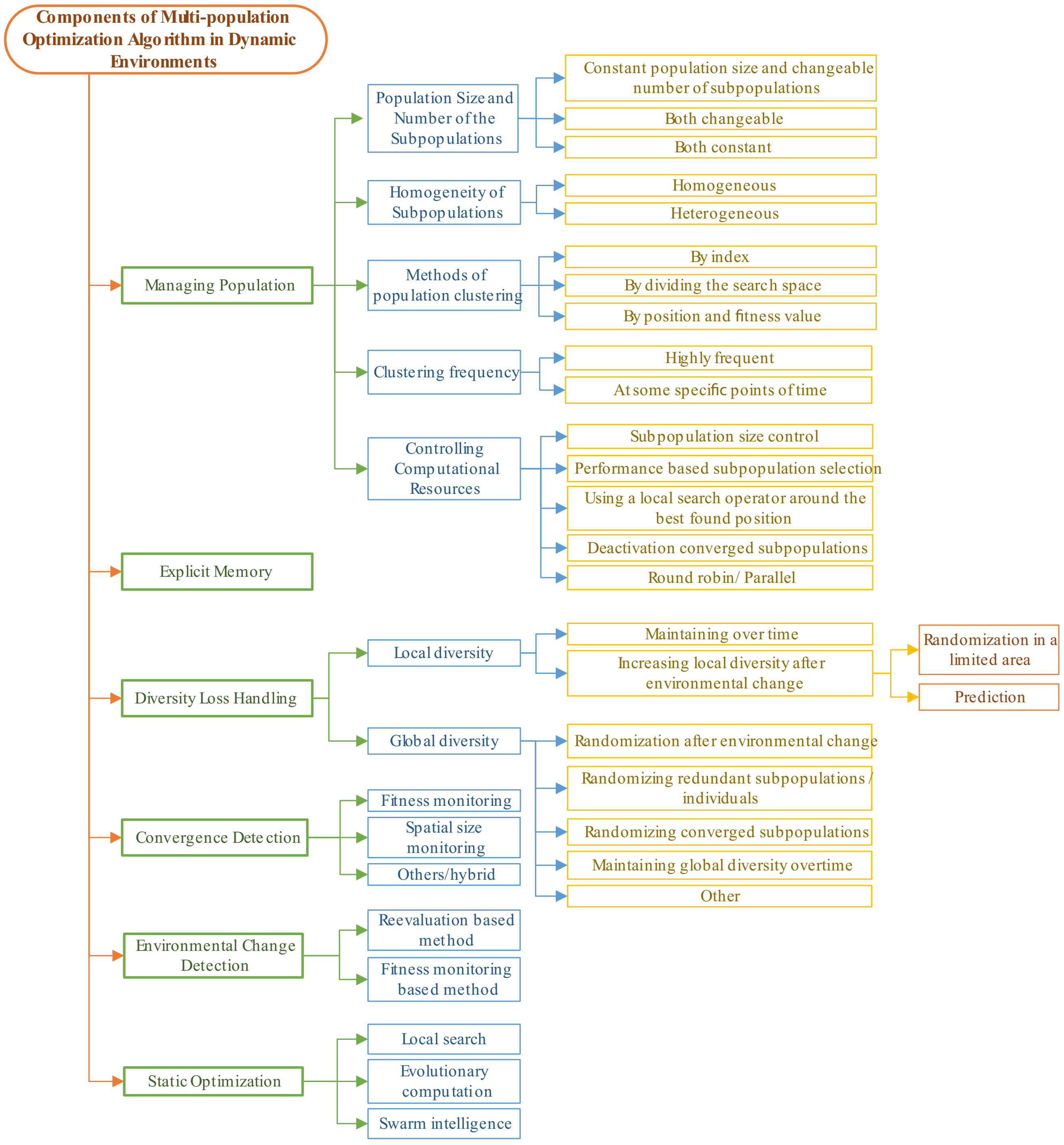

Components of Multi-population Optimization Algorithm in Dynamic Environments
Managing Population
Population Size and Number of the Subpopulations
Constant population size and changeable number of subpopulations
Both changeable
Both constant
Homogeneity of Subpopulations
Homogeneous
Heterogeneous
Methods of population clustering
By index
By dividing the search space
By position and fitness value
Clustering frequency
Highly frequent
At some specific points of time
Controlling Computational Resources
Subpopulation size control
Performance based subpopulation selection
Using a local search operator around the best found position
Deactivation converged subpopulations
Round robin/ Parallel
Explicit Memory
Diversity Loss Handling
Local diversity
Maintaining over time
Increasing local diversity after environmental change
Randomization in a limited area
Prediction
Global diversity
Randomization after environmental change
Randomizing redundant subpopulations / individuals
Randomizing converged subpopulations
Maintaining global diversity overtime
Other
Convergence Detection
Fitness monitoring
Spatial size monitoring
Others/hybrid
Environmental Change Detection
Reevaluation based method
Fitness monitoring based method
Static Optimization
Local search
Evolutionary computation
Swarm intelligence

Fig. 1 Classification of the components of mDOAs [15]

mQSO, through the use of quantum particles within a radius around each sub-population's best-found position. Additionally, a classic round-robin approach ensures the equitable distribution of computational resources among subpopulations. To adapt to environmental changes, the algorithms detect changes by reevaluating the best positions within each subpopulation and respond by updating their memory. Further enhancements to these mDOAs in [13] have introduced an anti-convergence method, aimed at increasing global diversity by randomizing the worst-performing subpopulations once convergence is achieved. The convergence is determined when the maximum distance along any dimension between individuals within a subpopulation falls below a predefined threshold.

The mDOA proposed in [43], DynDE, introduces a multi-population structure with predetermined member counts in each subpopulation and a fixed number of subpopulations. It utilizes the exclusion method outlined in [13] to randomize redundant subpopulations, thus increasing global diversity. To preserve local diversity over time, DynDE employs three methods: the use of quantum individuals [47], the incorporation of Brownian individuals with a Gaussian distribution, and the implementation of entropic models that introduce a Gaussian step in each iteration. Moreover, an equal distribution of computational resources among subpopulations is achieved through the classic round-robin method, ensuring that all subpopulations receive an equitable share of computational resources. The algorithm has been further enhanced as detailed in [48] by utilizing two methods for managing computational resources. The first method adjusts the size of subpopulations by a migration strategy, effectively reallocating more individuals to better-performing subpopulations and transferring individuals from inferior ones. The second method implements a prioritization mechanism, focusing all computational resources on higher performing subpopulations for a given iteration, while temporarily deactivating inferior ones. This prioritization is based on the fitness value of the best-found position within a subpopulation.

Brest et al. [49] introduce a mDOA that adopts the same multi-population structure as described in [47], employing several methods to effectively manage global diversity. First, it randomizes redundant, inferior subpopulations that overlap by utilizing an exclusion method similar to [47], but with a modification wherein the exclusion radius is a fixed, user-defined number. Secondly, the algorithm includes an aging-based method to introduce additional randomization among individuals or subpopulations. If the age of a non-best individual exceeds a predetermined threshold, it is randomized with a specified probability. Similarly, if the best individual of a subpopulation ages beyond a different threshold, the entire subpopulation is randomized, also with a specified probability. Moreover, if the distance between the individual and the best individual within a subpopulation is less than another threshold, the individual itself is randomized. Note that there are two modes for the location of randomization. Individuals can be randomized across the search space or around one of the best found positions in the previous environments, which are stored in explicit memory. Computational resources are evenly distributed among subpopulations through the application of a classic round-robin method. The algorithm detects a change by reevaluating the best found position within each subpopulation and responds by reevaluating all solutions to address outdated memory, and storing the best found position of each subpopulation in explicit memory from the previous environment.

Plessis and Engelbrecht [50] propose an improved version of the DynDE algorithm [43], with some modifications. Herein, to maintain local diversity over time, Brownian individuals [43] are used. Additionally, the exclusion method has been modified to address the challenge of differentiating between subpopulations located in the same promising regions or in extremely close ones. This method involves finding the midpoint between the best positions of overlapping sub-populations. If the fitness of this midpoint is worse than that of the best positions of both sub-populations, then neither sub-population should be re-initialized; otherwise, the inferior subpopulation is randomized to increase global diversity. Furthermore, the allocation of computational resources is managed by prioritizing subpopulations based on performance. First, all subpopulations run for two iterations, during which the improvement in fitness of the best position is calculated. Then, the difference between the fitness of the best position of a subpopulation and the worst best-found fitness among all sub-populations is considered. Based on these two factors, performance is assessed. In each iteration, only the subpopulation with the highest performance is activated and allowed to continue, ensuring efficient use of resources until another subpopulation demonstrates superior potential.

In [51], an mDOA is introduced, similar to the approach found in [13]. This approach involves a fixed number of subpopulations, each maintaining a set population size. To ensure local diversity during iterations, two randomization methods are employed. First, randomization around each individual, and secondly, a Levy distribution-based approach. Moreover, global diversity is enhanced through three randomization methods. The first method involves randomizing converged non-best subpopulations, which are identified by observing each member's fitness progress over a set period. A subpopulation is considered to be converged if its members fail to demonstrate significant progress within this duration. Second, individuals that have not shown sufficient improvement within a certain number of iterations are randomized. Third, redundant subpopulations are randomized through the exclusion method [13], wherein inferior overlapping subpopulations are randomized.

Zuo and Xiao [52] also utilize a multi-population approach similar to [13]. The proposed mDOA uses explicit memory to store and retrieve solutions. After a change occurs, a predefined number of individuals within a subpopulation are replaced with the stored solutions, while the remaining members are randomized, leading to increased global diversity. Furthermore, fair distribution of computational resources among subpopulations is achieved through the classic round-robin method. Change detection is facilitated by reevaluating the best-found position of each subpopulation.

The mDOA presented in [53] utilizes a multi-population method with a constant number of subpopulations. The local diversity of each subpopulation is preserved over time through randomizing individuals within a self-adaptive radius from the best individual within the subpopulation, termed as quantum individuals. In the proposed mDOA, change is detected by reevaluating the best individual in the entire population.

Vafashoar and Meybodi [54] propose a multi-population approach with a fixed number of subpopulations. In the proposed mDOA, the global diversity is increased in three ways. Firstly, by employing an anti-convergence method that randomizes converged subpopulations during the environment, noting that the algorithm benefits from explicit memory to archive the best-found position before randomization. Secondly, after each environmental change, individuals within each sub-population with fitnesses below the median of the sub-population's fitnesses are randomized. Thirdly, by randomizing redundant subpopulations through exclusion method [13]. The convergence of a subpopulation in the proposed mDOA, is determined by monitoring the fitness of its best-found position; if there is no improvement for a predefined iteration, it is considered the subpopulation is converged. Through the classic round robin method, computational resources are distributed evenly among subpopulations. The change detection component operates by re-evaluating the best individual of each sub-population at each generation. For change reaction, the best individual of each sub-population is added to explicit memory. Additionally, the median fitness of the individuals in each sub-population is calculated, and those with fitness worse than the median are randomized across the search space before all individuals are re-evaluated.

Novoa et al. [55], propose an mDOA with an adaptive number of subpopulations, while maintaining the overall size of the population and the number of members constant. The local and global diversity both increase after an environmental change through a grouping strategy. After each environmental change, individuals in each sub-population are sorted and divided into three predefined groups. The best group's individuals retain their previous positions. The second group's individuals are randomized around the best-found position using a Gaussian distribution to enhance local diversity. The last group, containing the worst individuals of the sub-population, is randomized across the search space to boost global diversity. Additionally, this algorithm incorporates exclusion and anti-convergence methods from [13] to further increase global diversity through the randomization of subpopulations. Computational resources are managed by deactivating low-quality subpopulations that have converged, based on a specific convergence determination rule. The quality of each subpopulation is assessed using a fuzzy decision rule, considering various factors, including the best fitness value, the average fitness of all sub-populations, and the fitness of each individual sub-population. If a subpopulation is considered converged and of low quality, without any recent environmental change, it is temporarily deactivated until the next environmental change occurs. Detect a change is by reevaluating the best-found position among all subpopulations. This is followed by a reaction that includes reevaluating all solutions to ensure the memory is updated.

