

## Algorithm xxx: EDOLAB, a Platform for Research and Education in Evolutionary Dynamic Optimization

MAI PENG\*, School of Automation, China University of Geosciences, Wuhan, Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, and Engineering Research Center of Intelligent Technology for Geo-Exploration, Ministry of Education, China

DELARAM YAZDANI\*, Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Faculty of Health, Innovation, Technology and Science, Liverpool John Moores University, United Kingdom

DANIAL YAZDANI\*, School of Computing Technologies, RMIT University, Australia

ZENENG SHE\*, School of Computer Science and Technology, Harbin Institute of Technology, China

WENJIAN LUO\*, Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Institute of Cyberspace Security, School of Computer Science and Technology, Harbin Institute of Technology, China

CHANGHE LI\*, School of Artificial Intelligence, Anhui University of Science & Technology, State Key Laboratory of Digital Intelligent Technology for Unmanned Coal Mining, Anhui University of Science & Technology, China

JUERGEN BRANKE, Warwick Business School, University of Warwick, United Kingdom

TRUNG THANH NGUYEN, The Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Faculty of Health, Innovation, Technology and Science, Liverpool John Moores University, United Kingdom

AMIR H. GANDOMI†, Faculty of Engineering & Information Technology, University of Technology Sydney, Australia and University Research and Innovation Center (EKIK), Obuda University, Hungary

SHENGXIANG YANG, Digital Future Institute, School of Computer Science and Informatics, De Montfort University, United Kingdom

YAOCHU JIN, Department of Artificial Intelligence, School of Engineering, Westlake University, China

XIN YAO†, School of Data Science, Lingnan University, China

---

\*Equal contribution

†Corresponding authors

---

This work was supported by a National Natural Science Foundation of China (Grant No. 62250710682), a Lingnan University internal grant, a Shenzhen Fundamental Research Program (Grant No. JCYJ20220818102414030), a Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (Grant No. 2022B1212010005), a National Natural Science Foundation of China (Grant No. 62476006), a Hubei Provincial Natural Science Foundation of China (Grant No. 2023AFA049), a Fundamental Research Funds of the AUST (Grant No. 2024JBZD0007), a National Natural Science Foundation of China (Grant No. U23B2058), an NSFC ICFCRT (Grant No. W2441019), an NSFC (Grant No. 62136003), a Liverpool John Moores University Vice-Chancellor PhD Scholarship, and Australian Government through the Australian Research Council under Project DE210101808.

Authors' Contact Information: Mai Peng, pengmai1998@gmail.com, School of Automation, China University of Geosciences, Wuhan, Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, and Engineering Research Center of Intelligent Technology for Geo-Exploration, Ministry of Education, China; Delaram Yazdani, delaram.yazdani@yahoo.com, Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Faculty of Health, Innovation, Technology and Science, Liverpool John Moores University, Liverpool, United Kingdom; Danial Yazdani, danial.yazdani@gmail.com, School of Computing Technologies, RMIT University, Melbourne, Australia; Zeneng She, 20s151103@stu.hit.edu.cn, School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China; Wenjian Luo, luowenjian@hit.edu.cn, Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Institute of Cyberspace Security, School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China; Changhe Li, changhe.li@gmail.com, School of Artificial Intelligence, Anhui University of Science & Technology, State Key Laboratory of Digital Intelligent Technology for Unmanned Coal Mining, Anhui University of Science & Technology, Hefei, China; Juergen Branke, Juergen.Branke@wbs.ac.uk, Warwick Business School, University of Warwick, Coventry, United Kingdom; Trung Thanh Nguyen, T.T.Nguyen@ljmu.ac.uk,

**Abstract**— Many real-world optimization problems exhibit dynamic characteristics, posing significant challenges for traditional optimization methods. Evolutionary Dynamic Optimization Algorithms (EDOAs) have been developed to address these challenges by adapting to changing environments over time. However, the reproducibility and consistency of experimental results in the literature remain limited due to the lack of publicly available source codes and the complexity of accurately re-implementing algorithms and performance evaluation protocols. To support the community, we introduce EDOLAB (*Evolutionary Dynamic Optimization LABoratory*), an open-source MATLAB platform designed for both research and educational purposes. EDOLAB includes 27 EDOAs, four highly configurable benchmark generators, and a growing suite of performance indicators. The platform supports full parameter tuning, batch experiment management, parallel execution, and automated statistical comparisons—including rankings, significance testing, box plots, and performance trend visualizations over time. An *educational application* allows users to observe: a) dynamic changes in a 2D problem landscape, b) the movement of individuals in response to these changes, and c) the ability of an algorithm to track moving optima. By providing an integrated environment for experimentation, benchmarking, and instructional use, EDOLAB promotes reproducibility, comparative analysis, and a deeper understanding of EDOAs in dynamic environments.

CCS Concepts: • **Applied computing** → **Interactive learning environments**; • **General and reference** → **Experimentation**.

Additional Key Words and Phrases: Dynamic optimization problems, evolutionary dynamic optimization, benchmarking platform, educational tools, algorithm analysis, reproducible research, MATLAB software.

## 1 Introduction

### 1.1 Motivation and Background

Many real-world optimization problems are dynamic in nature [Nguyen 2011], meaning that the characteristics of their search spaces change over time [Raquel and Yao 2013; Yazdani et al. 2021b]. These environmental changes in dynamic optimization problems (DOPs) introduce uncertainties that must be accounted for by the optimization algorithm [Jin and Branke 2005]. To effectively solve DOPs, it is important that the optimization algorithm not only locates an optimal solution efficiently but also continues to track it as the environment changes. This capability is referred to as tracking the moving optimum (TMO) [Nguyen et al. 2012].

Evolutionary algorithms and swarm intelligence methods are popular and effective optimization tools, originally designed for solving static optimization problems. Applying these tools directly to TMO in DOPs proves ineffective because they fail to account for environmental changes. These changes in DOPs present several challenges, including: a) outdated stored fitness values (also known as objective function values), b) local and global diversity loss, and c) a limited number of objective function evaluations that can be conducted between consecutive environmental changes (i.e., within each environment) [Yazdani et al. 2020a].

To cope with these challenges, evolutionary algorithms and swarm intelligence methods are typically augmented with additional components, forming evolutionary dynamic optimization algorithms (EDOAs). These components may include local and global diversity control, explicit archives, change detection and reaction mechanisms, population clustering and management, exclusion, convergence detection, and computational resource allocation [Yazdani et al. 2021b]. Consequently, EDOAs often become complex algorithms.

---

The Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Faculty of Health, Innovation, Technology and Science, Liverpool John Moores University, Liverpool, United Kingdom; Amir H. Gandomi, Gandomi@uts.edu.au, Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo, Australia and University Research and Innovation Center (EKIK), Obuda University, Budapest, Hungary; Shengxiang Yang, syang@dmu.ac.uk, Digital Future Institute, School of Computer Science and Informatics, De Montfort University, Leicester, United Kingdom; Yaochu Jin, jinyaochu@westlake.edu.cn, Department of Artificial Intelligence, School of Engineering, Westlake University, Hangzhou, China; Xin Yao, xinyao@ln.edu.hk, School of Data Science, Lingnan University, Hong Kong SAR, China.

This complexity also poses a challenge to reproducibility, as even minor changes or mistakes during re-implementation can significantly impact performance. In the absence of publicly available source codes, researchers often spend a significant amount of time and effort attempting to re-implement complex EDOAs based solely on their textual descriptions. This process is not only time-consuming and error-prone, but also discouraging for newcomers to the field, potentially limiting engagement and innovation.

Moreover, such algorithm descriptions often omit low-level implementation details that are crucial for achieving the intended behavior and performance. As a result, researchers must make assumptions about important aspects of the algorithm, which may diverge significantly from the original intentions. These implementation-sensitive behaviors are rarely explained in full, as papers naturally focus on high-level concepts rather than operational minutiae.

Meaningful statistical comparisons require access to the full set of run results for both the proposed algorithm and its baselines. Relying solely on published summary statistics—such as mean and standard deviation—is insufficient for conducting hypothesis testing or variance-based analyses. However, in many studies, the lack of available reference implementations leads authors to compare their proposed algorithms against reported averages from prior work without re-implementing the baselines. This practice undermines the validity of the statistical analysis. Although some estimation techniques exist to reconstruct missing data, they generally lack the statistical power to support robust conclusions. Even when re-implementations are attempted, subtle discrepancies can produce significantly different outcomes. For instance, in the Moving Peaks Benchmark (MPB) [Branke 1999], some implementations initialize all peaks with uniform height, while others randomize them—resulting in notably different performance during early environments. Similarly, we have observed inconsistent use of iteration-based and fitness-evaluation-based performance indicators across comparative studies [Yazdani et al. 2021c].

Another example of source of divergence lies in how and when environmental changes are handled. Since performance typically degrades immediately after a change, even minor differences—such as triggering change reactions at the end of an iteration, after a sub-population update, between internal mechanisms, or immediately upon detection—can yield statistically significant differences in performance. While this issue is separate from change detection itself (as many real-world problems assume informed changes), it nonetheless highlights the critical role of timing and synchronization in dynamic optimization algorithms.

Finally, the improper or inconsistent use of random seeds can lead to drastically different benchmark instances. Despite having predefined parameter ranges, randomized landscape configurations can produce instances with significantly different difficulty levels—e.g., a narrow global optimum hidden in a corner, surrounded by wide deceptive peaks—thus leading to unstable or irreproducible results.

The lack of standardization in these aspects has made meaningful comparisons more challenging and has limited the pace of progress. These challenges highlight the need for a platform that ensures standardization, promotes reproducibility, and provides accessible and consistent experimental environments for evaluating EDOAs. Such a platform would facilitate fair and consistent comparisons across studies, reduce the overhead of re-implementation, and significantly lower the entry barrier for new researchers entering the field of evolutionary dynamic optimization.

## 1.2 About EDOLAB

To address the lack of reproducible implementations, the complexity of evaluating performance indicators, and the absence of a unified software platform for experimentation in evolutionary dynamic optimization, we have developed an open-source MATLAB platform known as the *Evolutionary Dynamic Optimization LAB*oratory (EDOLAB). The current version of EDOLAB is primarily focused on single-objective, unconstrained, continuous DOPs. However, the

EDOAs designed for this class of DOPs have been demonstrated to be easily extendable to address other significant classes of DOPs, including robust optimization over time (ROOT) [Yazdani et al. 2024b, 2022], constrained DOPs [Bu et al. 2016; Nguyen and Yao 2012], and large-scale DOPs [Luo et al. 2017; Yazdani et al. 2019].

Moreover, although the structures of EDOAs designed for single-objective DOPs differ from those tailored for finding Pareto optimal solutions (POS) in each environment within multi-objective DOPs [Jiang et al. 2022], they remain effective for addressing many multi-objective DOPs. In fact, in many multi-objective DOPs, a *single* solution is deployed in each environment, chosen by a decision maker based on both user preferences and problem-specific characteristics. Therefore, finding the POS for each environment and selecting a solution for deployment may not always be the most effective approach, particularly in problems with high-frequency environmental changes. For example, given a real-world multi-objective DOP where the environment changes every few seconds, it can be challenging for a user to select a solution from the POS for each environment. To address such multi-objective DOPs, the problem can be transformed into a single-objective DOP by combining all objectives according to the decision maker's preferences. These preferences are both user-driven, based on the decision maker's goals and values, and problem-dependent, considering the specific nature and constraints of the problem at hand [Kaddani et al. 2017; Marler and Arora 2010]. Consequently, in the resulting single-objective problem, a single-objective EDOA may be more suitable, focusing on finding an optimal solution for deployment in each environment based on these combined preferences.

### 1.3 Design Goals and Key Features

In the following, we describe the major contributions and features of EDOLAB.

**1.3.1 Comprehensive library.** The current release of EDOLAB includes 27 EDOAs, as listed in Table 1, each with distinct characteristics such as varying structures, optimizers, population clustering and sub-population management methods, diversity control components, and computational resource allocation strategies. EDOLAB also includes four dynamic benchmark generators: MPB [Branke 1999], free peaks (FPs) [Li et al. 2018], Generalized Dynamic Benchmark Generator (GDBG) [Li et al. 2008], and Generalized MPB (GMPB) [Yazdani et al. 2021a, 2020b]. These benchmark generators are fully parametric and capable of producing dynamic problem instances with varying morphological and dynamical characteristics. To evaluate the efficiency of EDOAs in solving DOPs and facilitate comparisons, EDOLAB incorporates the two most commonly used performance indicators: offline error [Branke and Schmeck 2003] and the average error before environmental changes [Trojanowski and Michalewicz 1999].

**1.3.2 Easy to use.** EDOLAB is developed in MATLAB, a programming language that offers a vast collection of high-level mathematical functions for operating on arrays and matrices, along with various random number generators. These features make MATLAB an ideal choice for implementing EDOAs and dynamic benchmarks. The source codes of EDOLAB, particularly for the EDOAs and benchmarks, are designed to be easy to understand, trace, and modify, thanks to MATLAB's capabilities.

Based on our investigations, MATLAB has been one of the most frequently used programming languages in the field of evolutionary dynamic optimization due to its strengths and overall ease of use, including its straightforward syntax and readability. Moreover, MATLAB is highly popular not only in this field but also in other active areas such as evolutionary multi-objective optimization. This popularity is reflected in the widespread use of platforms like PlatEMO [Tian et al. 2017], which is implemented in MATLAB.

We have also structured and modularized the platform to ensure the clarity and readability of the source code. The descriptive name of the parameters, the clear distinctions between components, and the detailed comments make

the source code easy to trace and modify. EDOLAB features a graphical user interface (GUI) designed to enhance user-friendliness.

*1.3.3 Advanced Experimentation Interface.* EDOLAB includes a graphical interface specifically designed to support research-grade experimentation workflows. It allows users to fully configure all algorithmic and benchmark parameters through an intuitive GUI. In addition, the platform supports parallel execution of multiple experiments. Experiments can be saved and restored using `.m` files to reduce setup overhead for repeated runs. Furthermore, all results can be exported directly to Excel files for further analysis or reporting.

EDOLAB integrates built-in functionalities for statistical analysis of experimental results. Users can compare multiple algorithms using non-parametric tests such as the Friedman test and Wilcoxon signed-rank test, with automated generation of groupings and rankings. It also supports sensitivity analysis, enabling researchers to examine the impact of individual parameter settings on algorithm performance.

In addition, EDOLAB provides rich visualization features, including box plots for comparing distributions of performance metrics and trend plots for analyzing algorithm behavior over time or across environments. These tools enhance the interpretability of experimental results and support deeper insights into algorithm behavior and parameter interactions.