In the mDOA proposed in [56], a fixed number of subpopulations are formed according to the division of the search region. This involves a master node as a controller and several slave nodes. The master node first scans the entire search space and, depending on the number of slave nodes, generates population bounds. The search space and bounds are evenly divided, and within these bounds, random individuals are generated until each sub-population reaches a predetermined size. To maintain global diversity, the algorithm employs two methods: reinitialization based on success rate and a population aging mechanism. For the first method, if a sub-population is ineffective in finding better solutions, indicated by its success rate, it is reinitialized with random individuals. For the second method, a predefined threshold value as population age is set, and if a sub-population does not contribute to finding better solutions after this period, it is re-initialized with random individuals. The algorithm also benefits from two explicit memories. The first memory, collects the best individuals from all sub-populations, while the processed results, representing the optimal solutions up to the current iteration, are stored in second memory. These stored information are used under certain conditions. First, when an environmental change is detected, the algorithm uses historical information to quickly adapt to the changes. Second, when sub-populations are reinitialized, the historical information may be used to ensure that the new individuals generated are not just random but are somewhat informed by past successes. To detect an environmental change, some test points are reevaluated at each iteration, which are randomly selected individuals from the population. If there is a difference in the fitness of the chosen test points between iterations, a change is happen.

2.2 Homogeneous mDOAs with varying population size

Blackwell [57] introduces AmQSO, an adaptive version of the mQSO algorithm proposed in [13]. In AmQSO, the number of individuals within each subpopulation remains constant, but the number of subpopulations can dynamically vary over time, leading to a flexible overall population size. Initially, AmQSO starts with a single population. As it converges, additional subpopulation is initialized. Once multiple subpopulations exist, they must all converge before a new one can be added. The anti-convergence method [13] is removed here, as the algorithm's adaptation in the number of subpopulations to discover promising regions enhances global diversity. Similar to [13], local diversity is maintained over time using quantum particles [47]. AmQSO employs the exclusion method [13] with modifications, opting to remove the inferior subpopulation in case of a collapse instead of randomizing it. This modification saves computational resources and prevents unnecessary exploitation and overcrowding. The allocation of computational resources among subpopulations follows a classic round-robin method, ensuring fairness. As part of AmQSO, detecting a change involves reevaluating the best-found position of the entire population, followed by reevaluating all solutions to update outdated memory.

Rezazadeh et al. [58] introduce an mDOA where the number of subpopulations adjusts adaptively to the discovered promising regions. Global diversity is enhanced by initializing new subpopulations. To achieve this, if the count of non-converged subpopulations falls below a certain threshold, a new subpopulation is initiated. However, there is an upper bound for the maximum number of subpopulations. If the count exceeds this threshold, the worst subpopulations are removed until their number reaches the maximum limit. This strategy of controlling the number of subpopulations helps prevent overcrowding and minimizes the wastage of computational resources. In the proposed mDOA, the local diversity is increased under certain conditions during the optimization process by modifying the core optimizer rule. The mDOA introduced in [58] has been further improved in [59]. This enhancement addresses the issue of diversity loss after environmental changes. When a change occurs, the sub-populations are divided into two groups, the better and worse ones. The individuals in the better sub-populations are randomized around their best-found positions to increase local diversity. Meanwhile, those in the worse sub-populations are randomized across the search space to enhance global diversity. Moreover, two methods are employed to manage computational resources. First, by deactivating subpopulations with velocity vectors below a certain threshold for a set number of iterations. Secondly, by allocating additional computational resources to the best-performing subpopulation. This is achieved by performing a local search around the best-found position among all subpopulations in each iteration.

The mDOA proposed in [60] uses a population structure similar to [61] which features an adaptive number of subpopulations. Global diversity is increased by initializing a new subpopulation whenever all current subpopulations have converged. A subpopulation is considered converged when the longest Euclidean distance between any pair of individuals in a subpopulation is less than a threshold. Local diversity is enhanced after an environmental change by modifying the rules of the core optimizer. To manage computational resources more effectively, more resources are allocated for the best found position among all subpopulations through performing a local search around it.

An adaptive mDOA is introduced in [62], utilizing a population management approach similar to [61], where initially, there is a single population exploring the search space, and as this population converges, a new subpopulation is generated and introduced to explore other regions. The convergence of a subpopulation is determined by comparing the best position found in the current iteration with predefined previous iterations, indicating convergence if the difference is below a certain threshold. This mechanism enables the number of subpopulations to adaptively change based on the discovered promising regions, thus increasing the global diversity. Local diversity is enhanced after an environmental change by randomizing non-best individuals around the best individual within a radius. To effectively manage computational resources, more resources are allocated to the best-performing subpopulation, and by reevaluating a single individual, a change is detected. The proposed mDOA has been further improved in [63], first by utilizing the method for controlling fitness resources through deactivating converged subpopulations from [64], and second by detecting changes through the reevaluation of the best-found positions of each subpopulation rather than one.

Plessis and Engelbrecht [65] introduce a multi-population structure adapted from the AmQSO [61], wherein the number of subpopulations is adaptive, and the overall population size can change. This adaptability ensures the preservation of global diversity. Local diversity is maintained over time by utilizing the Brownian individuals [50]. To effectively manage fitness resources, the algorithm employs two strategies. First, an improved version of the resource allocation method from [50] is added, using a penalty value in each sub-population's performance measurement, which decreases when there is no improvement in their best-found position from previous iterations. The second method is a modified version of the exclusion from [50], where instead of randomizing the inferior subpopulation, it removes that subpopulation to prevent the unnecessary consumption of computational resources. In the proposed mDOA, the convergence status of a subpopulation is determined based on the fitness monitoring of the best-found position. If fitness does not improve from the last predefined iteration, it is considered to have converged. By re-evaluating the best individuals of each sub-population an environmental change is detected.

The proposed mDOA in [66] has been improved in [67]. The approach to population handling changed, allowing the overall population size to adaptively change. This adjustment is made by comparing the number of overall individuals, which meets the increasing global diversity condition, to the previously met condition. The increasing global diversity condition is defined based on a time period parameter. If the ratio of the difference between the number of subpopulations in the current fitness evaluation and the current fitness evaluation itself is less than a threshold, the mDOA will initiate a diversity-increasing process. Herein, if there are more individuals than the previous count, new individuals are reinitialized. If the number is less but the difference exceeds a threshold, some individuals are removed to decrease the population. If the number remains the same, the population size is maintained, and no changes are made.

Qin et al. [68] introduce a multi-population algorithm with an adaptive number of subpopulations. The number of individuals within subpopulations is fixed; however, due to adaptability, the overall population size can vary. Subpopulations can be either active or inactive. Initially, the algorithm starts with one active subpopulation to explore the search space. Upon convergence, it becomes inactive, and a new active subpopulation is created, which will maintain global diversity over time. An active subpopulation is considered converged by monitoring its spatial size, particularly when the average distance of its individuals from its best-found position is less than a predefined threshold. Inactive subpopulations remain deactivated until the environment ends, thus preventing overexploitation and conserving computational resources. Another method to avoid computational resource wastage is by removing redundant subpopulations when a new active subpopulation enters the search range of an inactive one, and the one with inferior best-found fitness is removed. After an environmental change, local diversity is enhanced by reactivating inactive subpopulations and randomizing subpopulation members around the best-found individual within the subpopulation. A detection subpopulation is employed to identify changes, separate from the active and inactive subpopulations. Each iteration re-evaluates each individual of the detection subpopulation, and a difference in fitness signifies a change.

3 mDOAs with heterogeneous subpopulations

Heterogeneous subpopulations in mDOAs differ in various aspects, such as optimization components, parameter settings, tasks, or roles [15]. For example, in some mDOAs that use the parent–child approach [69], subpopulations are heterogeneous, with some dedicated to exploring the search space to find promising regions and others focusing on exploiting these discovered regions [64]. These subpopulations usually vary in size. Additionally, they may utilize different core optimizers [70]. Clustering methods that use the position and/or fitness of individuals (e.g., k-means [71] and Nearest Better Clustering [72]) often result in heterogeneous subpopulations, forming groups with different member sizes. The overall number of individuals in these mDOAs may either vary throughout the optimization process or remain constant. In the following, the components of mDOAs that have heterogeneous subpopulations are described in details, and they are categorized based on the modification of the population size.

3.1 Heterogeneous mDOAs with constant population size

Biswas et al. [73] introduce an mDOA that utilizes a position-based clustering method to form subpopulations using the k-means algorithm. Clustering occurs at the beginning of each environment, and the number of subpopulations remains fixed. An explicit memory is employed to store the best-found position of converged subpopulations before randomization in the proposed mDOA. Subpopulation convergence is determined by the average Euclidean distance of its members to the center, which represents the average position of all members. These stored positions in explicit memory serve a crucial role, as they are utilized to replace the worst individuals within the population after an environmental change. Furthermore, local diversity is enhanced after a change by adding a random number generated from a normal distribution to the positions of all individuals. Computational resources are allocated fairly among subpopulations by employing the classic round-robin method. Additionally, the change detection mechanism involves re-evaluating a single individual to identify environmental changes accurately.

Moradi et al. [71] suggest a clustering multipopulation approach that relies on k-means algorithm which use the position of the individuals. At the start of each environment, this clustering technique creates diverse sub-populations. The number of sub-populations and the overall population size remain constant throughout the process. To enhance the algorithm's performance, an explicit memory is employed. This memory is initialized alongside the population using a Logistic Chaotic function and is updated every random number of iterations. The update process involves calculating the center of all sub-populations and determining the closest sub-population center for each archived solution in the explicit memory. Any memory solutions that are farthest from their closest sub-population center are replaced with new entries. Updating the population includes finding the center of clusters, assigning all individuals to their nearest cluster centers, and replacing the individual farthest from its corresponding cluster center with the best explicit memory solution of that cluster. If a cluster does not have any members, a nearest memory solution is chosen for replacement. The algorithm also maintains local diversity over time through the modification of optimizer rules. Detecting change involves re-evaluating a randomly chosen individual and a change reaction that updates the explicit memory, re-evaluates the archived solutions, and retrieves memory.

Oppacher and Wineberg [74] introduce a multi-population algorithm, SBGA, inspired by the Shifting Balance Theory. This algorithm begins with a large central population known as the Core Group, responsible for exploitation. To explore the search space and increase global diversity, the Core Group generates subpopulations, referred to as colonies, by distributing individuals. The number of colonies is variable and controlled by the Core Group. To prevent overlap between colonies and between colonies and the Core, the algorithm employs an exclusion method based on Hamming distance, pushing individuals of redundant colonies away if they lie within the search region of the Core Group, which also improves global diversity. Computational resources are evenly allocated among subpopulations using the classic round-robin method.

The Self-Organizing Scouts (SOS) algorithm, introduced in [75], employs a multi-population approach wherein the overall population size remains constant; however, the number of subpopulations and the members within each can vary. The algorithm starts with a single explorer population as the parent. Upon discovering a new promising region, some individuals split off to form a child population as exploiters. Resource allocation is managed through two methods: one is setting a threshold for the number of exploiters. If this number is reached, the least effective exploiter is removed. Secondly, by prioritizing better exploiters, more individuals are allocated to them. Utilizing an exclusion method increases the global diversity by randomizing the redundant subpopulations. If any individuals from the explorer population are found within an exploiter search space, they are randomly relocated to positions outside the exploiter search range. Furthermore, if the best individual of an exploiter is within the search space of another exploiter, it will be removed. Each child subpopulation exploits a hyper-ball search space with a specific diameter centered around its best individual, thus maintaining local diversity over time. In the proposed mDOA, there is no component for detecting changes, and all individuals are updated frequently to prevent outdated memory.