*1.3.4 Flexible and Comprehensive Research Capabilities.* EDOLAB offers researchers significant flexibility in conducting empirical studies by supporting both a “GUI Mode” for interactive use and a “Code Mode” for script-based experimentation and advanced customization. Its extensive library allows for the thorough investigation and comparison of various EDOAs, each designed with different structures, optimizers, and components to address dynamic optimization challenges across a wide range of problem instances. Additionally, EDOLAB is particularly valuable for researchers developing new EDOAs or improving existing algorithms. Since the platform’s source code is open-source and modularly designed, it becomes easy to incorporate new mechanisms—such as population control, diversity control, or other innovative components—into the existing algorithms. The platform includes four dynamic benchmark generators, capable of producing a broad range of problem instances with varying levels of difficulty and characteristics. This, coupled with a collection of comparison EDOAs, performance indicators, and visualization plots, makes EDOLAB an invaluable tool for evaluating algorithms and generating results for scientific reports and articles.

*1.3.5 Educational Support.* EDOLAB includes an educational application, which is particularly useful for new researchers, such as PhD students, to enter the field and contribute more efficiently. This application visualizes the 2-dimensional problem space (i.e., environment) and dynamically illustrates how the morphology of the search space evolves after each environmental change. Users can observe how individuals relocate over time, providing valuable insights into how EDOAs adapt to environmental changes. By highlighting the similarity factors between successive environments, this application enables users to grasp the significance of knowledge transfer from the previous to the current environment, accelerating their understanding of complex dynamic optimization behaviors.

*1.3.6 Extensibility.* EDOLAB is designed for easy extensibility, allowing researchers to expand its library by adding new EDOAs, benchmark problems, and performance indicators. With EDOLAB’s open-source code, researchers can also modify EDOA frameworks, incorporate their own components, and explore their effectiveness. For instance, if a researcher develops a new exclusion, change reaction, or convergence detection component, they can simply replace the existing code with the new one and assess its impact on the overall performance of an EDOA. Additionally, researchers can extend EDOAs to address other classes of DOPs by incorporating specific components necessary for those problems,

such as constraint-handling mechanisms for constrained DOPs [Nguyen and Yao 2012] or decision-making components for altering or maintaining deployed solutions in ROOT [Yazdani et al. 2018a, 2017].

**1.3.7 Innovative modular design.** One of the key contributions of EDOLAB is its innovative design, which unifies a diverse set of EDOAs through a modular structure. This design allows different algorithms, each with varying mechanisms and structures, to be integrated into a single, cohesive platform. By utilizing a novel approach to component modularization, EDOLAB enables these algorithms to share a common framework while maintaining their unique characteristics. In addition, the modular design supports the extensibility of the platform, allowing new algorithms, benchmarks, and performance indicators to be incorporated with minimal effort.

**1.3.8 Open-source availability and accessibility.** A significant contribution of EDOLAB is its open-source availability. The platform is publicly accessible through GitHub, allowing researchers to utilize and extend its functionality for experimental and comparative studies. By making EDOLAB open-source, we aim to provide a valuable resource for researchers working on evolutionary dynamic optimization. The platform can be accessed from [<https://github.com/EvoMindLab/EDOLAB>].

## 1.4 Paper Organization

The remainder of this paper is organized as follows: Section 2 introduces the class of dynamic optimization problems considered in the current version of EDOLAB and describes the four benchmark generators included in the platform. Section 3 presents the performance indicators used to evaluate EDOAs in EDOLAB, including both quantitative error-based metrics and their corresponding visualizations. Section 4 provides an overview of common EDOA frameworks and components, lists the 27 algorithms included in EDOLAB, and outlines implementation decisions and standardization policies adopted for fair experimentation. Section 5 offers an overview of the platform’s structure and technical architecture. Further details are provided in the user manual (available as a separate document), which includes comprehensive information on the software design, source code organization, GUI, usage instructions, and extensibility options. Section 6 reports benchmark results across various scenarios to illustrate the capabilities of EDOLAB and offers usage guidance to users. Finally, Section 7 concludes the paper with a summary and discussion of future directions.

## 2 Dynamic Optimization Problems and Benchmark Generators

In this section, we first provide a formal definition of the class of DOPs targeted in the current version of EDOLAB, focusing on single-objective, unconstrained, continuous problems. All algorithms, benchmark generators, and performance indicators in EDOLAB are designed for maximization problems within this class.

We then describe the four dynamic benchmark generators included in EDOLAB—MPB, GDBG, FPs, and GMPB—each of which can produce diverse problem instances with varying morphological and dynamical characteristics. These generators form the foundation for evaluating and comparing EDOAs under controlled and configurable scenarios.

### 2.1 General Definition of Dynamic Optimization Problems

A single-objective, unconstrained, continuous DOP can be defined as:

$$\text{Maximize : } f^{(t)}(\mathbf{x}) = f(\mathbf{x}, \alpha^{(t)}), \mathbf{x} = \{x_1, x_2, \dots, x_d\}, \quad (1)$$

where  $\mathbf{x}$  is a solution in the  $d$ -dimensional search space,  $f$  is the time-varying objective function,  $t$  is the time index, and  $\alpha$  is a set of time-varying environmental parameters. Most research in this field focuses on DOPs where environmental

changes occur only at discrete time steps, which is characteristic of many real-world DOPs [Nguyen 2011]. For a problem with  $T$  environments, there is a sequence of  $T$  stationary search spaces. Consequently, a DOP, with  $T$  environmental states (i.e.,  $T - 1$  environmental changes), can be reformulated as:

$$\text{Maximize : } f(\mathbf{x}) = \{f(\mathbf{x}, \alpha^{(t)})\}_{t=1}^T = \{f(\mathbf{x}, \alpha^{(1)}), f(\mathbf{x}, \alpha^{(2)}), \dots, f(\mathbf{x}, \alpha^{(T)})\}. \quad (2)$$

It is generally assumed that there is a degree of morphological similarity between successive environments, a characteristic commonly observed in many real-world DOPs [Branke 2012; Nguyen 2011; Yazdani 2018].

## 2.2 Dynamic Benchmark Generators

A wide variety of benchmark generators have been proposed for evaluating EDOAs. In EDOLAB, we have selected four of the most well-established and advanced benchmark generators in the field: MPB, GDBG, FPs, and GMPB. These benchmarks collectively span a range of landscape complexities, from simple and interpretable test functions suitable for educational use to highly configurable and challenging problem instances for advanced experimentation. A comprehensive review of other benchmark generators can be found in [Yazdani et al. 2021c], and readers interested in a broader survey of benchmarks are encouraged to refer to that work.

MPB [Branke 1999] is the most widely used dynamic benchmark generator in the field [Yazdani et al. 2020b]. The landscapes generated by the standard version of MPB are relatively straightforward to optimize, as they consist of a series of conical promising regions (i.e., peaks) that are regular, unimodal, symmetric, fully separable [Yazdani et al. 2019], and well-conditioned. Despite its simplicity, MPB is an essential component of EDOLAB due to its significant value for educational purposes. MPB's straightforward nature makes it easier for users to observe the behavior of EDOAs over time and investigate the effectiveness of various components, such as promising region coverage, change reaction, and diversity control. In the standard MPB, the height, width, and center of each promising region (represented as a cone) change over time. To enhance the realism of MPB in EDOLAB, we have made a few modifications. First, we removed the option that allowed all promising regions to be initialized with identical height and width. In EDOLAB, the attributes of all promising regions are initialized randomly within predefined ranges. Second, we eliminated the option for correlated movements of promising regions, which could result in linear relocation directions of the promising region centers. Instead, in EDOLAB, the directions of shifts are randomized, providing a stochastic and more challenging dynamic environment.

GDBG [Li et al. 2008] is the second most commonly used dynamic benchmark in the field. GDBG is created by introducing dynamics to composition benchmark functions [Liang et al. 2005; Suganthan et al. 2005], which are commonly employed in static global optimization. The problem instances generated by GDBG are generally more complex and challenging than those produced by MPB, as they involve landscapes with irregular, multimodal, and partially separable components. Despite its lack of fully controllable characteristics, GDBG has been widely used in research and can still provide valuable insights into the performance of EDOAs. To create a more comprehensive platform, we have included GDBG in EDOLAB, allowing researchers to leverage its complexity for a broader range of experimental evaluations.

FPs [Li et al. 2018] is the third benchmark generator included in EDOLAB. The landscapes generated by FPs are divided into several hypercubes using a k-d tree [Bentley and Friedman 1979], with each hypercube containing one promising region. As a result, the basin of attraction for each promising region is determined by the hypercube in which it lies. After each environmental change, the shape of each promising region is randomly selected from eight different unimodal functions. Additionally, the location and size of each hypercube change over time, which alter

the basin of attraction of the promising region. The center position of each promising region also shifts within the hypercube. Several transformations, such as symmetry breaking and condition number increasing, are applied in FPs, though these transformations remain fixed over time. FPs is also suitable for educational purposes, as its promising regions are clearly defined by the hypercubes.

GMPB [Yazdani et al. 2021a, 2020b] is the final benchmark generator included in EDOLAB. GMPB is a complex, fully configurable benchmark generator. The landscapes generated by GMPB are constructed by assembling several promising regions with a variety of controllable characteristics, ranging from unimodal to highly multimodal, symmetric to highly asymmetric, smooth to highly irregular, and varying degrees of variable interaction and ill-conditioning. All of these characteristics can change over time. With its high degree of configurability, GMPB allows users to examine the performance of proposed or existing EDOAs across a wide range of problem instances with different characteristics and difficulty levels. Consequently, GMPB is well-suited for experimentation and for investigating and comparing the performance of different EDOAs. However, due to the complexity of the search spaces generated by GMPB, it is not the ideal choice for educational purposes.

Figure 1 provides examples of the landscapes generated by the four benchmarks included in EDOLAB: MPB, GDBG, FPs, and GMPB. These contour plots illustrate the morphological characteristics of each benchmark’s problem space. However, it is important to note that these are just examples, and the characteristics shown in these figures cannot be generalized to all possible landscapes that can be generated by these benchmarks.

The benchmarks included in EDOLAB provide a flexible and robust framework for evaluating the performance of EDOAs. Their parametric nature allows researchers to generate a wide variety of problem instances with different morphological and dynamical characteristics, making them invaluable for algorithm evaluation and educational purposes. These benchmarks help researchers systematically understand how algorithms behave under controlled scenarios, revealing their strengths and weaknesses, which is crucial for guiding future improvements. Additionally, they serve as a foundation for new researchers to study and experiment with algorithmic concepts, making them highly suitable for educational use. In the user manual, we have provided commonly used parameter settings for generating problem instances from the four benchmark generators included in EDOLAB. These settings generate 12 instances per benchmark generator, covering a range of difficulties and characteristics that are commonly used in the literature.

At this stage, we do not include real-world problems in the platform for two primary reasons. First, many real-world problems are inflexible and do not allow the generation of diverse problem instances with varying characteristics, limiting their usefulness for the comprehensive evaluation of algorithms. Second, for many real-world problems, we do not fully understand their morphological characteristics, dynamic behavior, or the specific challenges they present. This lack of transparency makes them less suitable for testing and refining algorithms.

While there are real-world problem domains, such as dynamic facility location problems, that offer the flexibility to generate instances with controllable characteristics, the algorithms currently available in the field—including those implemented in the current version of EDOLAB—are not yet capable of addressing the complex challenges posed by such problems [Yazdani et al. 2024a]. It is not merely a matter of suboptimal performance; these algorithms often fail to function effectively in such scenarios, highlighting the need for further advancements before tackling real-world dynamic problems of this nature. Consequently, these problems—and benchmark generators that attempt to simulate their behavior—are not included in the current version of EDOLAB.



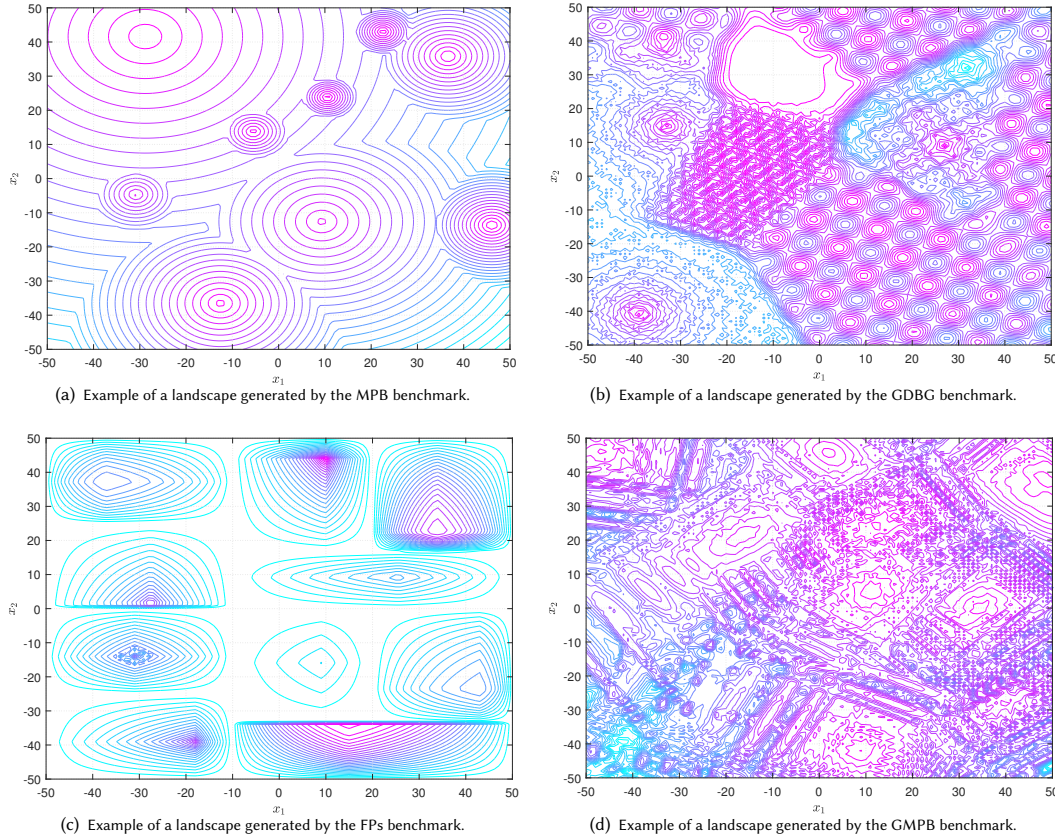


Fig. 1. Examples of two-dimensional landscapes generated by the four benchmarks: (a) MPB, (b) GDBG, (c) FPs, and (d) GMPB. These are representative examples, and the characteristics shown cannot be generalized to all possible landscapes generated by these benchmarks.