In [76], a mDOA is proposed, utilizing species-based clustering based on both position and fitness. Clustering occurs at every iteration. While the total population size remains constant, the number of subpopulations varies. The clustering procedure initiates by sorting individuals according to their fitness values. The individual with the best fitness is selected as the initial seed. This seed, along with any subsequent seeds, is stored in a seed set, denoted as S, while a non-seed set, M, remains empty at the start. Each individual in the sorted list is evaluated against the seeds in S: if its Euclidean distance to all seeds in S is greater than a predefined threshold, it is added to S; otherwise, it is placed in M. After the entire sorted list has been processed, individuals in M are assigned to their nearest seed in S, determined by Euclidean distance. Global diversity is enhanced in two ways. Firstly, redundant individuals are randomized when overlap between subpopulations occurs; they are merged using an exclusion method. If the number of individuals surpasses a specific threshold, the inferior ones are randomized. This randomization also happens post-clustering if a subpopulation becomes overly large. Secondly, solo seeds without any members after clustering are randomized. Utilizing a traditional round robin method, computational resources are equally distributed among subpopulations.

Li et al. [77], introduce an enhanced version of the mDOA from [76], named SPSO, by utilizing some key modifications. Firstly, to enhance global diversity, it utilizes an anti-convergence method from [13]. This method randomizes the worst-performing converged subpopulation when all subpopulations have converged. Subpopulation convergence is defined as the maximum Euclidean distance between the best individual and others, falling below a specific threshold. Secondly, for managing local diversity, the algorithm utilizes quantum individuals [47] to ensure continual diversity maintenance. Moreover, when subpopulation diversity falls below a predefined threshold, half of the non-optimal individuals are transformed into quantum particles within a defined radius to enhance local diversity. Lastly, the algorithm adopts a reevaluation-based approach for the top five best individuals of the entire population to detect environmental changes. This approach has been further refined in [61] with modifications to the SPSO components. In the improved SPSO (iSPSO), local diversity control is adjusted to be maintained only after environmental changes by converting neutral individuals into quantum ones for a single iteration, and the convergence status of subpopulations is calculated differently using the Euclidean distance between the two farthest individuals in each subpopulation.

The parent–child concept from [70] is utilized in [69], where a mDOA is proposed with a variable number of subpopulations but a fixed overall population size. In this algorithm, once the explorer converges, determined by the largest Euclidean distance between its individuals, it uses some of its best individuals to form the exploiter subpopulation, while the remaining individuals are randomized. This post-convergence randomization of the explorer contributes to increased global diversity. Additionally, the use of the exclusion method [47] aids in enhancing global diversity by randomizing redundant subpopulations in two situations. First, by randomizing the inferior exploiter if two of them enter into each other search range, and second, by randomizing the explorer if it gets closer to an exploiter than a predefined threshold. Local diversity within each exploiter is maintained over time by incorporating immigrant individuals that shows behavior similar to the quantum individuals

[47]. Computational resources are equally distributed among exploiters using the classic round robin method. By reevaluating the best-found positions within each subpopulation, changes are detected. This is followed by a response that involves reevaluating all solutions to address outdated memory.

Woldesenbet and Yen [78], introduce a mDOA that utilizes a density-based clustering method [79] to form subpopulations. This method clusters a fixed number of individuals based on their positions, where individuals within a predefined threshold distance are connected to form groups. These groups become subpopulations if their membership exceeds a predefined lower bound threshold. Any individual not belonging to a group is assigned to the closest one. This clustering process occurs after environmental changes, resulting in varying subpopulation sizes. Local diversity is increased after environmental changes through a variable relocation method. This method monitors individual progress in terms of fitness and position, using this information to estimate the severity of changes and adjust subpopulation distributions accordingly. Computational resources are distributed evenly among subpopulations using the classic round robin approach. The algorithm detects changes by reevaluating some individuals and responds by performing the variable relocation method on individuals, which changes their positions.

Li and Yang [80] introduce a mDOA that uses a clustering method on a fixed number of overall individuals at the beginning of each environment. This clustering algorithm starts with a randomized population, assigning each individual a single neighbor based on Euclidean distance to distribute them into sub-regions. Then, a bi-phase single-linkage hierarchical clustering process occurs. The first phase clusters individuals until each group has at least two members, forming temporary sub-populations, while the second phase refines these sub-populations based on distance and spatial size. Then, the neighborhood topology shifts to a global star configuration, where the best individual in each sub-population becomes the attractor. After an environmental change, the best positions of each sub-population is kept and the remaining individuals randomly redistributed throughout the search space. This approach not only enhances global diversity and exploration capability but also avoids the loss of local diversity. By utilizing the mutual exclusion method, if more than a predefined percentage of a sub-population's individuals lie inside the search region of another sub-population, they will be merged. To effectively manage computational resources, the converged subpopulations are deactivated until the end of the environment. The convergence of a subpopulation is determined by measuring its spatial size, which is defined as the average distance of all its members to the center.

The mDOA proposed in [80] has been further improved in [81] with several modifications. First, an upper bound limit is set for the maximum number of individuals in a subpopulation. Second, the method used for clustering is changed. Each individual initially forms its own cluster. Then the Euclidean distances between all pairs of clusters are calculated. Next, the closest clusters are identified, ensuring that their combined population does not exceed the specified upper limit, and are merged. After each merge, the distances between all clusters are updated based on the Euclidean distance between their closest individuals. This iterative process continues until all clusters contain at least two individuals. Third, after performing mutual exclusion, if the number of individuals exceeds the upper bound, the remaining inferior individuals are removed to prevent wastage of computational resources. Finally, a change detection component is also incorporated to detect environmental changes by reevaluating the best found position in each subpopulation.

Turky and Abdullah [82] further modified the [81], through two modifications. First, an explicit memory is added to store the best-found positions of subpopulations during convergence. The memory has a limited capacity, and if full, it replaces the worst archived solutions with new ones. Second, redundant individuals, through exclusion, are not removed but replaced with a solution from the explicit memory.

Daneshyari and Yen [83] introduce a mDOA where the total number of individuals and the number of individuals in each subpopulation are fixed. At the beginning of each environment, clustering is performed using the k-means algorithm. This method relies on the positions of the individuals, and the resulting subpopulations are distinct from each other. A migration strategy, is performed after environmental changes, aids in enhancing global diversity. By modifying some rules of the subpopulations' core optimizer, local diversity is maintain over time. Equal distribution of computational resources among subpopulations is achieved through the classical round-robin method. The proposed mDOA utilizes a cultural framework, which includes a memory for storing information about individuals and procedures. By re-evaluating positions in the cultural memory, changes can be detected. Change reaction involves re-evaluating individuals to address the outdated memory issue and boosting global diversity.

The mDOA proposed in [84] introduce a new method for clustering individuals. This clustering is done at every iteration and considers the position of the individuals to form subpopulations. First, for each individual, a predefined number of close individuals, measured by Euclidean distance, are connected to form a group. In this case, some individuals can be members of multiple groups. If this occurs, the groups with shared members will be merged, and the best individual among them becomes the cluster head. If two cluster heads become closer than a predefined threshold, they also merge. The total number of the individuals is fixed and there is a limitation on the maximum number of subpopulations and their members. Setting such thresholds controls the computational resources by avoiding overcrowding. Hence, the redundant inferior individuals or those from redundant subpopulations are randomized, help enhance global diversity. Allocation of computational resources among subpopulations follows a classic round robin method to ensure equal access.

Wang et al. [85] introduce a mDOA that employs a clustering method based on the positions and fitness of individuals. While the number of subpopulations can vary, the overall population size remains constant. In the proposed mDOA, there are two ways to increase global diversity. The first is by randomizing the converged subpopulations. To do so, the best-found position of a converged subpopulation is stored in explicit memory before the subpopulation is randomized. Convergence status is defined based on the average Euclidean distance of all members of a subpopulation to the center of that subpopulation. The second method uses an exclusion method to randomize redundant subpopulations. This exclusion applies to the best position of a subpopulation and the archived solutions in memory. If the Euclidean distance between them falls below a specific threshold, the subpopulation is randomized. To manage resource allocation, more resources are allocated to the best found position among all subpopulations using a local search operator. Changes are detected by reevaluating the best-found position of the entire population.

Li and Yang [66] introduce a framework that modifies [81]. This mDOA uses the same population managing approach, but the change detection component is removed. To address the outdated memory issue, all solutions are re-evaluated every iteration. In the proposed mDOA, global diversity is increased under various conditions. Herein, when the ratio of remaining individuals to the initial overall number falls below a specified threshold, a re-diversification process is initiated. During this process, the best positions found by converged subpopulations are stored in memory, and the remaining individuals are randomized.

Luo et al. [72] propose a multi-population clustering approach with a varying number of subpopulations, but the overall number of individuals remains constant. To form subpopulations, both the position and fitness of individuals are considered, and the population is clustered using a NBC method, which is based on graphs. Initially, distances between all individuals are calculated. Then, each individual is connected to its nearest better individual, creating a spanning tree. After that, the average of all edges in the tree is calculated, and edges longer than a threshold are removed. The remaining connected individuals form subpopulation. The mDOA also benefits from explicit memory [86], into which solutions are injected into the population after an environmental change, thus increasing global diversity. To maintain local diversity over time, Brownian individuals using a normal distribution are applied to the best-found position of each sub-population.

Zhu et al. [87] add an explicit memory component to [76], which aids in increasing global diversity after environmental changes. At the beginning of each environmental change, the algorithm identifies and marks sub-populations that have converged to local optima. If the number of marked sub-populations falls below a certain threshold, the best non-converged sub-populations are marked until the threshold is met. The best individuals from these marked sub-populations are then stored in explicit memory. If the memory reaches its capacity, existing solutions may be replaced by new ones based on predefined criteria. To update the population after changes, two methods are employed: the fittest memory solutions are distributed equally among sub-populations to replace their worst individuals, and a species seed detection procedure is used to distribute the fittest memory seeds among sub-populations similarly.

Liu et al. [88] utilize affinity propagation clustering (APC) to automatically generate subpopulations through a message-passing process. This approach considers each individual within the population as a potential exemplar (or cluster center). Subpopulations are then created based on the similarities between individuals, determined by the negative Euclidean distance, effectively grouping individuals into subpopulations. The overall population size remains constant; however, the number of subpopulations and their sizes can change. Exclusion method from [61] is utilized to avoid overlapping. Herein, when two subpopulations overlap, they are merged, and an equivalent number of the worst-performing individuals from the inferior subpopulation are randomized. This randomization of redundant individuals improves global diversity. The algorithm also incorporates a memory; when a subpopulation converges based on a threshold, it stores its best-found position in memory. A subpopulation is considered converged if the average Euclidean distance between individuals within a subpopulation and the subpopulation's centroid, the average position of all its members, is less than a threshold. When a change in the environment is detected, through fitness re-evaluation, the algorithm utilizes the information stored in memory about these exemplars to quickly relocate and adapt to the new conditions.