### 3 Performance Indicators for Dynamic Optimization

A wide range of performance indicators have been proposed to evaluate the behavior and efficiency of EDOAs in dynamic environments. The current version of EDOLAB incorporates two of the most widely used and well-established error-based indicators—offline error [Branke and Schmeck 2003] and average error before environmental changes [Trojanowski and Michalewicz 1999]—as they provide comprehensive insights into an algorithm’s ability to track moving global optimum and recover from changes. These indicators are particularly suited to synthetic benchmarks with known optimum, which are used in all evaluations within EDOLAB.

These two performance indicators are the most widely used in the field, with over 85% of studies relying on them [Yazdani et al. 2021c]. EDOLAB does not include the offline performance indicator [Branke 1999], a third commonly used metric, as it provides no additional insights beyond those captured by the offline error. We also exclude distance-to-optimum-based indicators [Duhain 2012], which measure proximity to the global optimum without considering solution quality. Since nearly all EDOAs are designed to optimize fitness values rather than Euclidean distance, such indicators may lead to misleading assessments of algorithm performance [Yazdani et al. 2021c].

To complement these numerical indicators, EDOLAB also provides a set of visualization tools that support in-depth temporal and statistical analysis of algorithm behavior. These include current error plots, offline error over time plots, moving averages of end-of-environment errors, and box plots summarizing indicator distributions across runs. Together, these tools enable both quantitative and visual comparisons of algorithms under dynamic conditions.

### 3.1 Quantitative Error-Based Indicators

**3.1.1 Offline error.** Offline error measures the mean error of the best-found solution across all objective function evaluations. It is calculated using the following equation [Branke and Schmeck 2003]:

$$E_o = \frac{1}{\sum_{t=1}^T v^{(t)}} \sum_{t=1}^T \sum_{\vartheta=1}^{v^{(t)}} \left( f^{(t)}(g^{\star(t)}) - f^{(t)}\left(g^{\star\left(\vartheta + \sum_{k=1}^{t-1} v^{(k)}\right)}\right) \right), \quad (3)$$

where  $E_o$  represents the offline error,  $g^{\star(t)}$  is the global optimum at the  $t$ -th environment,  $v^{(t)}$  is the number of objective function evaluations in the  $t$ -th environment,  $T$  is the number of environments,  $\vartheta$  is the function evaluation counter for each environment, and  $g^{\star\left(\vartheta + \sum_{k=1}^{t-1} v^{(k)}\right)}$  is the best-found solution at the  $\vartheta$ -th function evaluation in the  $t$ -th environment. The offline error measures the overall performance of an algorithm across all function evaluations and captures the convergence speed of an EDOA following environmental changes. This metric serves as an effective indicator for assessing how quickly (based on the number of function evaluations) an algorithm adapts to changes in the environment and converges towards the optimal solution.

**3.1.2 Average error before environmental changes.** The second performance indicator, average error before environmental changes ( $E_{bbc}$ ) [Trojanowski and Michalewicz 1999], focuses on the error of the best-found solution just before each environmental change. It is calculated as follows:

$$E_{bbc} = \frac{1}{T} \sum_{t=1}^T \left( f^{(t)}(g^{\star(t)}) - f^{(t)}(g^{\text{end}(t)}) \right), \quad (4)$$

where  $g^{\text{end}(t)}$  is the best found solution at the end of the  $t$ -th environment. Since  $E_{bbc}$  focuses solely on the best-found solution at the end of each environment, it does not capture the convergence speed or performance across all function evaluations within an environment.

### 3.2 Visualization of Performance Indicators

In addition to the numerical performance indicators, EDOLAB provides multiple visualization tools to support the temporal analysis and comparison of algorithm performance across environments and function evaluations. These include convergence trend plots, temporal error averages, and statistical box plots.

- **Current error plots** display the error of the best-found solution at each function evaluation, providing a detailed view of the convergence behavior within each environment. These plots help assess how quickly an algorithm responds to environmental changes and how effectively it converges within each stationary period.
- **Offline error over time plots** show the moving average of the error of the best-found solutions across all previous function evaluations. For each evaluation point, the plot reflects the algorithm's cumulative ability to track the moving optimum over time.
- **Average error before change over time plots** provide a complementary perspective by showing how the average error of the best found solution at the end of each environment evolves over successive changes. For

the  $t$ -th environment, this value is computed as the mean of all previous end-of-environment errors up to environment  $t$ .

- **Box plots of performance indicators** are provided to summarize the distribution of each performance metric across all runs. These plots support direct visual comparison of algorithms or configurations based on central tendency, spread, and outliers.

## 4 Evolutionary Dynamic Optimization Algorithms (EDOAs)

EDOAs are designed to address the unique challenges posed by DOPs, where the problem changes over time. These algorithms extend static evolutionary frameworks by incorporating mechanisms for adaptation, diversity maintenance, memory, and resource allocation, enabling them to track moving global optimum across changing environments.

This section provides an overview of the fundamental components, structural frameworks, and algorithmic strategies commonly used in the design of EDOAs. We first outline the components that constitute most EDOA architectures, followed by a taxonomy of popular algorithmic frameworks—ranging from single-population to advanced multi-population approaches. Readers interested in a more comprehensive survey of EDOA structures and classifications are referred to [Mavrovouniotis et al. 2017; Nguyen et al. 2012; Yazdani et al. 2021b, 2024c].

### 4.1 Common Frameworks and Component Design

EDOAs are typically composed of multiple components that work in coordination to improve performance across dynamic environments. These components may include population management strategies, memory-based mechanisms, diversity preservation techniques, convergence detectors, and optimization operators. Each plays a distinct role in enabling the algorithm to react to changes and maintain effective search behavior throughout the optimization process.

#### 4.1.1 Key Components of EDOAs.

- **Population Management:** These components are responsible for controlling the creation, organization, and activities of subpopulations in algorithms that utilize more than one population, such as bi-population and, more commonly, multi-population methods [Li et al. 2015]. They dictate how subpopulations are formed, how they share information with each other, and how computational resources are allocated across the subpopulations to ensure an effective balance between exploration and exploitation.
- **Explicit Memory:** EDOAs often rely on historical knowledge to enhance their optimization process over time and after environmental changes. One method for storing and retrieving this information is through explicit memory [Branke 1999], which keeps track of past optima. In some algorithms, explicit memory is used to accelerate the change reaction, while in others, it is employed to manage subpopulations and prevent over-exploitation of promising regions.
- **Diversity Control:** In evolutionary algorithms, *diversity loss* is a significant challenge when optimizing in dynamic environments [Branke 2012]. Diversity control components help address this challenge. There are two main types of diversity control components. The first type, *local diversity control*, addresses local diversity loss by facilitating the tracking of local optima and enhancing exploitation after environmental changes. The second type, *global diversity control*, aims to maintain the capability of exploration. Diversity control components are further classified into those that maintain diversity throughout the optimization process and those that inject diversity into subpopulations or the overall population when certain conditions trigger them.

- **Convergence Detection:** While not directly addressing any specific challenge of optimization in dynamic environments, convergence detection is essential for triggering other components, such as those related to diversity control or population management. It helps identify when a population or subpopulation has converged on a promising region.
- **Optimizer:** One of the core components of an EDOA is the optimizer itself, which is often an algorithm originally designed for static optimization. However, when integrated with other components such as population management and diversity control, these optimizers become capable of handling dynamic environments by adapting to environmental changes and maintaining effective search performance.

A significant line of research in the field involves the development of new versions or improvements of these core components. In EDOLAB, we have designed the platform with modularity in mind, ensuring that each of the components is clearly separated and well-structured within the codebase. This modularization allows researchers to easily locate, modify, or replace individual components—such as the population management or diversity control mechanisms—without requiring a full re-implementation of the algorithm. This design facilitates experimentation and enables users to evaluate the performance of existing algorithms when paired with newly developed components.

#### 4.1.2 Taxonomy of EDOA Frameworks.

Based on the components and strategies used in the framework of EDOAs, we can categorize them into the following classes [Yazdani et al. 2024c]:

- **Single-Population Algorithms:** These algorithms employ a single population throughout the optimization process [Eberhart and Shi 2001b]. Their primary limitation is the lack of flexibility in exploring different regions of the search space, making them less effective for complex dynamic problems.
- **Bi-Population Algorithms:** Bi-population algorithms divide the population into two groups, where one typically focuses on exploration and the other on exploitation [Branke 1999]. This basic division offers improved performance compared to single-population algorithms by balancing the exploration of new regions and the exploitation of the best known promising region.
- **Multi-Population Algorithms:** The most advanced and commonly used class, multi-population EDOAs employ several subpopulations to explore various regions of the search space [Blackwell and Branke 2004]. These algorithms tend to offer the best performance for solving dynamic optimization problems because they can simultaneously explore and exploit multiple regions. Multi-population algorithms are highly flexible and can incorporate a variety of other components such as diversity control and convergence detection. In terms of population management components, these algorithms can be further classified based on the following aspects:
  - **Fixed vs. Adaptive Subpopulations:** In multi-population methods, subpopulations can either have a fixed or adaptive structure. Algorithms with a fixed number of subpopulations maintain the same structure throughout the optimization process [Blackwell and Branke 2006], while those with adaptive subpopulations dynamically adjust the number of subpopulations based on the number of promising regions discovered [Blackwell 2007]. Adaptive algorithms rely on mechanisms for creating and eliminating subpopulations in response to environmental changes [Yazdani et al. 2020a].
  - **Clustering-Based Subpopulation Formation:** Subpopulations in multi-population EDOAs can be generated through clustering methods. These clustering strategies may use the positions and fitness values of individuals [Yang and Li 2010] or simply group individuals based on indices [Blackwell and

[Branke 2006](#)]. Clustering based on positions can be more effective, as it takes the spatial distribution of individuals into account when forming subpopulations [[Yazdani et al. 2021b](#)]. However, this approach is more computationally expensive as it requires frequent re-clustering of individuals.

- **Homogeneous vs. Heterogeneous Subpopulations:** Subpopulations in multi-population methods can be either homogeneous or heterogeneous. In homogeneous subpopulations, each subpopulation uses the same optimizer, structure, and parameter settings [[Mendes and Mohais 2005](#)]. On the other hand, heterogeneous subpopulations [[Li and Yang 2008](#)] have varying structures, parameter settings, or even optimizers, making them more flexible and potentially more effective at assigning different roles and covering diverse regions of the search space.

## 4.2 EDOAs Included in EDOLAB

The current release of EDOLAB includes 27 EDOAs, listed in Table 1. As seen in the table, Particle Swarm Optimization (PSO) [[Bonyadi and Michalewicz 2017](#)] and Differential Evolution (DE) [[Das and Suganthan 2010](#)] are the most commonly used optimizers in the algorithms included in EDOLAB. This choice is in line with the distribution of optimizers in the literature, where these two are the most frequently employed in the field of evolutionary dynamic optimization [[Yazdani et al. 2021c](#)]. Note that while Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [[Hansen and Ostermeier 2001](#)] has proven to be a powerful and widely used optimizer in other fields of optimization, its application to continuous single-objective, unconstrained DOPs, which is the focus of the current version of EDOLAB, has been limited. Consequently, CMA-ES is only represented by a single algorithm in EDOLAB, which mirrors its relatively low adoption in the field of evolutionary dynamic optimization.

Additionally, multi-population algorithms are heavily represented in EDOLAB, as they are widely recognized as the most effective class of algorithms for DOPs [[Li et al. 2015](#); [Yazdani et al. 2021b, 2024c](#)]. As discussed in Section 4.1, different versions of multi-population EDOAs have been proposed in the literature, each employing a variety of techniques to manage subpopulations. In EDOLAB, we have included algorithms with:

- Fixed and adaptive numbers of subpopulations,
- Fixed and adaptive population sizes,
- Various clustering methods for forming subpopulations, including clustering based on positions and fitness,
- Homogeneous and heterogeneous subpopulation structures, where subpopulations may either share similar characteristics or differ in terms of parameter settings, size, or optimization components.

These variations provide a broad spectrum of strategies for addressing the complexities of DOPs. Overall, the distribution of EDOAs in EDOLAB is consistent with the trends observed in the literature, ensuring that the platform accurately represents the current state of the field.

## 4.3 Implementation Notes and Standardization Policies

To ensure fair comparisons in experiments, we have standardized certain aspects of the optimization components used in the EDOAs, meaning that some EDOAs in EDOLAB may differ slightly from their original versions. For all EDOAs that utilize PSO as their optimization component, we employ PSO with a constriction factor [[Eberhart and Shi 2001a](#)]. Additionally, DE/rand/2/bin [[Mendes and Mohais 2005](#)] is used for most EDOAs that incorporate DE. In CESO [[Lung and Dumitrescu 2007](#)], crowding DE (DE/rand/1/exp) [[Thomsen 2004](#)] is used to maintain global

Table 1. EDOAs included in the initial release of EDOLAB.

EDO	Reference	Optimization component	Population structure	Number of sub-populations	Population size	Population clustering	Sub-population heterogeneity
ACF <sub>PSO</sub>	[Yazdani et al. 2020a]	PSO	Multi-population	Adaptive	Adaptive	By index	Homogeneous
AMP <sub>DE</sub>	[Li et al. 2016]	DE	Multi-population	Adaptive	Adaptive	By position	Heterogeneous
AMP <sub>PSO</sub>	[Li et al. 2016]	PSO	Multi-population	Adaptive	Adaptive	By position	Heterogeneous
AmQSO	[Blackwell et al. 2008]	PSO	Multi-population	Adaptive	Adaptive	By index	Homogeneous
AMSO	[Li et al. 2014]	PSO	Multi-population	Adaptive	Adaptive	By position	Heterogeneous
CDE	[Du Plessis and Engelbrecht 2012]	DE/best/2/bin	Multi-population	Fixed	Fixed	By index	Homogeneous
CESO	[Lung and Dumitrescu 2007]	DE/rand/1/exp and PSO	Bi-population	Fixed	Fixed	N/A	Heterogeneous
CPSO	[Yang and Li 2010]	PSO	Multi-population	Adaptive	Fixed	By position	Heterogeneous
CPSOR	[Li and Yang 2012]	PSO	Multi-population	Adaptive	Fixed	By position	Heterogeneous
DSPSO	[Parrott and Li 2006]	PSO	Multi-population	Adaptive	Fixed	By position and fitness	Heterogeneous
DynDE	[Mendes and Mohais 2005]	DE/best/2/bin	Multi-population	Fixed	Fixed	By index	Homogeneous
DynPopDE	[du Plessis and Engelbrecht 2013]	DE/best/2/bin	Multi-population	Adaptive	Adaptive	By index	Homogeneous
FTMPSO	[Yazdani et al. 2013]	PSO	Multi-population	Adaptive	Adaptive	By index	Heterogeneous
HmSO	[Kamosi et al. 2010]	PSO	Multi-population	Fixed	Fixed	By index	Homogeneous
IDSPSO	[Blackwell et al. 2008]	PSO	Multi-population	Adaptive	Fixed	By position and fitness	Heterogeneous
ImQSO	[Kordestani et al. 2019]	PSO	Multi-population	Fixed	Fixed	By index	Homogeneous
mCMA-ES	[Yazdani et al. 2019]	CMA-ES	Multi-population	Adaptive	Adaptive	By index	Homogeneous
mDE	[Yazdani et al. 2019]	DE/best/2/bin	Multi-population	Adaptive	Adaptive	By index	Homogeneous
mjDE	[Yazdani et al. 2019]	jDE	Multi-population	Adaptive	Adaptive	By index	Homogeneous
mPSO	[Yazdani et al. 2019]	PSO	Multi-population	Adaptive	Adaptive	By index	Homogeneous
mQSO	[Blackwell and Branke 2006]	PSO	Multi-population	Fixed	Fixed	By index	Homogeneous
psfNBC	[Luo et al. 2018]	PSO	Multi-population	Adaptive	Fixed	By position and fitness	Homogeneous
RPSO	[Hu and Eberhart 2002]	PSO	Single-population	Fixed	Fixed	N/A	N/A
SPSO <sub>AD+AP</sub>	[Yazdani et al. 2023]	PSO	Multi-population	Adaptive	Adaptive	By position and fitness	Homogeneous
TMPSO	[Wang et al. 2007]	PSO	Bi-population	Fixed	Fixed	N/A	Heterogeneous
APCPSO	[Liu et al. 2020]	PSO	Multi-population	Adaptive	Adaptive	By position	Heterogeneous
DPCPSO	[Li et al. 2024]	PSO	Multi-population	Adaptive	Adaptive	By position	Heterogeneous

diversity; thus, we have retained this DE version. Similarly, in mjDE, the jDE [Brest et al. 2006], a well-known self-adaptive version of DE, is employed. Furthermore, to handle the box constraints [Mezura-Montes and Coello 2011] (i.e., keeping the individuals/candidate solutions within search space boundaries), we apply the absorb boundary handling technique [Helwig et al. 2012] across all EDOAs.



We also assume that all EDOAs in EDOLAB are informed about environmental changes; therefore, change detection components are not included. As described in [Branke and Schmeck 2003], environmental changes in many real-world DOPs are often *visible*, with optimization algorithms being informed of these changes through other system components, such as agents, sensors, or the arrival of new entities (for example, new orders) [Yazdani et al. 2021b]. In such scenarios, the algorithms do not require a change detection component.

In the original versions of some EDOAs, internal parameters of benchmark generators, such as shift severity, were utilized by the algorithms. However, for fair and unbiased comparisons, problem instances must be treated as black boxes, and using such internal knowledge can disadvantage other EDOAs that do not have access to it. In EDOLAB, we employ the shift severity estimation method from [Yazdani et al. 2018b] for all EDOAs that require knowledge about shift severity. Furthermore, in EDOAs and components that originally required knowledge of the number of promising regions, we use the number of sub-populations instead [Blackwell et al. 2008].

## 5 Overview of Structure and Architecture of EDOLAB

This section provides a high-level overview of EDOLAB’s structure, focusing on its core components and their roles in experimentation and education. In this section, it is intended to give readers a conceptual understanding of the platform’s capabilities without exploring implementation details. Complete technical information is available in a comprehensive user manual that accompanies the platform. The manual offers detailed guidance on EDOLAB’s internal architecture, implementation structure, usage workflows, extension procedures, and Octave compatibility. It is provided as supplementary material to this article and is also accessible via the project’s GitHub repository: [<https://github.com/EvoMindLab/EDOLAB>].

EDOLAB supports two modes of operation: a *GUI Mode* and a *Code Mode* (script-based). The GUI Mode offers a comprehensive environment for configuring experiments, modifying algorithm and benchmark parameters, running multiple experiments in parallel, and performing statistical analysis with integrated visualization tools. The Code Mode remains available for those who prefer direct script-level control or need to customize internal components at a lower level. A detailed comparison of these two modes, along with usage instructions and workflow differences, is provided in Section M-II of the user manual.

EDOLAB provides two primary applications: *Experimentation* and *Education*, each targeting different user needs. The Experimentation application focuses on algorithm evaluation, benchmarking, and statistical analysis, while the Education application emphasizes interpretability and visual understanding of EDOA behavior in dynamic environments. Details about the functionality of these applications and how users interact with them are provided in Sections M-II.1.1 and M-II.1.2 of the user manual. Visual illustrations of their graphical interfaces are shown in Figures M-2 and M-8.

In the current version of EDOLAB, the *Experimentation* application enables users to run and compare 27 pre-implemented EDOAs across four configurable benchmark generators. Experiment parameters can be customized either through the GUI or by modifying the source code. When configured via the GUI, parameter changes are applied at runtime without altering the default values hardcoded in the source files. The Code Mode offers full flexibility, allowing users to directly modify algorithm components and internal logic. In contrast, the GUI Mode provides additional capabilities, including batch experiment configuration, parallel execution using thread or process pools, automated result aggregation, and an integrated statistical analysis panel that supports significance testing and visual comparisons (see Manual Section M-II.1.1).

The *Education* application is designed for visual exploration of EDOA behavior in 2D problem instances. It generates contour plots that dynamically display the movement of individuals and environmental changes. The application

operates identically in both GUI and Code Modes, providing equivalent functionality for configuring benchmark and algorithm parameters and for producing detailed visualizations. Step-by-step usage instructions for the Education application in both modes are provided in Sections [M-II.1.2](#) (GUI Mode) and [M-II.2](#) (Code Mode) of the user manual.

EDOLAB is designed with modularity as a core principle. Algorithms, benchmark generators, and performance indicators are encapsulated in separate folders and implemented using standardized interfaces and naming conventions. This structure ensures that modules can be added or modified independently, without affecting other parts of the platform. The user manual provides detailed guidance on how to extend EDOLAB with new modules, including file-level descriptions and a sequence diagram showing how components interact (see Sections [M-I](#) and [M-III](#) of the user manual).

To support users without access to MATLAB, EDOLAB also offers compatibility with GNU Octave, an open-source alternative. This compatibility is available for the Code Mode (non-GUI), as GUI functionalities depend on MATLAB-specific components that are not fully supported in Octave. A dedicated compatibility layer is included in the platform, and key scripts have been adapted to ensure proper functionality. Instructions for configuring and running EDOLAB in Octave are provided in Section [M-IV](#) of the user manual.

## 6 Benchmarking Results and Visualization Overview

To showcase EDOLAB’s capabilities in benchmarking and experimental analysis, we present a study evaluating nine recent EDOAs included in the current version of the EDOLAB library. These algorithms were compared across several scenarios generated by four benchmark generators described in Section [2.2](#). This analysis not only highlights algorithmic behavior but also demonstrates how EDOLAB supports comprehensive result interpretation through its statistical and visualization tools. In addition to the comparative benchmarking study, we include a standalone sensitivity analysis on a key parameter of one representative EDOA. This demonstrates how EDOLAB facilitates deeper investigations into algorithm behavior at the parameter level. The benchmark scenarios used in this study vary in the number and morphology of promising regions, enabling a broad evaluation of algorithm performance under diverse conditions. The results are intended to showcase EDOLAB’s features for analyzing benchmarking outcomes.

### 6.1 Algorithm Performance Comparison Across Benchmark Scenarios

To evaluate the behavior of recent EDOAs across diverse problem landscapes, we conducted a series of experiments involving the nine latest algorithms integrated into EDOLAB: SPSO<sub>AP+AD</sub> [[Yazdani et al. 2023](#)], ACF<sub>PSO</sub> [[Yazdani et al. 2020a](#)], APCPSO [[Liu et al. 2020](#)], DPCPSO [[Li et al. 2024](#)], ImQSO [[Kordestani et al. 2019](#)], mDE [[Yazdani et al. 2019](#)], mPSO [[Yazdani et al. 2019](#)], AMP<sub>PSO</sub> [[Li et al. 2016](#)], and psfNBC [[Luo et al. 2018](#)]. Each algorithm was tested across benchmark scenarios generated by all four generators described in Section [2.2](#), with the number of promising regions set to 10, 25, 50, and 100 to represent varying levels of multimodality. All benchmark generator parameters, except the number of promising regions, were set to their default values. Common benchmark settings across all scenarios included a problem dimension of 5, a change frequency of 5000 fitness evaluations, a shift severity of 1, and 100 environmental changes. Similarly, each algorithm was configured according to the default parameter settings recommended in its original publication. For each algorithm–scenario combination, 31 independent runs were performed to ensure statistical robustness. The resulting performance data were analyzed using EDOLAB’s integrated statistical and visualization tools, enabling a detailed examination of trends, robustness, and behavior under different landscape conditions.



Performance analysis was conducted using EDOLAB’s *Statistical Analysis* panel in GUI Mode, which supports three non-parametric hypothesis testing methods [Hollander et al. 2013]: the *Friedman test*, the *Wilcoxon signed-rank test*, and the *Wilcoxon rank-sum test*. By selecting the desired performance indicators and testing methods, users can automatically generate detailed statistical comparison results, including indicator values and associated test statistics.

The results of algorithm performance across the four benchmarks are summarized in Tables 2, 3, 4, and 5. Each table reports the average, standard error, and median of  $E_o$  and  $E_{bbc}$  across 31 runs. Furthermore, all three non-parametric statistical tests are applied to both indicators: the Friedman test provides average ranks along with Nemenyi post-hoc groupings (denoted by letters), while the Wilcoxon signed-rank and rank-sum tests yield win-loss ( $w-l$ ) results for pairwise comparisons. The best-performing entries in each row—according to each statistical test—are highlighted. For example, values tagged with “a” in the Friedman test or those with the highest win-loss count in either Wilcoxon test are highlighted. Note that due to differing statistical assumptions, the three tests may not always identify the same algorithms as the top performers, and minor inconsistencies between their conclusions are to be expected.

We emphasize that the simultaneous use of multiple statistical methods in this analysis is intended to showcase the flexibility and breadth of EDOLAB’s capabilities in statistical evaluation. In typical benchmarking studies, it is more common to select a single test based on the study design and comparison goals. For further discussion of the differences between these methods, see Section 6.3.

Based on the results of our benchmark experiments,  $ACF_{PSO}$  demonstrates the strongest overall performance on the MPB, FPs, and GDBG benchmarks. However, it does not maintain its superiority on GMPB, where it ranks among the second-tier algorithms alongside mPSO and behind  $AMP_{PSO}$  and  $SPSO_{AP+AD}$ .  $SPSO_{AP+AD}$  performs as a top-tier algorithm on GMPB, MPB, and FPs, and is often on par with  $ACF_{PSO}$  on MPB and FPs. However, its performance degrades on the GDBG benchmark, where it does not match the top-performing algorithms.  $AMP_{PSO}$  performs particularly well on GMPB and ranks among the top tier on that benchmark, together with  $SPSO_{AP+AD}$ . In contrast, its performance on MPB and FPs is weaker, placing it in the second tier in those scenarios. mPSO exhibits consistent second-tier performance across all benchmark types.

These trends highlight that  $ACF_{PSO}$  and  $SPSO_{AP+AD}$  frequently deliver competitive results, which is attributed to their resource-allocation strategies that dynamically assign more computational effort to more promising subpopulations. However, their relative effectiveness varies across benchmarks, emphasizing the role of problem characteristics in shaping algorithm performance. It is important to emphasize that these observations are limited to the specific benchmark generators and parameter settings used in this study, and performance rankings may differ under alternative configurations.

Note that the Friedman test often identifies three or more algorithms as statistically indistinguishable top performers; this is a consequence of its conservative nature due to the adjusted  $\alpha$  level used in multiple comparison settings. For additional discussion of these differences and guidance on interpreting the statistical tests used, see Section 6.3.

In terms of computational complexity, among the four algorithms that form the top and second tiers in our tests,  $ACF_{PSO}$ , mPSO, and  $AMP_{PSO}$  all exhibit approximately  $O(N_p^2)$  behavior, where  $N_p$  denotes the number of subpopulations.  $SPSO_{AP+AD}$  incurs a higher complexity of  $O(I^2)$ , where  $I$  is the total number of individuals, due to its internal species clustering procedure. Among these,  $ACF_{PSO}$  offers a relatively favorable balance between performance and computational cost. Nonetheless, the suitability of any algorithm should always be reassessed in the context of the specific problem landscape and application requirements.