3.2 Heterogeneous mDOAs with varying population size

In [70], a multi-population approach, FMSO, is introduced, featuring an adaptive number of subpopulations to allow a variable overall population size while maintaining a predefined number of individuals within each subpopulation. FMSO utilizes a parent–children approach, creating heterogeneous subpopulations using different core optimizers. Here, the parent acts as the explorer, exploring the search space to locate promising regions. Once the parent converges, a child, as the exploiter, is created. This is done by randomizing individuals within a hyper-ball centered around the best-found position of the explorer population, defined by a radius. This radius determines the search area for each exploiter subpopulation, taking inspiration from SPSO [76]. The exploration also increased using the exclusion method; it randomizes any explorer's individual that enters an exploiter's search region. Additionally, if two exploiter subpopulations encroach upon each other's search regions, the inferior one is removed to save computational resources. To prevent overcrowding and efficiently manage resources, a maximum number of exploiters is set. When this limit is reached, if a new exploiter needs to be introduced, the one that has not improved the best-found position for a specified number of iterations is removed first. Alternatively, if no new exploiter is created but the cap is reached, the worst-performing one is randomized using the anti-convergence method [13]. Through the use of a classic round robin method, computational resources are evenly divided among subpopulations.

Kamosi et al. [89] present a modified version of the FMSO [70]. In the proposed mDOA, both the explorer and exploiter utilize the same core optimizer. The process starts with one explorer population. When the best-found position of the explorer improves, a hyperball within a predefined radius around its best-found position is formed. Any individuals inside this hyperball create the exploiter. If the number of individuals within the hyperball exceeds the exploiter population size, the best ones are chosen as the exploiter members. Otherwise, additional individuals are randomly generated within the hyperball. After that, all explorer individuals within the hyperball are randomized. Using the explorer, global diversity is maintained over time. The proposed mDOA also addresses the issue of local diversity by increasing it within each exploiter after an environmental change. Within the proposed mDOA, the exclusion method removes the inferior exploiters if two of them overlap. However, if the explorer and exploiter overlap, and the fitness of the best-found position of the explorer is better than that of the exploiter, the best-found position of the exploiter is replaced with that of the explorer, and only the best-found solution of the explorer is randomized. Allocation of computational resources among subpopulations follows a classic round-robin method to ensure equal access. Additionally, changes are detected through a reevaluation of the best-found position of each exploiter.

In [90], resource allocation is added to the approach described in [89], a hibernation method similar to the deactivation method from [80]. If a sub-population converges, meaning the maximum Euclidean distance between all pairs of individuals within that sub-population falls below a predefined threshold, and the difference in fitness between its best-found position and the best-found position of the entire population falls below another threshold, that sub-population is deactivated until the next environmental change.

Halder et al. [91] introduce a mDOA that employs k-means clustering, based on the positions of individuals. This clustering occurs every predetermined number of iterations. The overall population size and the number of subpopulations can change, and there are upper and lower bounds for the number of subpopulations. Additionally, there is a threshold for the number of individuals within each subpopulation. Controlling the number of subpopulations and subpopulation sizes helps prevent wastage of computational resources by avoiding overcrowding. The algorithm benefits from explicit memory. To further manage computational resources, when a subpopulation converges, its best-found position is stored in explicit memory, and then the subpopulation is removed. A subpopulation is considered converged if the average Euclidean distance between the individuals and the midpoint of all individuals is less than a threshold. Global diversity is increased under certain conditions. First, after each environmental change, the whole population is reinitialized. Second, the best-found position does not improve for a predefined set number of iterations, new individuals are initialized throughout the search space to form a new sub-population. Third, by utilizing exclusion, in the case of overlap between two subpopulations, they will be merged, and if the number of members exceeds the threshold, the redundant ones will be randomized. A change is detect by reevaluating a sentry.

Yazdani et al. [64] propose a mDOA with an adaptive number of subpopulations and a varying overall population size. The subpopulations differ from each other; one is a free or explorer subpopulation responsible for exploring the search space. When the explorer subpopulation is considered to have converged, a new subpopulation, termed an exploiter, is created. Then, the explorer is randomized to discover another promising region, thus increasing global diversity. The free subpopulation's convergence is determined by monitoring the fitness of its best-found position over a predefined number of previous iterations. If the difference is less than a threshold, it is considered to be converged. Moreover, randomizing the explorer subpopulation through the exclusion method [61], when it is within the exploiter's search range, further increases global diversity. Local diversity is enhanced after environmental changes through randomization; the individuals of each subpopulation are distributed in a limited area around their best-found position with a uniform distribution. Resource allocation is managed in several ways. First, by deactivating converged subpopulations, excluding the best one, until the next environmental change; second, by applying a local search operator on the best-found position of the best exploiter subpopulation; and third, by removing an inferior exploiter subpopulation when two exploiters enter each other's search range. In the proposed mDOA, change is detected by reevaluating a single individual.

Li et al. [32], propose a multi-population approach that uses a position-based clustering method to form subpopulations. The overall number of individuals and the number of subpopulations can change over time. Clustering is initiated when the average radius of non-stagnating sub-populations, those still actively exploring and improving solutions, falls below a predefined percentage of the search range, known as the diversity adjustment point. Initially, each individual forms a cluster. Then, the clustering process merges the closest clusters until the total distance within each cluster becomes smaller than the distance between clusters, resulting in distinctly different subpopulations. Computational resources are saved under two conditions. First, when a subpopulation has converged, it becomes inactive until the next diversity adjustment point occurs. A subpopulation is converged if the average distance of individuals to the subpopulation center is less than a threshold. Second, redundant inferior subpopulations are removed by exclusion method if two sub-populations have at least one individual overlapping in each other's search areas. When the diversity adjustment point is reached, individuals previously removed by the exclusion mechanism are reinitialized, along with any new individuals added the population if the size needed to be increased. This reinitialization of individuals increases global diversity. Moreover, using Brownian individuals with a Gaussian distribution, local diversity is maintained over time. The proposed mDOA is change-independent and do not use any specific method to detect a change.

Kordestani et al. [92] describe a multi-population approach with varying number of subpopulations and overall population sizes. In the proposed mDOA, the local diversity is maintain over time by modifying the core optimizer rule. Additionally, global diversity is preserved by introducing new subpopulations when all existing ones have converged. Herein, a subpopulation is considered converged when the fitness of its best-found position has not improved in the previous iteration. Resource allocation is managed in three ways. First, by employing a local search operator around the best position found among all subpopulations; second, by controlling the size of the subpopulations. This control is achieved by calculating the Euclidean distances between the best-found positions of all subpopulations and the global best-found position. Subpopulations closer to the global best position than a predefined percentage of the longest distance will receive more individuals, while the others will have fewer. Third, the exclusion method [61] is used to remove redundant inferior subpopulations in case of a collapse. Change detection is done by reevaluating three individuals, prompting a comprehensive reevaluation of all individuals to address the outdated memory issue.

The mDOA proposed in [37] utilizes the same clustering method as [45], where individuals are clustered based on their fitness and positions. The number of subpopulations adapts to the discovered promising regions. Clustering occurs at each iteration, and the subpopulation size is predetermined. Subpopulations differ in their tasks and are divided into tracker and non-tracker subpopulations. A non-tracker subpopulation becomes a tracker when it is considered to have converged. Convergence is detected if the maximum Euclidean distance between the best-found position in the subpopulation and any other member is less than a threshold. To increase global diversity, when all subpopulations have converged, new randomly initialized individuals are injected into the population. Local diversity is increased after an environmental change for tracker subpopulations by randomizing the positions of all members around the best-found position, within a subpopulation in a hyper-ball. The radius of this hyper-ball is determined by the estimated environment shift severity. Computational resources are managed in three ways. First, by introducing an innovative adaptive deactivation mechanism that systematically allocates computational resources to the subpopulations. This resource allocation component deactivates subpopulations that have sufficiently converged to their local optima, thus saving computational resources. These saved resources are then allocated to subpopulations that are still actively exploring and to the best-performing subpopulation. Second, by removing redundant subpopulations when they overlap, using the exclusion method. Third, by setting an upper limit for the maximum number of subpopulations to avoid overcrowding. If the number of species reaches this limit and all subpopulations have converged, an anti-convergence is employed to re-initialize the individuals of the subpopulations with the worst performance, which also improves global diversity. The algorithm is informed of environmental changes and does not have any specific component to detect changes.

4 Benchmarking methods for evaluating mDOAs

Benchmarks play a crucial role in evaluating and comparing the performance of various evolutionary optimization algorithms. These benchmarks consist of standardized test problems that allow researchers to assess the strengths and weaknesses of their algorithms under controlled and reproducible conditions. Selecting the right benchmarks is crucial for effectively evaluating and comparing mDOAs. In addition, utilizing the proper performance indicators is also crucial for evaluating the effectiveness of the algorithms in solving optimization problems. A performance indicator is an statistical indicator used to assess and compare the performance of different solutions or algorithms.

The following presents a review of state-of-the-art benchmark generators, along with commonly used performance indicators that measure the effectiveness of algorithms based on the fitness or error of obtained solutions.[1] For a comprehensive survey of dynamic benchmark generators and a complete list of performance indicators, researchers can refer to [26] and [25], respectively. In the last part of this section, the real-world applications of DOAs are reviewed.

4.1 Dynamic benchmark generators

Dynamic benchmark generators can be classified into two types, those that construct the problem landscape and those that generate dynamic datasets. The majority of research in this field employs dynamic landscape generators, characterized by solution spaces that feature "multiple moving peaks" [25]. These benchmark generators are favored for their ease of understanding and implementation, and offering high configurability to produce numerous dynamic scenarios with varied, controlled characteristics. Recently, a dynamic dataset generator has been introduced [95], designed specifically for creating dynamic datasets for clustering in dynamic environments. This new tool is capable of generating real-world morphological and dynamical characteristics that were not previously captured by dynamic landscape generators. In the subsequent sections, we will explore widely used moving-peaks-based dynamic landscape generators, as well as this innovative dynamic dataset generator.

4.1.1 Dynamic landscape generators

Constructed landscapes in moving peaks baseline functions are created by utilizing multiple elements. Typically, a $\max(\cdot)$ function is employed to establish the attraction basin for these components. Each element within the moving peaks baseline functions often includes a *peak*, the characteristics of which, like height and position, change with time. The Moving Peaks Benchmark (MPB), as introduced in [39], stands as one of the initial DOP benchmarks to utilize moving peaks baseline functions. The foundational function of MPB is described as follows:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \frac{h_i^{(t)}}{1 + w_i^{(t)} \sum_{j=1}^{d} \left(x_j - c_{i,j}^{(t)}\right)^2}, \tag{3}$$

where $c_{i,j}^{(t)}$ is the jth dimension of the center of ith peak in the tth environment ($\mathbf{c}_i^{(t)}$), $h_i^{(t)}$ and $w_i^{(t)}$ are the height and width of the ith peak in the tth environment, respectively, x_j is the jth dimension of a solution ($\mathbf{x}$) in a d-dimensional problem space, and m is the number of peaks. The second version of MPB, which is known as *Scenario 2* in the literature [2] uses a baseline function that generates landscapes with conical peaks:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \mathbf{x} - \mathbf{c}_i^{(t)} \right\| \right\}. \tag{4}$$

The Generalized Dynamic Benchmark Generator (GDBG) offers six different types of dynamic scenarios to create benchmark landscapes [96], including small step changes, large step changes, random changes, chaotic changes, recurrent changes, and recurrent changes with noise [25]. Among these scenarios, the baseline function for scenario F_1 of the GDBG, which is frequently utilized in the DOP literature, is the Dynamic Rotation Peak Benchmark Generator (DRPBG) [97], formulated as follows:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \frac{h_i^{(t)}}{1 + w_i^{(t)} \sqrt{\sum_{j=1}^{d} \frac{(\mathbf{x}_j - \mathbf{c}_{i,j}^{(t)})^2}{d}}}. \tag{5}$$

Yazdani et al. [98] introduce the Generalized Moving Peaks Benchmark (GMPB), the most state-of-the-art benchmark that produces landscapes by combining several components. Unlike the typical moving peaks baseline functions, the components generated by the GMPB baseline function can range from smooth to extremely uneven, from unimodal to multimodal, and from symmetric to asymmetric forms. Additionally, these components might have circular contours or could be significantly ill-conditioned. The baseline function used by GMPB is outlined as follows:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - \sqrt{\mathbb{T}\left(\left(\mathbf{x} - \mathbf{c}_i^{(t)}\right)^{\top} \mathbf{R}_i^{(t)\top}, i\right) \mathbf{W}_i^{(t)} \mathbb{T}\left(\mathbf{R}_i^{(t)}\left(\mathbf{x} - \mathbf{c}_i^{(t)}\right), i\right)} \right\}, \tag{6}$$

where $\mathbb{T}(\mathbf{y}, i) : \mathbb{R}^d \mapsto \mathbb{R}^d$ is calculated as

[1] Evolutionary Dynamic Optimization Laboratory (EDOLAB) [93] is an open-source [94] MATLAB platform that provides a comprehensive library, which includes the benchmark landscape generators and performance indicators reviewed in this section, as well as a collection of 25 DOAs. EDOLAB features two main modules. The Experimentation Module allows for extensive configuration and comparison of algorithms, and the Education Module is designed to visually demonstrate the optimization process and enhance learning about dynamic optimization challenges.

$$\mathbb{T}\left(y_j, i\right) = \begin{cases} \exp\left(\log(y_j) + \tau_i^{(t)}\left(\sin\left(\eta_{i,1}^{(t)}\log(y_j)\right) + \sin\left(\eta_{i,2}^{(t)}\log(y_j)\right)\right)\right) & \text{if } y_j > 0 \\ 0 & \text{if } y_j = 0 \\ -\exp\left(\log(|y_j|) + \tau_i^{(t)}\left(\sin\left(\eta_{i,3}^{(t)}\log(|y_j|)\right) + \sin\left(\eta_{i,4}^{(t)}\log(|y_j|)\right)\right)\right) & \text{if } y_j < 0, \end{cases} \quad (7)$$

where $\mathbf{W}_i^{(t)}$ represents a diagonal matrix of size $d \times d$, where the diagonal elements indicate the width of the ith component across different dimensions. The parameters $\eta_{i,k\in1,2,3,4}^{(t)}$ and $\tau_i^{(t)}$ characterize the irregularity of the ith component. $\mathbf{R}_i^{(t)}$ stands for the rotation matrix of the ith component in the tth environment. Finally, y_j refers to the jth dimension of the vector $\mathbf{y}$. Similar to other baseline functions involving moving peaks, GMPB offers control over the position, height, and width of individual components. Moreover, GMPB allows for variations in component width across different dimensions, providing control over the condition number of components. By employing a rotation matrix for each component, the degree of variable interactions within each component can be adjusted. Additionally, the irregularity degree and modality can be regulated through parameters $\tau^{(t)}$ and $\eta_{k\in1,2,3,4}^{(t)}$. Lastly, adjusting the values of $\eta_{k\in1,2,3,4}^{(t)}$ allows for control over the symmetry of the components, with symmetric components generated when all four η_k values are the same.

Li et al. [99] introduce the Free Peaks benchmark (FPs), which generates landscapes featuring multiple moving peaks. FPs divides the landscape into hypercubes, which define the basins of attraction for the peaks. This characteristic of FPs distinguishes it from other moving peaks baseline functions, which typically employ the $\max(\cdot)$ function to determine basin boundaries. Within each hypercube, there is precisely one peak, and its boundaries are restricted to that hypercube. Both the position of a component within a hypercube and the boundaries of the hypercube are adjustable. Additionally, [99] propose several single-peak baseline functions, such as the conical peak, to be utilized within each hypercube.

Figure 2 illustrates landscapes generated by reviewed benchmark. The parameter settings for the baseline functions in (3), (4), and (5) are the same among them for a fair comparison. These landscapes have five peaks, and the same values for height and width are used for all of them. The search range for (4) is $[-50, 50]$, and for (3) and (5), it is $[-5, 5]$. To generate peak positions in all baseline functions according to their search ranges, the same random seed numbers are used. Figure 2(d) shows a landscape created by baseline function (6). This landscape includes three components characterized by high irregularity and asymmetry, two of which are also ill-conditioned and rotated. Figure 2(e) illustrates a landscape generated using FPs [99], featuring three *conical* peaks, each defined by its surrounding hypercubes that dictate the basins of attraction.

4.1.2 Dynamic dataset generator

The Dynamic Dataset Generator (DDG) proposed in [95] is a benchmark generator tool designed for creating dynamic datasets with known and controllable characteristics, specifically for evaluating clustering algorithms in dynamic environments. The DDG simulates a broad range of dynamic scenarios using multiple dynamic Gaussian components (DGCs), enabling the systematic performance evaluation of clustering algorithms across diverse and realistic dynamic conditions. DDG utilizes these DGCs for data generation, with each DGC in the tth environment defined as follows:

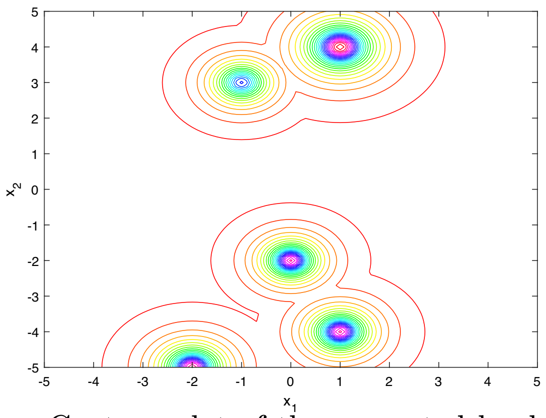
$$\mathcal{N}_i^{(t)}\left(\mathbf{c}_i^{(t)}, \boldsymbol{\sigma}_i^{(t)}, \boldsymbol{\Theta}^{(t)}\right) = \mathbf{r}\boldsymbol{\sigma}_i^{(t)}\mathbb{R}\left(\boldsymbol{\Theta}^{(t)}\right) + \mathbf{c}_i^{(t)}, \quad (8)$$

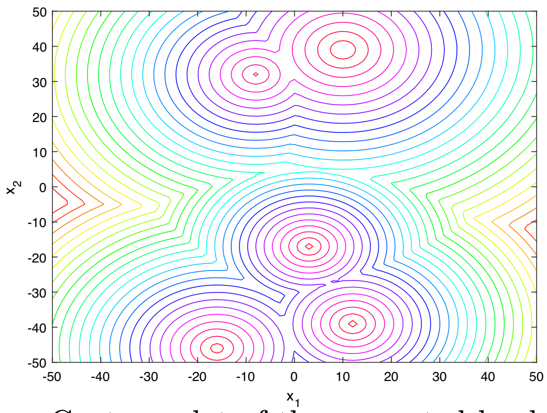
where at time $\mathcal{N}_i^{(t)}\left(\mathbf{c}_i^{(t)}, \boldsymbol{\sigma}_i^{(t)}, \boldsymbol{\Theta}^{(t)}\right)$ represents the ith DGC. The equation describes the relationship where $\mathbf{r}$ is a $d^{(t)}$-dimensional random vector, with each component drawn from a standard normal distribution $\mathcal{N}(0, 1)$. The vector $\mathbf{c}_i^{(t)}$ represents the mean or center position of the ith DGC within the tth environment, and $d^{(t)}$ denotes the number of variables at time t. Additionally, $\boldsymbol{\sigma}_i^{(t)}$ is a $d^{(t)}$-dimensional vector indicating the standard deviation, reflecting the spread of the ith DGC across each dimension. The $d^{(t)} \times d^{(t)}$ matrix $\boldsymbol{\Theta}_i^{(t)}$ is pivotal in determining the rotation of each component. In the context of DDG, $m^{(t)}$ DGCs are utilized for data generation, each assigned a weight $w_i^{(t)}$ governing the probability of generating a data point via the ith DGC. The DDG structure, utilizing DGCs, provides an easy approach for adjusting parameters to generate a range of dynamic scenarios. Parameters such as the number of DGCs ($m^{(t)}$), probability weights for data generation ($w_i^{(t)}$), DGCs' standard deviations ($\boldsymbol{\sigma}_i^{(t)}$), number of variables ($d^{(t)}$), rotation angles ($\theta_{i,j,k} = \boldsymbol{\Theta}_i^{(t)}(j,k)$), and center positions ($\mathbf{c}_i^{(t)}$) can all be modified over time through various patterns of change including random, chaotic, circular, or pendulum movements, as discussed in [25].

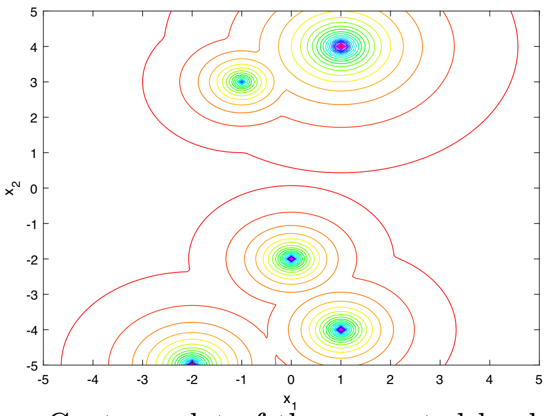
4.2 Performance indicators

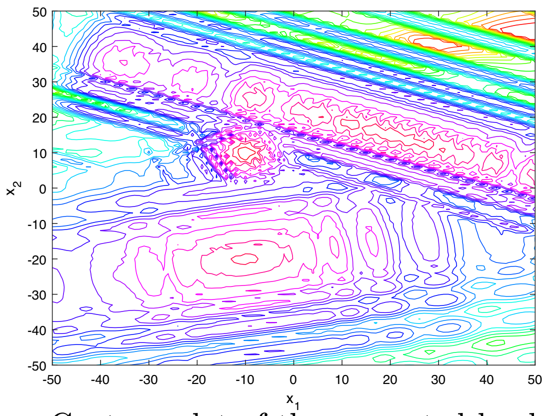
Performance indicators are crucial for assessing how effectively mDOAs adapt to changing environments, track moving optima, and maintain solution quality over time. Among the different types of performance indicators, those based on fitness or error are most commonly utilized in the DOP literature. These include offline error (E_O), best before change error (E_{BBC}), and offline performance indicator (P_O) [25], as described below.

When the details about the global optimum in each environment are known, it is possible to measure the error of


(a) Contour plot of the generated landscape by (3).


(b) Contour plot of the generated landscape by (4).


(c) Contour plot of the generated landscape by (5).


(d) Contour plot of the generated landscape by (6).

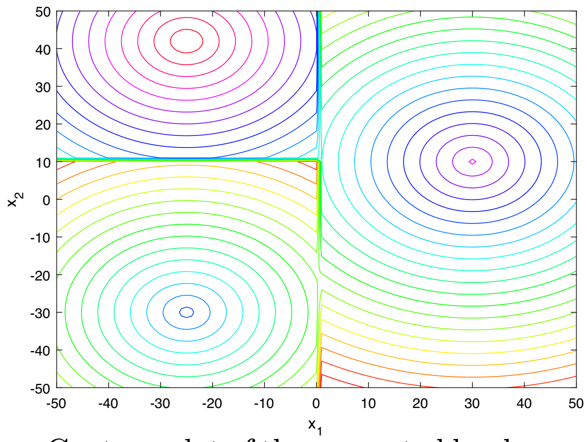

(e) Contour plot of the generated landscape by FPs.