In addition to statistical testing, EDOLAB provides visualization tools that facilitate intuitive interpretation of benchmarking outcomes. Specifically, *Trend Plots* depict the evolution of selected performance indicators over time

Table 2. Statistical results on the MPB benchmark with varying numbers of promising regions ( $P = 10, 25, 50, 100$ ), based on 31 runs. Here,  $P$  denotes the number of promising regions. The table reports the average, median, and standard error ( $SE$ ) of  $E_o$  and  $E_{bbc}$ . Performance differences are analyzed using non-parametric tests: Friedman test ranks with Nemenyi post-hoc groups (marked by letters), and Wilcoxon signed-rank and rank-sum tests with win-loss ( $w-l$ ) results. The best performance for each indicator is highlighted. These statistical results were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

$P$	Indicator	Algorithms								
		ACF <sub>PSO</sub>	SPSO <sub>AD+AP</sub>	AMP <sub>PSO</sub>	mPSO	APCPSO	DPCPSO	ImQSO	mDE	psfNBC
10	$\bar{E}_o \pm SE$	1.02 $\pm$ 0.07	1.02 $\pm$ 0.06	1.61 $\pm$ 0.05	1.34 $\pm$ 0.06	3.80 $\pm$ 0.11	3.28 $\pm$ 0.09	2.16 $\pm$ 0.08	1.86 $\pm$ 0.07	2.59 $\pm$ 0.08
	$E_o$ Median	1.06	0.97	1.55	1.39	3.64	3.17	2.09	1.86	2.55
	Friedman Rank	1.61 a	1.61 a	4.00 b	3.00 a	9.00 c	7.97 c	5.90 b	4.97 b	6.94 c
	Signed Rank $w-l$	7	7	2	4	-8	-6	-2	0	-4
	Rank Sum $w-l$	7	7	2	4	-8	-6	-2	0	-4
	$\bar{E}_{bbc} \pm SE$	0.53 $\pm$ 0.06	0.56 $\pm$ 0.06	0.63 $\pm$ 0.06	0.59 $\pm$ 0.06	1.90 $\pm$ 0.08	1.67 $\pm$ 0.08	1.00 $\pm$ 0.07	1.19 $\pm$ 0.07	1.12 $\pm$ 0.07
	$E_{bbc}$ Median	0.50	0.50	0.57	0.51	1.90	1.66	0.99	1.10	1.04
	Friedman Rank	2.16 a	2.68 a	3.13 a	2.65 a	8.77 c	8.06 c	5.29 b	6.35 b	5.90 b
	Signed Rank $w-l$	5	5	5	5	-8	-6	0	-4	-2
	Rank Sum $w-l$	5	5	5	5	-7	-7	-2	-2	-2
25	$\bar{E}_o \pm SE$	1.37 $\pm$ 0.06	1.44 $\pm$ 0.06	2.14 $\pm$ 0.06	1.92 $\pm$ 0.07	3.65 $\pm$ 0.11	3.48 $\pm$ 0.11	2.83 $\pm$ 0.10	2.33 $\pm$ 0.07	3.26 $\pm$ 0.09
	$E_o$ Median	1.30	1.40	2.14	1.90	3.64	3.39	2.81	2.32	3.24
	Friedman Rank	1.45 a	1.55 a	4.13 b	3.13 a	8.65 c	7.97 c	6.10 b	4.81 b	7.23 c
	Signed Rank $w-l$	7	7	2	4	-8	-6	-2	0	-4
	Rank Sum $w-l$	7	7	2	4	-7	-6	-2	0	-5
	$\bar{E}_{bbc} \pm SE$	0.78 $\pm$ 0.05	0.85 $\pm$ 0.06	1.13 $\pm$ 0.05	0.97 $\pm$ 0.06	2.11 $\pm$ 0.09	2.18 $\pm$ 0.10	1.83 $\pm$ 0.09	1.53 $\pm$ 0.07	1.90 $\pm$ 0.08
	$E_{bbc}$ Median	0.73	0.80	1.10	0.92	2.09	2.07	1.77	1.51	1.90
	Friedman Rank	1.71 a	1.81 a	3.74 a	2.81 a	7.81 c	8.32 c	6.68 b	5.32 b	6.81 b
	Signed Rank $w-l$	7	7	2	4	-7	-7	-3	0	-3
	Rank Sum $w-l$	7	6	2	5	-6	-7	-3	0	-4
50	$\bar{E}_o \pm SE$	1.54 $\pm$ 0.04	1.55 $\pm$ 0.04	2.33 $\pm$ 0.05	2.10 $\pm$ 0.05	3.22 $\pm$ 0.07	3.26 $\pm$ 0.07	2.67 $\pm$ 0.07	2.39 $\pm$ 0.05	3.14 $\pm$ 0.06
	$E_o$ Median	1.51	1.57	2.28	2.07	3.18	3.20	2.63	2.37	3.15
	Friedman Rank	1.45 a	1.55 a	4.29 b	3.10 a	8.00 c	8.32 c	5.84 b	4.90 b	7.55 c
	Signed Rank $w-l$	7	7	2	4	-6	-7	-2	0	-5
	Rank Sum $w-l$	7	7	1	4	-6	-6	-2	1	-6
	$\bar{E}_{bbc} \pm SE$	0.89 $\pm$ 0.04	0.91 $\pm$ 0.04	1.37 $\pm$ 0.05	1.12 $\pm$ 0.04	1.93 $\pm$ 0.06	2.17 $\pm$ 0.06	1.83 $\pm$ 0.07	1.52 $\pm$ 0.04	1.88 $\pm$ 0.05
	$E_{bbc}$ Median	0.86	0.85	1.31	1.09	1.85	2.17	1.75	1.52	1.86
	Friedman Rank	1.55 a	1.65 a	4.13 b	2.87 a	7.35 c	8.61 c	6.81 c	5.03 b	7.00 c
	Signed Rank $w-l$	7	7	2	4	-5	-8	-3	0	-4
	Rank Sum $w-l$	7	7	2	4	-4	-8	-4	0	-4
100	$\bar{E}_o \pm SE$	1.54 $\pm$ 0.03	1.56 $\pm$ 0.03	2.37 $\pm$ 0.04	2.20 $\pm$ 0.03	2.85 $\pm$ 0.05	2.94 $\pm$ 0.05	2.55 $\pm$ 0.04	2.38 $\pm$ 0.03	3.16 $\pm$ 0.06
	$E_o$ Median	1.55	1.55	2.35	2.22	2.83	2.91	2.59	2.36	3.12
	Friedman Rank	1.55 a	1.45 a	4.52 b	3.23 a	7.32 c	7.90 c	5.77 b	4.61 b	8.65 c
	Signed Rank $w-l$	7	7	1	4	-4	-6	-2	1	-8
	Rank Sum $w-l$	7	7	1	4	-5	-5	-2	1	-8
	$\bar{E}_{bbc} \pm SE$	0.92 $\pm$ 0.02	0.92 $\pm$ 0.03	1.49 $\pm$ 0.03	1.25 $\pm$ 0.02	1.68 $\pm$ 0.04	1.98 $\pm$ 0.05	1.81 $\pm$ 0.03	1.46 $\pm$ 0.02	1.98 $\pm$ 0.06
	$E_{bbc}$ Median	0.93	0.93	1.46	1.26	1.68	1.97	1.80	1.45	1.91
	Friedman Rank	1.58 a	1.45 a	4.94 b	3.03 a	6.16 b	8.42 c	7.13 c	4.35 b	7.94 c
	Signed Rank $w-l$	7	7	1	4	-2	-7	-4	1	-7
	Rank Sum $w-l$	7	7	1	4	-2	-7	-4	1	-7

(e.g.,  $E_o$  versus the number of fitness evaluations,  $E_{bbc}$  versus the number of environmental changes), averaged across all runs to provide a robust view of overall performance dynamics. *Box Plots* display the distributions of final indicator

Table 3. Statistical results on the GMPB benchmark with varying numbers of promising regions ( $P = 10, 25, 50, 100$ ), based on 31 runs. Here,  $P$  denotes the number of promising regions. The table reports the average, median, and standard error ( $SE$ ) of  $E_o$  and  $E_{bbc}$ . Performance differences are analyzed using non-parametric tests: Friedman test ranks with Nemenyi post-hoc groups (marked by letters), and Wilcoxon signed-rank and rank-sum tests with win-loss ( $w-l$ ) results. The best performance for each indicator is highlighted. These statistical results were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

$P$	Indicator	Algorithms								
		ACF <sub>PSO</sub>	SPSO <sub>AD+AP</sub>	AMP <sub>PSO</sub>	mPSO	APCPSO	DPCPSO	ImQSO	mDE	psfNBC
10	$\bar{E}_o \pm SE$	1.70 ± 0.09	1.41 ± 0.08	1.30 ± 0.05	1.76 ± 0.06	2.37 ± 0.08	2.49 ± 0.07	1.88 ± 0.06	1.87 ± 0.06	2.07 ± 0.07
	$E_o$ Median	1.55	1.30	1.25	1.68	2.27	2.48	1.75	1.83	2.11
	Friedman Rank	4.10 b	2.32 a	1.71 a	4.45 b	7.77 c	8.06 c	5.23 b	5.13 b	6.23 b
	Signed Rank $w-l$	3	7	7	1	-7	-7	0	0	-4
	Rank Sum $w-l$	3	7	7	1	-7	-7	0	-1	-3
	$\bar{E}_{bbc} \pm SE$	1.15 ± 0.09	1.02 ± 0.08	0.71 ± 0.05	1.01 ± 0.05	1.36 ± 0.07	1.61 ± 0.06	1.30 ± 0.06	1.26 ± 0.05	1.17 ± 0.06
	$E_{bbc}$ Median	0.98	0.89	0.64	0.92	1.31	1.63	1.17	1.19	1.14
	Friedman Rank	4.74 b	3.52 a	1.55 a	3.52 a	6.55 b	8.00 c	6.06 b	5.81 b	5.26 b
	Signed Rank $w-l$	1	4	8	4	-4	-8	-2	-2	-1
	Rank Sum $w-l$	3	4	8	4	-4	-8	-3	-3	-1
25	$\bar{E}_o \pm SE$	2.04 ± 0.06	1.83 ± 0.05	1.82 ± 0.04	2.18 ± 0.04	2.79 ± 0.04	3.18 ± 0.05	2.61 ± 0.06	2.27 ± 0.06	3.05 ± 0.08
	$E_o$ Median	1.98	1.77	1.76	2.15	2.75	3.18	2.55	2.23	2.93
	Friedman Rank	3.16 a	2.06 a	1.87 a	3.97 a	6.90 c	8.45 c	6.19 b	4.55 b	7.84 c
	Signed Rank $w-l$	3	7	7	2	-4	-7	-2	1	-7
	Rank Sum $w-l$	4	7	7	1	-4	-7	-2	1	-7
	$\bar{E}_{bbc} \pm SE$	1.50 ± 0.06	1.32 ± 0.05	1.19 ± 0.04	1.40 ± 0.04	1.80 ± 0.04	2.30 ± 0.05	1.97 ± 0.05	1.54 ± 0.05	2.10 ± 0.08
	$E_{bbc}$ Median	1.43	1.31	1.17	1.37	1.79	2.35	1.92	1.51	2.00
	Friedman Rank	4.00 a	2.61 a	1.90 a	3.32 a	6.00 b	8.26 c	7.16 c	4.29 b	7.45 c
	Signed Rank $w-l$	2	6	8	3	-2	-8	-5	1	-5
	Rank Sum $w-l$	2	6	7	4	-2	-8	-5	1	-5
50	$\bar{E}_o \pm SE$	2.27 ± 0.06	1.98 ± 0.05	2.00 ± 0.03	2.34 ± 0.04	2.97 ± 0.05	3.47 ± 0.05	2.84 ± 0.05	2.50 ± 0.04	3.29 ± 0.06
	$E_o$ Median	2.22	1.95	1.96	2.31	2.96	3.46	2.81	2.49	3.24
	Friedman Rank	3.42 a	1.77 a	1.90 a	3.71 a	6.84 c	8.65 c	6.19 b	4.65 b	7.87 c
	Signed Rank $w-l$	3	7	7	3	-3	-8	-3	0	-6
	Rank Sum $w-l$	3	7	7	3	-3	-8	-3	0	-6
	$\bar{E}_{bbc} \pm SE$	1.73 ± 0.05	1.42 ± 0.05	1.36 ± 0.03	1.54 ± 0.04	1.98 ± 0.04	2.61 ± 0.04	2.25 ± 0.05	1.71 ± 0.04	2.30 ± 0.06
	$E_{bbc}$ Median	1.70	1.37	1.35	1.56	1.94	2.57	2.22	1.69	2.28
	Friedman Rank	4.48 b	2.13 a	1.61 a	3.23 a	5.94 b	8.71 c	7.16 c	4.29 b	7.45 c
	Signed Rank $w-l$	1	7	7	4	-2	-8	-5	1	-5
	Rank Sum $w-l$	1	7	7	4	-2	-8	-5	1	-5
100	$\bar{E}_o \pm SE$	2.37 ± 0.04	2.11 ± 0.04	2.15 ± 0.03	2.54 ± 0.02	3.00 ± 0.04	3.56 ± 0.05	3.03 ± 0.04	2.70 ± 0.04	3.27 ± 0.06
	$E_o$ Median	2.32	2.10	2.14	2.52	2.97	3.56	3.07	2.70	3.28
	Friedman Rank	2.90 a	1.65 a	1.84 a	3.94 b	6.42 c	8.65 d	6.71 c	5.16 b	7.74 c
	Signed Rank $w-l$	4	7	7	2	-3	-8	-3	0	-6
	Rank Sum $w-l$	4	7	7	2	-3	-8	-3	0	-6
	$\bar{E}_{bbc} \pm SE$	1.84 ± 0.03	1.53 ± 0.04	1.53 ± 0.03	1.74 ± 0.02	2.05 ± 0.03	2.75 ± 0.04	2.42 ± 0.04	1.90 ± 0.03	2.31 ± 0.05
	$E_{bbc}$ Median	1.82	1.50	1.52	1.72	2.04	2.73	2.42	1.90	2.32
	Friedman Rank	4.26 b	1.71 a	1.74 a	3.32 a	5.58 b	8.77 c	7.55 c	4.90 b	7.16 c
	Signed Rank $w-l$	1	7	7	4	-2	-8	-5	1	-5
	Rank Sum $w-l$	1	7	7	4	-2	-8	-5	1	-5

values across multiple runs, highlighting performance variability and robustness. Both *Trend Plots* and *Box Plots* support user-defined indicators. *Current Error Plots* are also supported and visualize the error of the best-so-far solution

Table 4. Statistical results on the GDBG benchmark with varying numbers of promising regions ( $P = 10, 25, 50, 100$ ), based on 31 runs. Here,  $P$  denotes the number of promising regions. The table reports the average, median, and standard error ( $SE$ ) of  $E_o$  and  $E_{bbc}$ . Performance differences are analyzed using non-parametric tests: Friedman test ranks with Nemenyi post-hoc groups (marked by letters), and Wilcoxon signed-rank and rank-sum tests with win-loss ( $w-l$ ) results. The best performance for each indicator is highlighted. These statistical results were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