Fig. 2 Landscapes generated by MPB (3), MPB Scenario 2 (4), DRPBG (5), GMPB (6), and FPs [25]

solutions. However, in practical real-world scenarios, accessing the global optimum's data is often unfeasible, though such information is mostly available in numerous DOP benchmarks. One performance indicator requiring the global optimum's details is E_O, as outlined by [2]. This indicator, widely referenced in academic studies, determines the mean error of the optimal position identified across all evaluations of fitness, according to a specific formula:

$$E_O = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta}\left(f^{(t)}\left(\mathbf{x}^{\star(t)}\right) - f^{(t)}\left(\mathbf{x}^{*((t-1)\vartheta+c)}\right)\right), \tag{9}$$

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

Another performance indicator is E_{BBC} [100], which considers only the final error at the end of each environment.

$$E_{\text{BBC}} = \frac{1}{T}\sum_{t=1}^{T}\left(f^{(t)}\left(\mathbf{x}^{\star(t)}\right) - f^{(t)}\left(\mathbf{x}^{*(t)}\right)\right), \tag{10}$$

where $\mathbf{x}^{*(t)}$, fetched at the end of the tth environment, represents the best-found position.

The E_O and E_{BBC} both require knowledge of the global optimum's position and fitness, which does not align well with practical dynamic optimization problems. Branke [39] proposed the offline-performance indicator (P_O), which does not require such details. Instead, it calculates the average fitness of the best position found across all evaluations, according to the provided formula:

$$P_O = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta} f^{(t)}\left(\mathbf{x}^{*((t-1)\vartheta+c)}\right). \tag{11}$$

For a fair assessment with P_O, it is crucial that environmental aspects of the problem, including peak features in the moving peaks baseline scenario, are kept constant across every algorithm being evaluated.

4.3 Real-world applications of DOAs

Exploring the application of DOAs in real-world scenarios offers a rich field for discovery and innovation. Various studies have demonstrated the practical utility of DOAs across different domains. Rakitianskaia et al. [101, 102] utilized DOAs to train supervised feed-forward neural networks, addressing dynamic classification challenges caused by concept drift. Similarly, Kalita and Singh [103] employed DOAs to optimize the hyper-parameters of support vector machines (SVMs) in dynamic environments, considering both gradual and batch data inflows. In the coal mining industry, Liu et al. [104] proposed a cutting pattern recognition method based on an SVM optimized by a DOA.

In the agricultural sector, Jin et al. [105] applied DOAs to develop adaptive farming strategies, maximizing income through optimal mixed grazing techniques. For dynamic load balancing, Sesum and Kuhn [106] employed DOAs to achieve optimal workload distribution among resources, enhancing system efficiency.

The work done in [107] utilize DOAs to optimize the parameters of a control strategy for a distribution static compensator on an all-electric ship power system, focusing on maintaining consistent bus voltage regulation. In the realm of robotics, Jatmiko et al. [108] applied DOAs to solve the Odor Source Localization problem, enabling mobile robots to locate chemical odor sources in dynamically changing environments.

In energy management, Wang et al. [109] utilized DOAs to tackle the dynamic economic dispatch problem, aiming to minimize operational costs in electrical power systems. Liu et al. [110] addressed contaminant source identification in water distribution networks using DOAs, where the search space evolved with new information.

The adaptability of DOAs in existing applications pave the way for their use in emerging fields. One key application is in dynamic facility location problems [111]. DOAs optimize facility placement and relocation to minimize costs and response times as demand and geographical factors change. This is crucial for strategic redeployment, such as relocating security forces in response to crowd dynamics or incidents. In real-time data analysis and concept drift management [112], DOAs are essential. They help maintain accurate clustering in fields like finance, marketing, and cybersecurity, where data patterns constantly evolve. Online deep clustering methods, for instance, adjust to new data without retraining, keeping data analysis relevant and precise. Resource allocation and management also benefit from DOAs [113]. They dynamically adjust resources to meet fluctuating demands in logistics, supply chain management, and smart city infrastructures. In disaster response, DOAs optimize the deployment of search and rescue operations, improving efficiency and outcomes in critical situations.

5 Future directions

Despite extensive research in the field, a significant gap remains between academic research and practical application. To bridge this gap, future research should focus on topics that narrow it. Considering the current state of research, we identify several potential future research directions aimed at reducing the disparity between academic research and their real-world applicability.

5.1 Adaptive parameter tuning over time

mDOAs typically comprise several components, each governed by its own set of parameters. Conventionally, these parameters operate based on fixed values, optimized according to the initial state of the problem. Given that commonly used benchmarks in the field often exhibit homogeneous behavior over time, this static tuning approach is usually sufficient; there is no compelling need to adjust parameters dynamically as the problem's characteristics remain consistent. However, real-world problems frequently show dynamic behavior, evolving significantly over time. This evolution calls for strategies that can adaptively modify parameter values in response to the current state of the problem. Consequently, the development of adaptive parameter tuning methods represents a crucial direction for future research, promising to bridge the gap between the static assumptions of academic models and the fluid dynamics of real-world applications.

5.2 Designing algorithms for problems with high temporal severity

Current mDOAs are primarily designed for scenarios where environmental changes occur at discrete intervals. Yet, the real world presents numerous challenges where changes unfold continuously over time. The current methodologies, being reactive in nature, activate certain mechanisms and components only in direct response to these changes. This approach, while effective for discrete adjustments, may prove detrimental in the face of continuous or highly frequent temporal changes, as the frequent triggering of reactions can lead to inefficiency. A crucial direction for future research lies in developing components and algorithms specifically designed to excel in environments that undergo continuous change.

5.3 Handling problems with multiple types of changes

The body of existing research has mainly concentrated on problems showing regular characteristics, including consistent frequencies and severities of changes, along with uniform dynamics during environmental changes. However, the real-world often presents scenarios where multiple types of environmental changes occur over time, each distinguished by its own spatial and temporal severities, patterns of change, and domains of influence. Current methods are designed to address a singular type of change, assuming static characteristics over time, and therefore fall short when confronted with multiple types of environmental changes. Addressing these challenges necessitates mDOAs capable of identifying the specific nature of each change and deploying appropriate responses designed for each situation. The development of such algorithms, equipped to differentiate and react suitably to various types of changes, represents a critical avenue for future research. This approach would allow for more effective handling of the complex dynamics characteristic of real-world problems.

5.4 Solution deployment and quick recovery

The design of most existing mDOAs does not take into account the time in which a new solution must be deployed. In real-world scenarios, there often exists a deadline or temporal constraint for selecting and deploying a solution following each environmental change, necessitating what is termed as "quick recovery" [3]. To enhance the optimization of the best-found solution before this deadline, mDOAs require strategic allocation of computational resources along with other mechanisms aimed at acceleration. Such considerations, however, have largely been overlooked in the current algorithmic designs within the field. Developing components for mDOAs that incorporate the need for quick recovery and efficient solution deployment under tight deadlines represents an important direction for future research.

5.5 Changes in the boundaries of the search range

One significant challenge that has not been investigated in the field is the change in the boundaries of the search space. This challenge poses a significant difficulty for mDOAs in many real-world problems where the search range, or at least the effective search range, changes over time. For example, in facility location problems, the effective search range includes areas where there are demands. A change in the distribution of demands causes the effective search range to change over time. When the search ranges change over time, they may not remain hypercubes. In fact, assuming hypercubes is a simplistic assumption. However, when there is a disparity between the range in different dimensions, many components of the algorithms that assume the search range is a hypercube face difficulties. For example, many mechanisms rely on Euclidean distances and hyperballs around specific points, such as the best-found solution, which may become ineffective when the search range is not a hypercube. One initial way to address this challenge is to normalize the ranges. However, normalizing the ranges itself introduces new dynamics into the system, which affects the algorithm. Therefore, developing methods that can dynamically adjust to non-hypercube search ranges without introducing additional complexities remains a crucial area for future research.

6 Conclusion

In this survey, we focused on the domain of single-objective unconstrained continuous dynamic optimization problems (DOPs), with a specific focus on multi-population dynamic optimization algorithms (mDOAs) as a main approach for tracking the moving optimum (TMO). We commenced a comprehensive review of mDOAs by providing a detailed analysis of mDOAs' components, their assembly into cohesive algorithms, and how these elements collectively create a framework for addressing TMO challenges. Additionally, we reviewed benchmarking methodologies, highlighting the use of dynamic benchmark generators and performance indicators to assess the effectiveness of mDOAs. By offering insights into the current state-of-the-art and suggesting avenues for future research, this survey aims to advance the understanding and development of mDOAs in the context of TMO in dynamic environments.

Funding This research is supported by a Liverpool John Moores University Vice-Chancellor PhD Scholarship.

Data availability This is a survey paper, with no source codes or data.

References

1. Jin, Y., & Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation, 9*(3), 303–317.
2. Branke, J., & Schmeck, H. (2003). Designing evolutionary algorithms for dynamic optimization problems. In: Ghosh, A., Tsutsui, S. (eds.) Advances in Evolutionary Computing, pp. 239–262. Springer.
3. Nguyen, T.T. (2011). Continuous dynamic optimisation using evolutionary algorithms. PhD thesis, University of Birmingham.
4. Kessels, J. T. B. A., Koot, M. W. T., Bosch, P. P. J., & Kok, D. B. (2008). Online energy management for hybrid electric vehicles. *IEEE Transactions on Vehicular Technology, 57*(6), 3428–3440.
5. Barlow, G. J., & Smith, S. F. (2008). A memory enhanced evolutionary algorithm for dynamic scheduling problems. In M. Giacobini, A. Brabazon, S. Cagnoni, G. A. Di Caro, R. Drechsler, A. Ekárt, A. I. Esparcia-Alcázar, M. Farooq, A. Fink, J. McCormack, M. O'Neill, J. Romero, F. Rothlauf, G. Squillero, A. Ş Uyar, & S. Yang (Eds.), *Applications of Evolutionary Computing* (pp. 606–615). Berlin, Heidelberg: Springer.
6. Yang, S., & Yao, X. (2005). Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing, 9*(4), 815–834.
7. Chai, R., Tsourdos, A., Savvaris, A., Chai, S., & Xia, Y. (2019). Two-stage trajectory optimization for autonomous ground vehicles parking maneuver. *IEEE Transactions on Industrial Informatics, 15*(7), 3899–3909.
8. Chai, R., Savvaris, A., Tsourdos, A., Chai, S., & Xia, Y. (2018). Unified multiobjective optimization scheme for aeroassisted vehicle trajectory planning. *Journal of Guidance, Control, and Dynamics, 41*(7), 1521–1530.
9. Chai, R., Savvaris, A., Tsourdos, A., Xia, Y., & Chai, S. (2020). Solving multiobjective constrained trajectory optimization problem by an extended evolutionary algorithm. *IEEE Transactions on Cybernetics, 50*(4), 1630–1643.
10. Nguyen, T. T., Yang, S., & Branke, J. (2012). Evolutionary dynamic optimization: a survey of the state of the art. *Swarm and Evolutionary Computation, 6*, 1–24.
11. Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: algorithms and applications. *Swarm and Evolutionary Computation, 33*, 1–17.
12. Yazdani, D. (2018). Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost. PhD thesis, Liverpool John Moores University, Liverpool, UK.
13. Blackwell, T., & Branke, J. (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation, 10*(4), 459–472.
14. Yang, S., Jiang, Y., & Nguyen, T. T. (2013). Metaheuristics for dynamic combinatorial optimization problems. *IMA Journal of Management Mathematics, 24*(4), 451–480.
15. Yazdani, D., Cheng, R., Yazdani, D., Branke, J., Jin, Y., & Yao, X. (2021). A survey of evolutionary continuous dynamic optimization over two decades - Part A. *IEEE Transactions on Evolutionary Computation, 25*(4), 609–629.
16. Raquel, C., & Yao, X. (2013). Dynamic multi-objective optimization: a survey of the state-of-the-art. In: Evolutionary Computation for Dynamic Optimization Problems, pp. 85–106. Springer.
17. Azzouz, R., Bechikh, S., & Said, L.B. (2017). Dynamic multi-objective optimization using evolutionary algorithms: a survey. In: Recent Advances in Evolutionary Multi-objective Optimization, pp. 31–70. Springer.
18. Azzouz, R. (2017). Evolutionary approaches for dynamic multi-objective optimization. PhD thesis, Computer Science Department, University of Tunis.
19. Yazdani, D., Yazdani, D., Branke, J., Omidvar, M.N., Amir H. Gandomi, & Yao, X. (2022). Robust optimization over time by estimating robustness of promising regions. IEEE Transactions on Evolutionary Computation 27(3), 657–670.
20. Yu, X., Jin, Y., Tang, K., & Yao, X. (2010). Robust optimization over time-a new perspective on dynamic optimization problems. In: Congress on Evolutionary Computation, pp. 1–6. IEEE.
21. Cruz, C., González, J. R., & Pelta, D. A. (2011). Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing, 15*(7), 1427–1448.
22. Yazdani, D., Omidvar, M.N., Yazdani, D., Branke, J., Nguyen, T.T., Gandomi, A.H., & Jin, Y., Yao, X. (2023). Robust optimization over time: A critical review. IEEE Transactions on Evolutionary Computation (Early Access, 2023).
23. Novoa-Hernández, P., Puris, A., & Pelta, D. A. (2023). Robust optimization over time problems-characterization and literature review. *Electronics, 12*(22), 4609.

24. Jiang, S., Zou, J., Yang, S., & Yao, X. (2022). Evolutionary dynamic multi-objective optimisation: a survey. *ACM Computing Surveys, 55*(4), 1–47.
25. Yazdani, D., Cheng, R., Yazdani, D., Branke, J., Jin, Y., & Yao, X. (2021). A survey of evolutionary continuous dynamic optimization over two decades - Part B. *IEEE Transactions on Evolutionary Computation, 25*(4), 630–650.
26. Yazdani, D., Branke, J., Omidvar, M.N., Li, X., Li, C., Mavrovouniotis, M., Nguyen, T.T., Yang, S., & Yao, X. (2021). IEEE CEC 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark. arXiv preprint arXiv:2106.06174
27. Novoa-Hernández, P., Corona, C. C., & Pelta, D. A. (2016). Self-adaptation in dynamic environments- a survey and open issues. *International Journal of Bio-Inspired Computation, 8*(1), 1–13.
28. Macias-Escobar, T., Dorronsoro, B., Cruz-Reyes, L., Rangel-Valdez, N., & Gómez-Santillán, C. (2020). A survey of hyper-heuristics for dynamic optimization problems. Intuitionistic and type-2 fuzzy logic enhancements in neural and optimization algorithms: Theory and applications, pp. 463–477.
29. Yazdani, D., Nguyen, T. T., & Branke, J. (2019). Robust optimization over time by learning problem space characteristics. *IEEE Transactions on Evolutionary Computation, 23*(1), 143–155.
30. Bu, C., Luo, W., & Yue, L. (2016). Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies. *IEEE Transactions on Evolutionary Computation, 21*(1), 14–33.
31. Branke, J. (2012). Evolutionary Optimization in Dynamic Environments vol. 3. Springer.
32. Li, C., Nguyen, T. T., Yang, M., Mavrovouniotis, M., & Yang, S. (2016). An adaptive multipopulation framework for locating and tracking multiple optima. *IEEE Transactions on Evolutionary Computation, 20*(4), 590–605.
33. Yang, S. (2006). Associative memory scheme for genetic algorithms in dynamic environments. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) Applications of Evolutionary Computing, pp. 788–799. Springer.
34. Yang, S. (2008). Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. *Evolutionary Computation 16*(3), 385–416.
35. Blackwell, T.M., & Bentley, P.J. (2002). Dynamic search with charged swarms. In: Conference on Genetic and Evolutionary Computation, pp. 19–26. Morgan Kaufmann Publishers Inc.
36. Das, S., Mandal, A., & Mukherjee, R. (2014). An adaptive differential evolution algorithm for global optimization in dynamic environments. *IEEE Transactions on Cybernetics, 44*(6), 966–978.
37. Yazdani, D., Yazdani, D., Yazdani, D., Omidvar, M. N., Gandomi, A. H., & Yao, X. (2023). A species-based particle swarm optimization with adaptive population size and deactivation of species for dynamic optimization problems. *ACM Transactions on Evolutionary Learning and Optimization, 3*(4), 1–25.
38. Yazdani, D., Cheng, R., He, C., & Branke, J. (2022). Adaptive control of subpopulations in evolutionary dynamic optimization. *IEEE Transactions on Cybernetics, 52*(7), 6476–6489.
39. Branke, J. (1999). Memory enhanced evolutionary algorithms for changing optimization problems. In: Congress on Evolutionary Computation *3*, 1875–1882. IEEE.
40. Hu, X., & Eberhart, R.C. (2002). Adaptive particle swarm optimization: detection and response to dynamic systems. In: Congress on Evolutionary Computation, *2*, 1666–1670. IEEE.
41. Floreano, D., & Nolfi, S. (1997). Adaptive behavior in competing co-evolving species. In: 4th European Conference on Artificial Life, 378–387.
42. Valenzuela, C. L., & Jones, A. J. (1993). Evolutionary divide and conquer (I): a novel genetic approach to the TSP. *Evolutionary Computation, 1*(4), 313–333.
43. Mendes, R., & Mohais, A.S. (2005). DynDE: a differential evolution for dynamic optimization problems. In: Congress on Evolutionary Computation, *3*, 2808–2815. IEEE.
44. Kundu, S., Basu, D., & Chaudhuri, S.S. (2013). Multipopulation-based differential evolution with speciation-based response to dynamic environments. In: Panigrahi et al., B.K. (ed.) Swarm, Evolutionary, and Memetic Computing, pp. 222–235. Springer.
45. Luo, W., Yi, R., Yang, B., & Xu, P. (2019). Surrogate-assisted evolutionary framework for data-driven dynamic optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence, 3*(2), 137–150.
46. Liu, X.-F., Zhou, Y.-R., Yu, X., & Lin, Y. (2019). Dual-archive-based particle swarm optimization for dynamic optimization. Applied Soft Computing, 105876.
47. Blackwell, T., & Branke, J. (2004). Multi-swarm optimization in dynamic environments. In: Raidl et al., G.R. (ed.) Applications of Evolutionary Computing, *3005*, pp. 489–500. Lecture Notes in Computer Science.
48. du Plessis, M.C., & Engelbrecht, A.P. (2008). Improved differential evolution for dynamic optimization problems. In: Congress on Evolutionary Computation, pp. 229–234. IEEE.
49. Brest, J., Zamuda, A., Boskovic, B., Maucec, M.S., & Zumer, V. (2009). Dynamic optimization using self-adaptive differential evolution. In: Congress on Evolutionary Computation, pp. 415–422. IEEE.
50. Plessis, M. C., & Engelbrecht, A. P. (2012). Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research, 218*(1), 7–20.
51. Bose, D., Biswas, S., Kundu, S., & Das, S. (2012). A strategy pool adaptive artificial bee colony algorithm for dynamic environment through multi-population approach. In: Panigrahi et al., B.K. (ed.) Swarm, Evolutionary, and Memetic Computing, pp. 611–619. Springer.
52. Zuo, X., & Xiao, L. (2013). A de and pso based hybrid algorithm for dynamic optimization problems. *Soft Computing, 18*(7), 1405–1424.
53. Novoa-Hernández, P., Corona, C. C., & Pelta, D. A. (2013). Self-adaptive, multipopulation differential evolution in dynamic environments. *Soft Computing, 17*(10), 1861–1881.
54. Vafashoar, R., & Meybodi, M.R. (2019). A multi-population differential evolution algorithm based on cellular learning automata and evolutionary context information for optimization in dynamic environments. Applied Soft Computing, 106009.
55. Novoa-Hernández, P., Pelta, D.A., & Corona, C.C. (2010). In: González et al., J.R. (ed.) Improvement Strategies for Multi-swarm PSO in Dynamic Environments, pp. 371–383. Springer.
56. Raghul, S., & Jeyakumar, G. (2023). A hybrid multi-population reinitialization strategy to tackle dynamic optimization problems. *IEEE Access, 11*, 114270–114282.
57. Blackwell, T. (2007). In: Yang, S., Ong, Y.-S., Jin, Y. (eds.) Particle Swarm Optimization in Dynamic Environments, pp. 29–49. Springer.
58. Rezazadeh, I., Meybodi, M.R., & Naebi, A. (2011). Adaptive particle swarm optimization algorithm in dynamic environments. In: Computational Intelligence, Modelling and Simulation, pp. 74–79. IEEE.
59. Rezazadeh, I., Meybodi, M.R., & Naebi, A. (2011). Particle swarm optimization algorithm in dynamic environments: Adapting inertia weight and clustering particles. In: European Symposium on Computer Modeling and Simulation, pp. 76–82. IEEE.
60. Sepas-Moghaddam, A., Arabshahi, A., Yazdani, D., & Dehshibi, M.M. (2012). A novel hybrid algorithm for optimization in