$P$	Indicator	Algorithms								
		ACF <sub>PSO</sub>	SPSO <sub>AD+AP</sub>	AMP <sub>PSO</sub>	mPSO	APCPSO	DPCPSO	lmQSO	mDE	psfNBC
10	$\bar{E}_o \pm SE$	$3.78 \pm 0.12$	$5.87 \pm 0.19$	$4.97 \pm 0.13$	$4.85 \pm 0.15$	$13.21 \pm 0.31$	$12.51 \pm 0.32$	$9.62 \pm 0.27$	$6.12 \pm 0.17$	$11.40 \pm 0.22$
	$E_o$ Median	3.76	5.63	4.89	4.72	12.66	11.83	9.28	5.93	11.39
	Friedman Rank	1.03 a	4.23 b	2.65 a	2.55 a	8.84 c	8.00 c	6.06 b	4.55 b	7.10 c
	Signed Rank $w-l$	8	1	5	5	-8	-6	-2	1	-4
	Rank Sum $w-l$	8	1	5	5	-7	-7	-2	1	-4
	$\bar{E}_{bbc} \pm SE$	$1.55 \pm 0.07$	$4.01 \pm 0.16$	$2.11 \pm 0.09$	$1.69 \pm 0.09$	$7.92 \pm 0.21$	$7.12 \pm 0.21$	$5.74 \pm 0.21$	$3.66 \pm 0.13$	$5.13 \pm 0.15$
	$E_{bbc}$ Median	1.60	3.78	2.02	1.62	7.73	6.88	5.73	3.50	5.04
	Friedman Rank	1.42 a	4.84 b	2.90 a	1.68 a	8.84 d	8.13 c	6.58 c	4.35 b	6.26 b
	Signed Rank $w-l$	8	0	4	6	-8	-6	-4	2	-2
	Rank Sum $w-l$	7	1	4	7	-8	-6	-3	1	-3
25	$\bar{E}_o \pm SE$	$2.93 \pm 0.07$	$6.19 \pm 0.15$	$4.23 \pm 0.10$	$4.20 \pm 0.10$	$8.74 \pm 0.14$	$9.37 \pm 0.14$	$8.87 \pm 0.14$	$4.84 \pm 0.12$	$9.54 \pm 0.11$
	$E_o$ Median	2.96	6.29	4.28	4.22	8.77	9.11	8.93	4.88	9.65
	Friedman Rank	1.00 a	4.94 b	2.65 a	2.48 a	6.71 c	8.00 c	6.90 c	3.94 b	8.39 c
	Signed Rank $w-l$	8	0	5	5	-3	-7	-3	2	-7
	Rank Sum $w-l$	8	0	5	5	-3	-7	-3	2	-7
	$\bar{E}_{bbc} \pm SE$	$1.32 \pm 0.05$	$3.55 \pm 0.11$	$2.12 \pm 0.07$	$1.96 \pm 0.08$	$4.89 \pm 0.13$	$5.10 \pm 0.11$	$4.94 \pm 0.12$	$2.68 \pm 0.09$	$4.26 \pm 0.10$
	$E_{bbc}$ Median	1.36	3.58	2.00	1.88	4.86	5.13	5.06	2.59	4.28
	Friedman Rank	1.00 a	5.19 b	2.81 a	2.23 a	7.52 c	8.23 c	7.87 c	3.97 b	6.19 c
	Signed Rank $w-l$	8	0	4	6	-5	-7	-6	2	-2
	Rank Sum $w-l$	8	0	4	6	-6	-6	-6	2	-2
50	$\bar{E}_o \pm SE$	$3.35 \pm 0.05$	$7.12 \pm 0.16$	$4.11 \pm 0.07$	$4.14 \pm 0.07$	$7.52 \pm 0.11$	$8.37 \pm 0.12$	$8.76 \pm 0.18$	$4.46 \pm 0.08$	$8.82 \pm 0.13$
	$E_o$ Median	3.37	7.01	4.20	4.19	7.46	8.31	8.44	4.41	8.65
	Friedman Rank	1.00 a	5.48 b	2.61 a	2.68 a	5.74 b	7.45 c	8.00 c	3.71 b	8.32 c
	Signed Rank $w-l$	8	0	5	5	-2	-4	-7	2	-7
	Rank Sum $w-l$	8	0	5	5	-2	-5	-6	2	-7
	$\bar{E}_{bbc} \pm SE$	$1.77 \pm 0.04$	$4.16 \pm 0.12$	$2.41 \pm 0.04$	$2.39 \pm 0.05$	$3.98 \pm 0.06$	$4.46 \pm 0.06$	$4.94 \pm 0.13$	$2.73 \pm 0.06$	$3.96 \pm 0.08$
	$E_{bbc}$ Median	1.72	4.15	2.38	2.40	3.94	4.43	4.82	2.68	3.97
	Friedman Rank	1.00 a	6.68 c	2.61 a	2.65 a	6.19 c	7.77 c	8.48 d	3.74 b	5.87 b
	Signed Rank $w-l$	8	-3	5	5	-2	-6	-8	2	-1
	Rank Sum $w-l$	8	-2	5	5	-2	-6	-8	2	-2
100	$\bar{E}_o \pm SE$	$3.59 \pm 0.07$	$7.41 \pm 0.13$	$4.00 \pm 0.07$	$4.16 \pm 0.08$	$6.91 \pm 0.12$	$8.10 \pm 0.14$	$8.74 \pm 0.15$	$4.33 \pm 0.08$	$8.14 \pm 0.10$
	$E_o$ Median	3.62	7.38	4.01	4.08	6.90	8.00	8.57	4.31	8.19
	Friedman Rank	1.35 a	6.19 c	2.35 a	2.77 a	5.29 b	7.61 c	8.48 d	3.52 b	7.42 c
	Signed Rank $w-l$	8	-2	6	4	0	-5	-8	2	-5
	Rank Sum $w-l$	8	-2	5	4	0	-5	-8	3	-5
	$\bar{E}_{bbc} \pm SE$	$2.01 \pm 0.05$	$4.16 \pm 0.09$	$2.54 \pm 0.05$	$2.63 \pm 0.05$	$3.59 \pm 0.08$	$4.13 \pm 0.10$	$4.79 \pm 0.11$	$2.78 \pm 0.06$	$3.61 \pm 0.07$
	$E_{bbc}$ Median	2.01	4.18	2.54	2.56	3.62	4.08	4.66	2.83	3.62
	Friedman Rank	1.13 a	7.39 c	2.55 a	2.87 a	5.74 c	7.35 c	8.68 d	3.48 b	5.81 c
	Signed Rank $w-l$	8	-5	5	5	-1	-5	-8	2	-1
	Rank Sum $w-l$	8	-5	5	5	-1	-5	-8	2	-1

within each environment as the algorithm progresses. Averaged across multiple runs, these plots highlight how quickly an algorithm adapts to environmental changes and how effectively it converges within each environment, offering insights into real-time convergence dynamics beyond what is captured by traditional indicators.

Table 5. Statistical results on the FPs benchmark with varying numbers of promising regions ( $P = 10, 25, 50, 100$ ), based on 31 runs. Here,  $P$  denotes the number of promising regions. The table reports the average, median, and standard error (SE) of  $E_o$  and  $E_{bbc}$ . Performance differences are analyzed using non-parametric tests: Friedman test ranks with Nemenyi post-hoc groups (marked by letters), and Wilcoxon signed-rank and rank-sum tests with win-loss ( $w-l$ ) results. The best performance for each indicator is highlighted. These statistical results were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

$P$	Indicator	Algorithms								
		ACF <sub>PSO</sub>	SPSO <sub>AD+AP</sub>	AMP <sub>PSO</sub>	mPSO	APCPSO	DPCPSO	ImQSO	mDE	psfNBC
10	$\bar{E}_o \pm SE$	2.96 $\pm$ 0.47	3.03 $\pm$ 0.41	4.19 $\pm$ 0.52	3.37 $\pm$ 0.44	5.63 $\pm$ 0.61	5.17 $\pm$ 0.57	4.55 $\pm$ 0.51	4.01 $\pm$ 0.48	4.53 $\pm$ 0.49
	$E_o$ Median	2.25	2.46	3.32	2.74	4.57	4.22	3.56	3.22	3.92
	Friedman Rank	1.94 a	2.13 a	4.84 b	2.55 a	8.71 c	7.45 c	6.45 b	4.45 b	6.48 b
	Signed Rank $w-l$	7	7	1	4	-8	-6	-3	1	-3
	Rank Sum $w-l$	6	4	-1	3	-4	-3	-2	0	-3
	$\bar{E}_{bbc} \pm SE$	2.06 $\pm$ 0.44	2.28 $\pm$ 0.39	3.04 $\pm$ 0.45	2.23 $\pm$ 0.38	4.12 $\pm$ 0.47	3.75 $\pm$ 0.45	3.34 $\pm$ 0.44	3.14 $\pm$ 0.43	3.38 $\pm$ 0.43
	$E_{bbc}$ Median	1.06	1.66	2.14	1.48	3.66	3.02	2.69	2.38	3.03
	Friedman Rank	1.97 a	2.68 a	4.58 b	2.23 a	8.61 c	7.23 c	6.23 b	5.13 b	6.35 b
	Signed Rank $w-l$	7	6	2	5	-8	-4	-3	0	-5
	Rank Sum $w-l$	6	4	0	4	-4	-3	-3	-1	-3
25	$\bar{E}_o \pm SE$	2.10 $\pm$ 0.21	2.18 $\pm$ 0.23	2.85 $\pm$ 0.24	2.47 $\pm$ 0.24	3.31 $\pm$ 0.30	3.46 $\pm$ 0.32	2.85 $\pm$ 0.26	2.64 $\pm$ 0.26	3.31 $\pm$ 0.25
	$E_o$ Median	1.92	1.83	2.35	2.14	2.79	2.98	2.39	2.22	3.16
	Friedman Rank	1.81 a	1.81 a	5.48 b	3.32 a	7.45 c	7.84 c	5.35 b	4.13 b	7.81 c
	Signed Rank $w-l$	7	7	-1	4	-5	-7	-1	2	-6
	Rank Sum $w-l$	5	4	-2	3	-3	-4	-1	2	-4
	$\bar{E}_{bbc} \pm SE$	1.43 $\pm$ 0.18	1.68 $\pm$ 0.21	2.04 $\pm$ 0.21	1.63 $\pm$ 0.19	2.42 $\pm$ 0.25	2.64 $\pm$ 0.27	2.14 $\pm$ 0.22	1.88 $\pm$ 0.21	2.54 $\pm$ 0.23
	$E_{bbc}$ Median	1.34	1.56	1.79	1.45	2.10	2.34	1.89	1.56	2.39
	Friedman Rank	1.77 a	2.87 a	5.00 b	2.71 a	7.10 c	7.94 c	5.68 b	4.10 b	7.84 c
	Signed Rank $w-l$	8	5	-1	5	-5	-7	-1	2	-6
	Rank Sum $w-l$	5	3	-1	3	-3	-4	-1	2	-4
50	$\bar{E}_o \pm SE$	2.01 $\pm$ 0.14	1.86 $\pm$ 0.14	2.53 $\pm$ 0.15	2.11 $\pm$ 0.15	2.73 $\pm$ 0.17	2.88 $\pm$ 0.19	2.47 $\pm$ 0.15	2.18 $\pm$ 0.16	3.03 $\pm$ 0.19
	$E_o$ Median	1.92	1.66	2.54	1.98	2.68	2.87	2.38	2.00	3.09
	Friedman Rank	2.77 a	2.03 a	5.65 b	3.35 a	6.87 b	7.23 b	5.55 b	3.65 a	7.90 c
	Signed Rank $w-l$	5	8	-1	4	-5	-6	-1	3	-7
	Rank Sum $w-l$	5	5	-3	4	-4	-4	-1	3	-5
	$\bar{E}_{bbc} \pm SE$	1.60 $\pm$ 0.12	1.51 $\pm$ 0.13	1.94 $\pm$ 0.13	1.56 $\pm$ 0.12	2.20 $\pm$ 0.14	2.36 $\pm$ 0.16	2.06 $\pm$ 0.13	1.63 $\pm$ 0.13	2.41 $\pm$ 0.18
	$E_{bbc}$ Median	1.45	1.48	1.89	1.49	2.21	2.38	1.89	1.59	2.29
	Friedman Rank	2.94 a	2.68 a	5.13 b	2.87 a	6.90 b	7.39 c	6.16 b	3.39 a	7.55 c
	Signed Rank $w-l$	5	5	-1	5	-5	-6	-2	5	-6
	Rank Sum $w-l$	5	5	-1	5	-4	-5	-4	4	-5
100	$\bar{E}_o \pm SE$	2.29 $\pm$ 0.13	2.10 $\pm$ 0.12	2.50 $\pm$ 0.11	2.34 $\pm$ 0.12	2.82 $\pm$ 0.11	3.15 $\pm$ 0.14	2.88 $\pm$ 0.12	2.27 $\pm$ 0.12	3.40 $\pm$ 0.15
	$E_o$ Median	2.15	1.90	2.45	2.23	2.74	3.12	2.79	2.16	3.43
	Friedman Rank	3.39 a	2.48 a	4.35 a	3.26 a	6.13 b	7.61 b	6.48 b	3.32 a	7.97 b
	Signed Rank $w-l$	4	7	0	5	-3	-6	-3	4	-8
	Rank Sum $w-l$	4	5	3	4	-4	-5	-4	4	-7
	$\bar{E}_{bbc} \pm SE$	1.96 $\pm$ 0.12	1.80 $\pm$ 0.11	2.03 $\pm$ 0.10	1.87 $\pm$ 0.10	2.42 $\pm$ 0.09	2.67 $\pm$ 0.12	2.57 $\pm$ 0.11	1.79 $\pm$ 0.11	2.89 $\pm$ 0.14
	$E_{bbc}$ Median	1.87	1.67	1.96	1.85	2.32	2.75	2.51	1.61	2.85
	Friedman Rank	3.77 a	3.35 a	3.84 a	2.97 a	6.32 b	7.48 b	6.84 b	2.74 a	7.68 b
	Signed Rank $w-l$	4	5	1	5	-3	-6	-4	5	-7
	Rank Sum $w-l$	4	4	4	4	-4	-5	-5	4	-6

To illustrate these capabilities, we present results from a benchmark scenario generated by MPB with 25 promising regions. Figure 2 displays the average trends of  $E_o$  and  $E_{bbc}$  across 31 runs for each of the nine algorithms, plotted

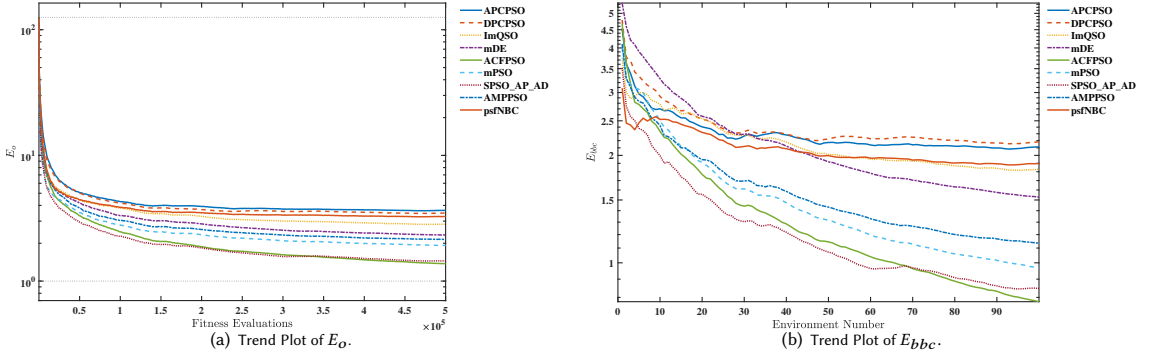


Fig. 2. Trend plots showing the evolution of offline error ( $E_o$ ) over fitness evaluations and best-before-change error ( $E_{bbc}$ ) over environment index for nine EDOAs on the MPB benchmark with 25 promising regions. Curves represent averages over 31 independent runs. These visualizations were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

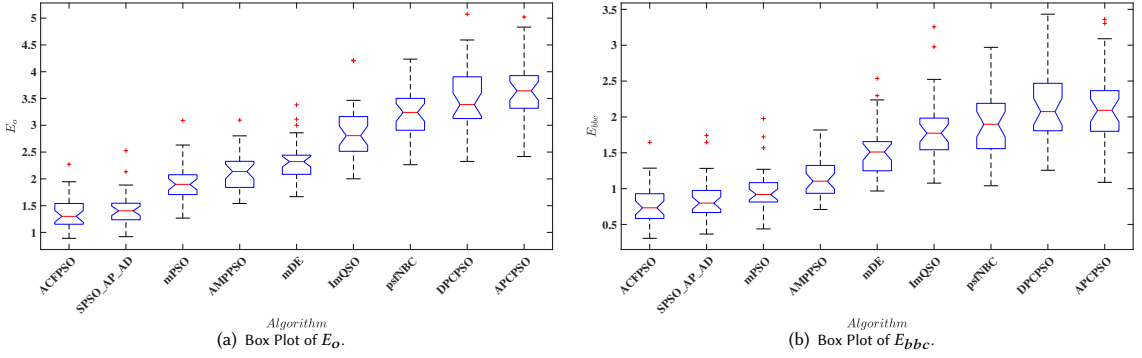


Fig. 3. Box plots comparing the final offline error ( $E_o$ ) and best-before-change error ( $E_{bbc}$ ) for nine EDOAs on the MPB benchmark with 25 promising regions. The plots summarize performance distributions over 31 runs. These plots were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

against the number of fitness evaluations and environment indices, respectively. Figure 3 presents the corresponding box plots, summarizing the distributions of  $E_o$  and  $E_{bbc}$  across 31 independent runs for each algorithm. The tighter boxes and shorter whiskers observed for ACFPSO, SPSO<sub>AP+AD</sub>, mPSO, and AMPPSO highlight their superior stability under this highly multimodal problem instance. Figure 4 presents the *Current Error Plots*, which show the current error of the best-so-far solution in each environment, plotted over fitness evaluations and averaged across 31 independent runs for each algorithm. Unlike  $E_o$  and  $E_{bbc}$ , this plot captures real-time convergence dynamics within each environment and reveals how quickly algorithms adapt and re-converge after environmental changes. Spikes in the plots correspond to environmental change points, after which the error rises and then gradually decreases as the algorithm readapts.

## 6.2 Parameter Tuning and Sensitivity Analysis Example

EDOLAB's Statistical Analysis panel also supports parameter tuning and sensitivity studies by enabling performance comparisons of an algorithm under different configurations. Within EDOLAB's GUI Mode, users can configure a series

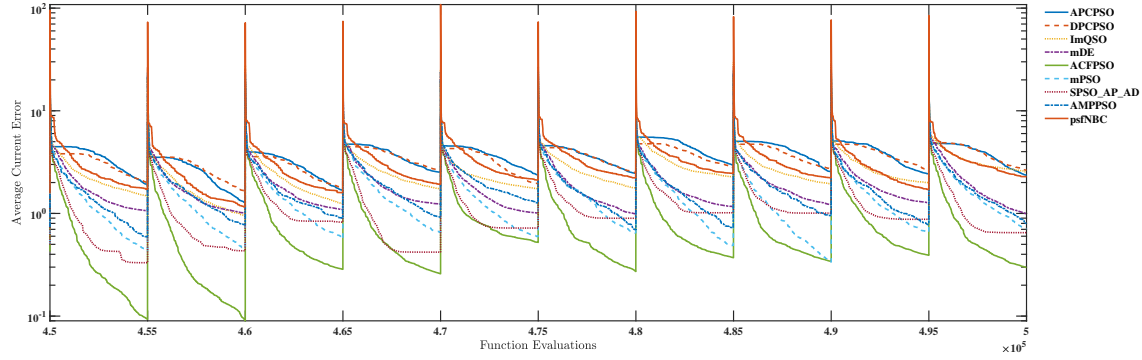


Fig. 4. Trend of current error over fitness evaluations for each algorithm on the GMPB benchmark with 100 promising regions, averaged across 31 runs. Although the experiment spans 100 environments, only the *final* 10 are shown here to enhance readability. Note that the change frequency is set to 5000 fitness evaluations. This plot was generated using the Statistical Analysis panel in GUI Mode.

of experimental tasks where the same algorithm is tested with varying parameter values under identical problem instances. Once these experiments are complete, the Statistical Analysis panel can be used to perform significance testing and generate result plots—following the same workflow as for comparing different algorithms. When all compared tasks involve the same algorithm and problem instance, EDOLAB automatically detects this and allows users to specify the parameter of interest for analysis. This selected parameter is then used to generate appropriate legends, axis labels, and column headers in the resulting visualizations and summary tables.

To demonstrate this functionality, we conduct an experiment to evaluate the effect of the subpopulation size parameter ( $N$ ) in mQSO [Blackwell and Branke 2006]—a widely studied classic algorithm in the field—on its performance. Experiments are performed on the GMPB benchmark under four configurations (10, 25, 50, and 100 promising regions), with  $N$  varied across five values (5, 10, 20, 30, and 50). All other parameters of the GMPB benchmark and the mQSO algorithm are set to their default values. The results are then analyzed using EDOLAB’s Statistical Analysis panel.

The results in Table 6 indicate that, under the selected experimental settings, a subpopulation size of  $N = 10$  yields the best overall performance for mQSO across different levels of problem complexity, with  $N = 20$  performing comparably in many cases. Subpopulation sizes in this range appear to offer a good trade-off between rapid adaptation to environmental changes and accurate search within each environment. Larger subpopulation sizes (e.g.,  $N = 50$ ) tend to slow convergence, as the number of fitness evaluations per generation increases, which reduces the algorithm’s responsiveness and limits its ability to recover quickly after changes.

Figure 5 shows box plots of  $E_o$  and  $E_{bbc}$  for different subpopulation sizes ( $N = 5, 10, 20, 30, 50$ ) of mQSO on the GMPB benchmark with 25 promising regions, based on 31 independent runs. These plots visually confirm the trends observed in Table 6, highlighting  $N = 10$  and  $N = 20$  as the most stable and best-performing configurations under the selected conditions.

In addition to identifying configurations with relatively strong performance, the results provide insights into the sensitivity of mQSO’s behavior with respect to the subpopulation size parameter. Subpopulation sizes between 10 and 20 show broadly stable performance, as supported by Friedman rankings and Wilcoxon-based pairwise comparisons, suggesting a degree of robustness to moderate variations in this parameter.

Table 6. Analyzing the impact of subpopulation size on mQSO [Blackwell and Branke 2006] performance and conducting sensitivity analysis on GMPB. Results show the effect of varying subpopulation sizes ( $N = 5, 10, 20, 30, 50$ ) across different numbers of promising regions ( $P = 10, 25, 50, 100$ ), evaluated using  $E_o$  and  $E_{bbc}$  (averages, medians, and standard errors ( $SE$ ) over 31 runs). Statistical significance is assessed through the Friedman test with Nemenyi post-hoc grouping (marked by letters) and Wilcoxon win-loss ( $w-l$ ) counts. Best-performing configurations are highlighted. These statistical results were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

$P$	Indicator	Subpopulation Size ( $N$ ) in mQSO				
		5	10	20	30	50
10	$\bar{E}_o \pm SE$	$1.90 \pm 0.07$	$1.55 \pm 0.06$	$1.57 \pm 0.04$	$1.64 \pm 0.06$	$2.04 \pm 0.06$
	$E_o$ Median	1.82	1.49	1.49	1.57	2.03
	Friedman Rank	3.58 b	1.94 a	2.23 a	2.61 a	4.65 b
	Signed Rank $w-l$	-2	3	2	1	-4
	Rank Sum $w-l$	-3	2	2	2	-3
	$\bar{E}_{bbc} \pm SE$	$1.28 \pm 0.07$	$0.96 \pm 0.05$	$0.85 \pm 0.04$	$0.81 \pm 0.04$	$1.01 \pm 0.05$
	$E_{bbc}$ Median	1.21	0.94	0.80	0.77	0.98
	Friedman Rank	4.32 c	3.00 b	2.23 a	1.87 a	3.58 b
	Signed Rank $w-l$	-4	0	2	3	-1
	Rank Sum $w-l$	-4	0	2	3	-1
25	$\bar{E}_o \pm SE$	$2.59 \pm 0.06$	$2.35 \pm 0.05$	$2.42 \pm 0.06$	$2.55 \pm 0.07$	$2.98 \pm 0.07$
	$E_o$ Median	2.56	2.35	2.36	2.53	2.85
	Friedman Rank	3.29 b	1.87 a	2.32 a	2.94 a	4.58 c
	Signed Rank $w-l$	-1	3	2	0	-4
	Rank Sum $w-l$	-1	3	2	0	-4
	$\bar{E}_{bbc} \pm SE$	$1.97 \pm 0.05$	$1.75 \pm 0.04$	$1.75 \pm 0.06$	$1.82 \pm 0.07$	$2.10 \pm 0.07$
	$E_{bbc}$ Median	1.96	1.69	1.77	1.77	2.04
	Friedman Rank	3.58 b	2.23 a	2.16 a	2.90 a	4.13 b
	Signed Rank $w-l$	-2	2	2	1	-3
	Rank Sum $w-l$	-2	2	2	1	-3
50	$\bar{E}_o \pm SE$	$2.86 \pm 0.05$	$2.68 \pm 0.04$	$2.83 \pm 0.06$	$2.98 \pm 0.05$	$3.56 \pm 0.08$
	$E_o$ Median	2.79	2.65	2.83	2.96	3.46
	Friedman Rank	2.58 a	1.90 a	2.42 a	3.26 b	4.84 c
	Signed Rank $w-l$	1	4	1	-2	-4
	Rank Sum $w-l$	0	3	2	-1	-4
	$\bar{E}_{bbc} \pm SE$	$2.25 \pm 0.04$	$2.08 \pm 0.04$	$2.17 \pm 0.05$	$2.26 \pm 0.05$	$2.67 \pm 0.07$
	$E_{bbc}$ Median	2.23	2.05	2.15	2.27	2.60
	Friedman Rank	2.94 a	2.00 a	2.45 a	3.06 a	4.55 b
	Signed Rank $w-l$	0	3	1	0	-4
	Rank Sum $w-l$	0	3	1	0	-4
100	$\bar{E}_o \pm SE$	$2.97 \pm 0.04$	$2.90 \pm 0.05$	$3.01 \pm 0.05$	$3.23 \pm 0.06$	$3.75 \pm 0.09$
	$E_o$ Median	2.97	2.86	2.98	3.18	3.73
	Friedman Rank	2.29 a	1.77 a	2.58 a	3.61 b	4.74 c
	Signed Rank $w-l$	2	2	2	-2	-4
	Rank Sum $w-l$	2	2	2	-2	-4
	$\bar{E}_{bbc} \pm SE$	$2.38 \pm 0.04$	$2.30 \pm 0.05$	$2.38 \pm 0.05$	$2.54 \pm 0.06$	$2.92 \pm 0.08$
	$E_{bbc}$ Median	2.36	2.25	2.35	2.49	2.90
	Friedman Rank	2.68 a	2.06 a	2.65 a	3.19 b	4.42 c
	Signed Rank $w-l$	2	2	2	-2	-4
	Rank Sum $w-l$	0	3	1	0	-4



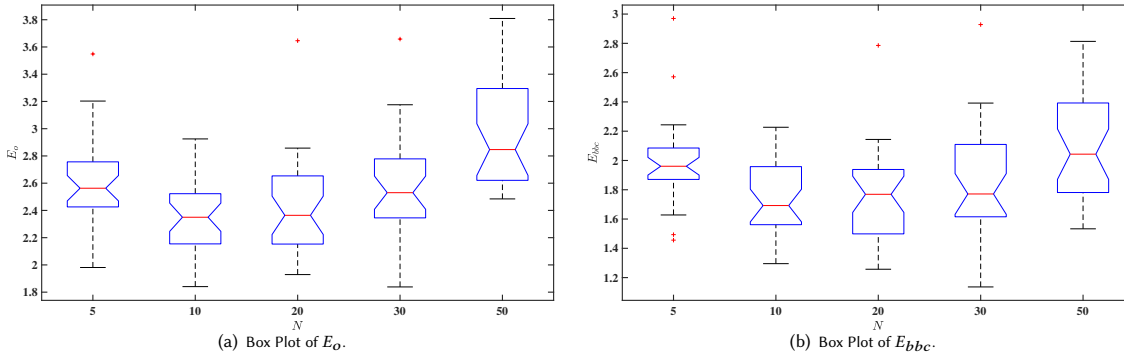


Fig. 5. Box plots comparing the final offline error ( $E_o$ ) for mQSO [Blackwell and Branke 2006] under different subpopulation sizes ( $N$ ) on the GMPB benchmark with 25 promising regions. The plots summarize performance distributions over 31 runs and were generated using the Statistical Analysis panel in the GUI Mode of EDOLAB.

### 6.3 A Note on Statistical Testing Methods and Their Interpretation

In the current version of EDOLAB, three widely used statistical tests are available for comparing algorithmic performance: the Friedman test, the Wilcoxon signed-rank test, and the Wilcoxon rank-sum test [Hollander et al. 2013]. Each serves a distinct purpose and offers different trade-offs in terms of statistical rigor and granularity.