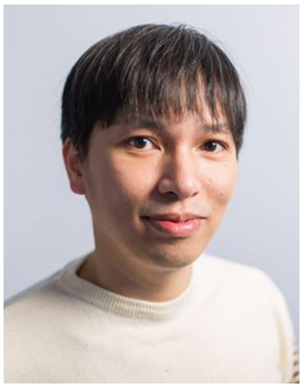
multimodal dynamic environments. In: International Conference on Hybrid Intelligent Systems, pp. 143–148. IEEE.
61. Blackwell, T., Branke, J., & Li, X. (2008). Particle swarms for dynamic optimization problems. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence: Introduction and Applications, pp. 193–217. Springer.
62. Yazdani, D., Akbarzadeh-Totonchi, M.R., Nasiri, B., & Meybodi, M.R. (2012). A new artificial fish swarm algorithm for dynamic optimization problems. In: Congress on Evolutionary Computation, pp. 1–8. IEEE.
63. Yazdani, D., Nasiri, B., Sepas-Moghaddam, A., Meybodi, M., & Akbarzadeh-Totonchi, M. (2014). mNAFSA: a novel approach for optimization in dynamic environments with global changes. *Swarm and Evolutionary Computation, 18*, 38–53.
64. Yazdani, D., Nasiri, B., Sepas-Moghaddam, A., & Meybodi, M. R. (2013). A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing, 13*(4), 2144–2158.
65. du Plessis, M. C., & Engelbrecht, A. P. (2013). Differential evolution for dynamic environments with unknown numbers of optima. *Journal of Global Optimization, 55*(1), 73–99.
66. Li, C., & Yang, S. (2012). A general framework of multipopulation methods with clustering in undetectable dynamic environments. *IEEE Transactions on Evolutionary Computation, 16*(4), 556–577.
67. Li, C., Yang, S., & Yang, M. (2014). An adaptive multi-swarm optimizer for dynamic optimization problems. *Evolutionary Computation, 22*(4), 559–594.
68. Qin, J., Huang, C., & Luo, Y. (2021). Adaptive multi-swarm in dynamic environments. *Swarm and Evolutionary Computation, 63*, 100870.
69. Wang, H., Wang, N., & Wang, D. (2008). Multi-swarm optimization algorithm for dynamic optimization problems using forking. In: Control and Decision Conference, pp. 2415–2419. IEEE.
70. Li, C., & Yang, S. (2008). Fast multi-swarm optimization for dynamic optimization problems. In: International Conference on Natural Computation, vol. 7, pp. 624–628. IEEE.
71. Moradi, M., Nejatian, S., Parvin, H., & Rezaie, V. (2018). Cmcabc: clustering and memory-based chaotic artificial bee colony dynamic optimization algorithm. *International Journal of Information Technology & Decision Making, 17*(04), 1007–1046.
72. Luo, W., Yang, B., Bu, C., & Lin, X. (2017). A hybrid particle swarm optimization for high-dimensional dynamic optimization. In: Shi et al., Y. (ed.) Simulated Evolution and Learning, pp. 981–993. Springer, Cham.
73. Biswas, S., Bose, D., & Kundu, S. (2012). A clustering particle based artificial bee colony algorithm for dynamic environment. In: Panigrahi et al., B.K. (ed.) Swarm, Evolutionary, and Memetic Computing, pp. 151–159. Springer.
74. Oppacher, F., & Wineberg, M. (1999). The shifting balance genetic algorithm: Improving the ga in a dynamic environment. In: Conference on Genetic and Evolutionary Computation, *1*, 504–510. ACM.
75. Branke, J., Kaussler, T., Schmidt, C., & Schmeck, H. (2000). A multi-population approach to dynamic optimization problems. In: Evolutionary Design and Manufacture, pp. 299–307. Springer.
76. Parrott, D., & Li, X. (2004). A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In: Congress on Evolutionary Computation, *1*, 98–103. IEEE.
77. Li, X., Branke, J., & Blackwell, T. (2006). Particle swarm with speciation and adaptation in a dynamic environment. In: Conference on Genetic and Evolutionary Computation, pp. 51–58. ACM.
78. Woldesenbet, Y. G., & Yen, G. G. (2009). Dynamic evolutionary algorithm with variable relocation. *IEEE Transactions on Evolutionary Computation, 13*(3), 500–513.
79. Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Knowledge Discovery and Data Mining, pp. 226–231. AAAI Press.
80. Li, C., & Yang, S. (2009). A clustering particle swarm optimizer for dynamic optimization. In: Congress on Evolutionary Computation, pp. 439–446. IEEE.
81. Yang, S., & Li, C. (2010). A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation, 14*(6), 959–974.
82. Turky, A. M., & Abdullah, S. (2014). A multi-population harmony search algorithm with external archive for dynamic optimization problems. *Information Sciences, 272*, 84–95.
83. Daneshyari, M., & Yen, G.G. (2011). Dynamic optimization using cultural based pso. In: Congress of Evolutionary Computation, pp. 509–516. IEEE.
84. Nickabadi, A., Ebadzadeh, M. M., & Safabakhsh, R. (2012). A competitive clustering particle swarm optimizer for dynamic optimization problems. *Swarm Intelligence, 6*(3), 177–206.
85. Wang, H., Yang, S., Ip, W. H., & Wang, D. (2012). A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems. *International Journal of Systems Science, 43*(7), 1268–1283.
86. Luo, W., Sun, J., Bu, C., & Liang, H. (2016). Species-based particle swarm optimizer enhanced by memory for dynamic optimization. *Applied Soft Computing, 47*, 130–140.
87. Zhu, T., Luo, W., & Yue, L. (2014). Combining multipopulation evolutionary algorithms with memory for dynamic optimization problems. In: Congress on Evolutionary Computation, pp. 2047–2054. IEEE.
88. Liu, Y., Liu, J., Jin, Y., Li, F., & Zheng, T. (2020). An affinity propagation clustering based particle swarm optimizer for dynamic optimization. *Knowledge-Based Systems, 195*, 105711.
89. Kamosi, M., Hashemi, A.B., & Meybodi, M.R. (2010). A new particle swarm optimization algorithm for dynamic environments. In: Panigrahi et al., B.K. (ed.) Swarm, Evolutionary, and Memetic Computing, pp. 129–138. Springer.
90. Kamosi, M., Hashemi, A.B., & Meybodi, M.R. (2010). A hibernating multi-swarm optimization algorithm for dynamic environments. In: Nature and Biologically Inspired Computing, pp. 363–369. IEEE.
91. Halder, U., Maity, D., Dasgupta, P., & Das, S. (2011). Self-adaptive cluster-based differential evolution with an external archive for dynamic optimization problems. In: Panigrahi et al., B.K. (ed.) Swarm, Evolutionary, and Memetic Computing, pp. 19–26. Springer, Berlin, Heidelberg.
92. Kordestani, J. K., Firouzjaee, H. A., & Reza Meybodi, M. (2017). An adaptive bi-flight cuckoo search with variable nests for continuous dynamic optimization problems. *Applied Intelligence, 48*(1), 97–117.
93. Peng, M., She, Z., Yazdani, D., Yazdani, D., Luo, W., Li, C., Branke, J., Nguyen, T.T., Gandomi, A.H., Jin, Y., et al. (2023). Evolutionary dynamic optimization laboratory: A matlab optimization platform for education and experimentation in dynamic environments. arXiv preprint arXiv:2308.12644
94. Peng, M., She, Z., Yazdani, D., Yazdani, D., Luo, W., Li, C., Branke, J., Nguyen, T.T., Gandomi, A.H., Jin, Y., et al. (2023). Evolutionary Dynamic Optimization Laboratory (EDOLAB). GitHub repository. https://github.com/Danial-Yazdani/EDOLAB-MATLAB
95. Yazdani, D., Branke, J., Khorshidi, M.S., Omidvar, M.N., Li, X., Gandomi, A.H., & Yao, X. (2024). Clustering in dynamic environments: a framework for benchmark dataset generation with heterogeneous changes.

96. Li, C., & Yang, S. (2008). A generalized approach to construct benchmark problems for dynamic optimization. In: Simulated Evolution and Learning, pp. 391–400. Springer.
97. Li, C., Yang, S., Nguyen, T. T., Yu, E. L., Yao, X., Jin, Y., Beyer, H.-G., & Suganthan, P. N. (2008). *Benchmark generator for cec'2009 competition on dynamic optimization*. Center for Computational Intelligence: Technical report.
98. Yazdani, D., Omidvar, M. N., Cheng, R., Branke, J., Nguyen, T. T., & Yao, X. (2022). Benchmarking continuous dynamic optimization: survey and generalized test suite. *IEEE Transactions on Cybernetics, 52*(5), 3380–3393.
99. Li, C., Nguyen, T.T., Zeng, S., Yang, M., & Wu, M. (2018). An open framework for constructing continuous optimization problems. IEEE Transactions on Cybernetics, 1–15
100. Trojanowski, K., & Michalewicz, Z. (1999). Searching for optima in non-stationary environments. *Congress on Evolutionary Computation, 3*, 1843–1850.
101. Rakitianskaia, A. & Engelbrecht, A.P. (2009). Training neural networks with pso in dynamic environments. In: Congress on Evolutionary Computation, pp. 667–673. IEEE.
102. Rakitianskaia, A. S., & Engelbrecht, A. P. (2012). Training feedforward neural networks with dynamic particle swarm optimisation. *Swarm Intelligence, 6*(3), 233–270.
103. Kalita, D. J., & Singh, S. (2020). Svm hyper-parameters optimization using quantized multi-pso in dynamic environment. *Soft Comput., 24*(2), 1225–1241.
104. Liu, X., He, S., Gu, Y., Xu, Z., Zhang, Z., Wang, W., & Liu, P. (2020). A robust cutting pattern recognition method for shearer based on least square support vector machine equipped with chaos modified particle swarm optimization and online correcting strategy. *ISA transactions, 99*, 199–209.
105. Jin, N., Termansen, M., Hubacek, K., Holden, J., & Kirkby, M. (2007). Adaptive farming strategies for dynamic economic environment. In: Congress on Evolutionary Computation, pp. 1213–1220. IEEE.
106. Sesum-Cavic, V., & Kuhn, E. (2010). Comparing configurable parameters of swarm intelligence algorithms for dynamic load balancing. In: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop, pp. 42–49. IEEE.
107. Mitra, P., & Venayagamoorthy, G. K. (2009). An adaptive control strategy for dstatcom applications in an electric ship power system. *IEEE Transactions on power electronics, 25*(1), 95–104.
108. Jatmiko, W., Nugraha, A., Effendi, R., Pambuko, W., Mardian, R., Sekiyama, K., & Fukuda, T. (2009). Localizing multiple odor sources in a dynamic environment based on modified niche particle swarm optimization with flow of wind. *WSEAS Transactions on Systems, 8*(11), 1187–1196.
109. Wang, Y., Zhou, J., Lu, Y., Qin, H., & Wang, Y. (2011). Chaotic self-adaptive particle swarm optimization algorithm for dynamic economic dispatch problem with valve-point effects. *Expert Systems with Applications, 38*(11), 14231–14237.
110. Liu, L., Ranjithan, S. R., & Mahinthakumar, G. (2011). Contamination source identification in water distribution systems using an adaptive dynamic optimization procedure. *Journal of Water Resources Planning and Management, 137*(2), 183–192.
111. Karatas, M. (2021). A dynamic multi-objective location-allocation model for search and rescue assets. *European Journal of Operational Research, 288*(2), 620–633.
112. Moulton, R.H., Viktor, H.L., Japkowicz, N., & Gama, J. (2019). Clustering in the presence of concept drift. In: Machine Learning and Knowledge Discovery in Databases, pp. 339–355. Springer.
113. Li, T., Chen, L., Jensen, C.S., Pedersen, T.B., Gao, Y., & Hu, J. (2022). Evolutionary clustering of moving objects. In: International Conference on Data Engineering, pp. 2399–2411. IEEE.

Delaram Yazdani received her M.Sc. degree in Information Technology Engineering from Azad University, Iran, in 2019. She is currently pursuing a Ph.D. degree at the Faculty of Engineering and Technology, Liverpool John Moores University, Liverpool, U.K. Her research interests include optimization and evolutionary in dynamic environments, operational research, and optimization solutions for port operations and logistics.

Danial Yazdani received his Ph.D. degree in computer science from Liverpool John Moores University, Liverpool, U.K., in 2018. He is currently a Research Fellow at the Data Science Institute, University of Technology Sydney. Prior to that, he was a Research Assistant Professor with the Department of Computer Science and Engineering at the Southern University of Science and Technology, Shenzhen, China. His primary research interests include learning and optimization in dynamic environments, where he has contributed as the first author in over 20 peer reviewed publications in this field, nine of which were published in prestigious IEEE/ACM Transactions. He is an IEEE Senior Member and was a recipient of the 2023 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award, the Best Thesis Award from the Faculty of Engineering and Technology at Liverpool John Moores University, and the SUSTech Presidential Outstanding Postdoctoral Award from Southern University of Science and Technology.

Eddie Blanco-Davis is a Reader in Marine Engineering at the Faculty of Engineering and Technology, Liverpool John Moores University, Liverpool, U.K. He started his position as a Senior Lecturer and a member of the Liverpool Logistics Offshore and Marine Research Institute (LOOM) in 2015. He earned his B.Sc. in Marine Engineering and Shipyard Management from the United States Merchant Marine Academy, Kings Point, N.Y., in 2004. He holds an M.Sc. and a Ph.D. in Naval Architecture, Ocean, and Marine Engineering from the University of Strathclyde, Glasgow. His research interests include human factors applied to shipboard and pilotage operations; Life Cycle Assessment applied to marine operations and machinery; sustainable development, energy efficiency, and renewable energy; marine environmental protection and regulation, including emission control, ship ballast water management and alternatives, and environmentally sound marine propulsion systems.



Trung Thanh Nguyen is a Professor of Operational Research in Transportation and Co-Director of the Liverpool Logistics, Offshore and Marine (LOOM) Research Institute at Liverpool John Moores University, Liverpool, U.K. He leads the Smart Green Things group, focusing on optimization, simulation, data analytics, and machine learning for various transportation and smart city applications. He has received multiple awards, including the LJMU Vice-Chancellor's Award for Excellence in Research Impact in 2023. His recent work includes projects on zero-emission vessels, smart city solutions, and traffic congestion mitigation.