The Friedman test is a non-parametric multiple comparison procedure designed for comparing more than two related groups (i.e., algorithms tested on the same tasks). It provides global rankings of the algorithms and identifies statistically significant differences across all competitors simultaneously. However, multiple comparison procedures like the Friedman test must correct for the increased risk of Type I errors (false positives) that arise when conducting many simultaneous comparisons. This correction—typically applied through post-hoc methods such as the Nemenyi or Holm-Bonferroni adjustment—results in a more conservative threshold for declaring statistical significance. Importantly, the conservativeness of the test increases with the number of algorithms under comparison, as the adjusted  $\alpha$  becomes smaller to account for the larger number of pairwise tests. In our experiments, where nine algorithms are compared, this effect is particularly noticeable: the Friedman test often identifies broad groupings of three or more algorithms as statistically indistinguishable top performers, even if their individual differences are modest.

By contrast, the Wilcoxon signed-rank and Wilcoxon rank-sum tests are pairwise comparison methods. Since they assess differences between two algorithms at a time, they do not inherently require correction for multiple comparisons unless interpreted within a broader multiple testing framework. This allows them to operate with a less conservative threshold per test, often leading to sharper distinctions between algorithms and fewer ties. As a result, it is typical to observe that Wilcoxon-based win-loss comparisons highlight fewer best-performing algorithms than the Friedman test, even when both tests are applied to the same dataset.

In our analysis, these two Wilcoxon-based tests yield slightly different win-loss counts due to their underlying assumptions: the signed-rank test accounts for pairing between samples and thus captures consistent performance trends more effectively, whereas the rank-sum test is more sensitive to distributional differences when treating the samples independently. While EDOLAB offers both Wilcoxon signed-rank and rank-sum tests for pairwise analysis, note that for benchmarking experiments where algorithms are tested under identical tasks, the Wilcoxon signed-rank test is generally more statistically appropriate due to the paired nature of the data and its higher statistical power.

In practice, it is customary to select a single test based on the study’s goals and data characteristics—typically the Friedman test when a global comparison across multiple algorithms is desired, or one of the Wilcoxon tests when finer pairwise differences are of interest [Bartz-Beielstein et al. 2010; Demšar 2006; Derrac et al. 2011]. In this study, we present results from all three methods not to recommend redundant testing but to illustrate the breadth of EDOLAB’s statistical capabilities and to provide a comprehensive example for users.

## 7 Conclusion

Evolutionary dynamic optimization algorithms (EDOAs) and dynamic benchmark generators are inherently complex, making their re-implementation both time-consuming and error-prone. Over the past two decades, the lack of standardized platforms and the limited availability of public source codes have posed serious challenges for researchers attempting to reproduce results or conduct meaningful algorithmic comparisons.

To address these issues, we presented *EDOLAB*, an open-source MATLAB platform specifically designed to support experimentation, evaluation, and education in evolutionary dynamic optimization. EDOLAB provides a standardized and extensible environment that lowers the entry barrier for new researchers and improves reproducibility in the field.

The current version of EDOLAB includes 27 EDOAs, four configurable benchmark generators, and two widely used performance indicators. It is structured around two core applications: the *Education* application, which focuses on visualizing algorithm behavior in dynamic landscapes; and the *Experimentation* application, which supports full-cycle experimentation, including parameter configuration, parallel execution, statistical analysis, ranking, and sensitivity analysis. Users can interact with the platform through a graphical interface or via direct scripting, depending on their preference.

This paper outlined the structure and usage of EDOLAB, introduced its modular architecture, and provided guidance for extending the platform with new algorithms, benchmarks, and indicators. While this manuscript presents a high-level overview, a comprehensive user manual accompanies EDOLAB, offering detailed technical documentation, usage instructions, architectural diagrams, and practical examples to help users effectively interact with and extend the platform.

Future directions include expanding EDOLAB to support additional subfields of dynamic optimization, such as dynamic multi-objective optimization [Chen et al. 2017; Ruan et al. 2025, 2024], dynamic constrained optimization [Nguyen and Yao 2012], dynamic multimodal optimization [Cheng et al. 2019], and robust optimization [Yazdani et al. 2024b] over time. Another important direction is re-implementing the platform in Python to increase accessibility and broaden its user base in the research community.

## References

- Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss. 2010. *Experimental methods for the analysis of optimization algorithms*. Springer.
- Jon Louis Bentley and Jerome H. Friedman. 1979. Data Structures for Range Searching. *Comput. Surveys* 11, 4 (1979), 397–409.
- Tim Blackwell. 2007. *Particle Swarm Optimization in Dynamic Environments*. Springer Berlin Heidelberg, 29–49.
- Tim Blackwell and Juergen Branke. 2004. Multi-swarm Optimization in Dynamic Environments. In *Applications of Evolutionary Computing*, Günther R. Raidl et al. (Eds.). Vol. 3005. Lecture Notes in Computer Science, 489–500.
- Tim Blackwell and Juergen Branke. 2006. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation* 10, 4 (2006), 459–472.
- Tim Blackwell, Juergen Branke, and Xiaodong Li. 2008. Particle swarms for dynamic optimization problems. In *Swarm Intelligence: Introduction and Applications*, Christian Blum and Daniel Merkle (Eds.). Springer Lecture Notes in Computer Science, 193–217.
- Mohammad Reza Bonyadi and Zbigniew Michalewicz. 2017. Particle swarm optimization for single objective continuous space problems: a review. *Evolutionary Computation* 25, 1 (2017), 1–54.

- Juergen Branke. 1999. Memory enhanced evolutionary algorithms for changing optimization problems. In *IEEE Congress on Evolutionary Computation*, Vol. 3. IEEE, 1875–1882.
- Juergen Branke. 2012. *Evolutionary optimization in dynamic environments*. Vol. 3. Springer Science & Business Media.
- Juergen Branke and Hartmut Schmeck. 2003. Designing Evolutionary Algorithms for Dynamic Optimization Problems. In *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui (Eds.). Springer Natural Computing Series, 239–262.
- Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. 2006. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation* 10, 6 (2006), 646–657.
- Chenyang Bu, Wenjian Luo, and Lihua Yue. 2016. Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies. *IEEE Transactions on Evolutionary Computation* 21, 1 (2016), 14–33.
- Renzhi Chen, Ke Li, and Xin Yao. 2017. Dynamic multiobjectives optimization with a changing number of objectives. *IEEE Transactions on Evolutionary Computation* 22, 1 (2017), 157–171.
- Shi Cheng, Hui Lu, Yi-nan Guo, Xiujuan Lei, Jing Liang, Junfeng Chen, and Yuhui Shi. 2019. Dynamic multimodal optimization: A preliminary study. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 279–285.
- Swagatam Das and Ponnuthurai Nagarathnam Suganthan. 2010. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation* 15, 1 (2010), 4–31.
- Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7, Jan (2006), 1–30.
- Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1, 1 (2011), 3–18.
- Mathys C. Du Plessis and Andries P. Engelbrecht. 2012. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research* 218, 1 (2012), 7–20.
- Mathys C. du Plessis and Andries P. Engelbrecht. 2013. Differential evolution for dynamic environments with unknown numbers of optima. *Journal of Global Optimization* 55, 1 (2013), 73–99.
- Julien Georges Omer Louis Duhain. 2012. *Particle swarm optimisation in dynamically changing environments - an empirical study*. Master's thesis. University of Pretoria, Pretoria, South Africa.
- Russell C. Eberhart and Yuhui Shi. 2001a. Comparing inertia weights and constriction factors in particle swarm optimization. In *Congress on Evolutionary Computation*, Vol. 1. IEEE, 84–88.
- Russell C. Eberhart and Yuhui Shi. 2001b. Tracking and optimizing dynamic systems with particle swarms. In *Congress on Evolutionary Computation*, Vol. 1. IEEE, 94–100.
- Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195.
- Sabine Helwig, Juergen Branke, and Sanaz Mostaghim. 2012. Experimental analysis of bound handling techniques in particle swarm optimization. *IEEE Transactions on Evolutionary computation* 17, 2 (2012), 259–271.
- Myles Hollander, Douglas A Wolfe, and Eric Chicken. 2013. *Nonparametric statistical methods*. John Wiley & Sons.
- Xiaohui Hu and Russell C. Eberhart. 2002. Adaptive particle swarm optimization: detection and response to dynamic systems. In *Congress on Evolutionary Computation*, Vol. 2. IEEE, 1666–1670.
- Shouyong Jiang, Juan Zou, Shengxiang Yang, and Xin Yao. 2022. Evolutionary dynamic multi-objective optimisation: A survey. *Comput. Surveys* 55, 4 (2022), 1–47.
- Yaochu Jin and Juergen Branke. 2005. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on evolutionary computation* 9, 3 (2005), 303–317.
- Sami Kaddani, Daniel Vanderpooten, Jean-Michel Vanpeperstraete, and Hassene Aissi. 2017. Weighted sum model with partial preference information: application to multi-objective optimization. *European Journal of Operational Research* 260, 2 (2017), 665–679.
- Masoud Kamosi, Ali Baradaran Hashemi, and Mohommad Reza Meybodi. 2010. A hibernating multi-swarm optimization algorithm for dynamic environments. In *Nature and Biologically Inspired Computing*. IEEE, 363–369.
- Javidan Kazemi Kordestani, Mohammad Reza Meybodi, and Amir Masoud Rahmani. 2019. A note on the exclusion operator in multi-swarm PSO algorithms for dynamic environments. *Connection Science* (2019), 1–25.
- Changhe Li, Trung Thanh Nguyen, Ming Yang, Michalis Mavrouniotis, and Shengxiang Yang. 2016. An Adaptive Multipopulation Framework for Locating and Tracking Multiple Optima. *IEEE Transactions on Evolutionary Computation* 20, 4 (2016), 590–605.
- Changhe Li, Trung Thanh Nguyen, Ming Yang, Shengxiang Yang, and Sanyou Zeng. 2015. Multi-population methods in unconstrained continuous dynamic environments: the challenges. *Information Sciences* 296 (2015), 95 – 118.
- Changhe Li, Trung Thanh Nguyen, Sanyou Zeng, Ming Yang, and Min Wu. 2018. An Open Framework for Constructing Continuous Optimization Problems. *IEEE Transactions on Cybernetics* 49, 6 (2018).
- Changhe Li and Shengxiang Yang. 2008. Fast Multi-Swarm Optimization for Dynamic Optimization Problems. In *International Conference on Natural Computation*, Vol. 7. IEEE, 624–628.
- Changhe Li and Shengxiang Yang. 2012. A General Framework of Multipopulation Methods With Clustering in Undetectable Dynamic Environments. *IEEE Transactions on Evolutionary Computation* 16, 4 (2012), 556–577.

- Changhe Li, Shengxiang Yang, Trung Thanh Nguyen, E. Ling Yu, Xin Yao, Yaochu Jin, Hans-Georg Beyer, and Ponnuthurai N. Suganthan. 2008. *Benchmark Generator for CEC'2009 Competition on Dynamic Optimization*. Technical Report. Center for Computational Intelligence.
- Changhe Li, Shengxiang Yang, and Ming Yang. 2014. An Adaptive Multi-Swarm Optimizer for Dynamic Optimization Problems. *Evolutionary Computation* 22, 4 (2014), 559–594.
- Fei Li, Qiang Yue, Yuanchao Liu, Haibin Ouyang, and Fangqing Gu. 2024. A fast density peak clustering based particle swarm optimizer for dynamic optimization. *Expert Systems with Applications* 236 (2024), 121254.
- Jing Liang, Ponnuthurai N. Suganthan, and Kalyan Deb. 2005. Novel composition test functions for numerical global optimization. In *Swarm Intelligence Symposium*. IEEE, 68–75.
- Yuanchao Liu, Jianchang Liu, Yaochu Jin, Fei Li, and Tianzi Zheng. 2020. An affinity propagation clustering based particle swarm optimizer for dynamic optimization. *Knowledge-Based Systems* 195 (2020), 105711.
- Rodica Ioana Lung and Dumitru Dumitrescu. 2007. A collaborative model for tracking optima in dynamic environments. In *Congress on Evolutionary Computation*. IEEE, 564–567.
- Wenjian Luo, Juan Sun, Chenyang Bu, and Ruikang Yi. 2018. Identifying Species for Particle Swarm Optimization under Dynamic Environments. In *Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1921–1928.
- Wenjian Luo, Bin Yang, Chenyang Bu, and Xin Lin. 2017. A Hybrid Particle Swarm Optimization for High-Dimensional Dynamic Optimization. In *Simulated Evolution and Learning*, Yuhui Shi et al. (Ed.). Springer International Publishing, Cham, 981–993.
- Timothy Marler and Jasbir S. Arora. 2010. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization* 41, 6 (2010), 853–862.
- Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. 2017. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation* 33 (2017), 1 – 17.
- Rui Mendes and Arvind Mohais. 2005. DynDE: a differential evolution for dynamic optimization problems. In *Congress on Evolutionary Computation*, Vol. 3. IEEE, 2808–2815.
- Efrén Mezura-Montes and Carlos A Coello Coello. 2011. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation* 1, 4 (2011), 173–194.
- Trung Thanh Nguyen. 2011. *Continuous dynamic optimisation using evolutionary algorithms*. Ph. D. Dissertation. University of Birmingham.
- Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. 2012. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation* 6 (2012), 1 – 24.
- Trung Thanh Nguyen and Xin Yao. 2012. Continuous dynamic constrained optimization—The challenges. *IEEE Transactions on Evolutionary Computation* 16, 6 (2012), 769–786.
- Daniel Parrott and Xiaodong Li. 2006. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation* 10, 4 (2006), 440–458.
- Carlo Raquel and Xin Yao. 2013. Dynamic multi-objective optimization: a survey of the state-of-the-art. In *Evolutionary computation for dynamic optimization problems*. Springer, 85–106.
- Gan Ruan, Zhanglu Hou, and Xin Yao. 2025. Coping With a Severely Changing Number of Objectives in Dynamic Multi-Objective Optimization. *IEEE Transactions on Evolutionary Computation* (2025).
- Gan Ruan, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. 2024. Knowledge transfer for dynamic multi-Objective optimization With a changing number of objectives. *IEEE Transactions on Emerging Topics in Computational Intelligence* 8, 6 (2024), 4210–4224.
- Ponnuthurai N. Suganthan, Nikolaus Hansen, Jing Liang, Kalyan Deb, Ying ping Chen, Anne Auger, and S Tiwari. 2005. *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*. Technical Report. Nanyang Technological University.
- Rene Thomsen. 2004. Multimodal optimization using crowding-based differential evolution. In *Congress on Evolutionary Computation*, Vol. 2. IEEE, 1382–1389.
- Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. 2017. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine* 12, 4 (2017), 73–87.
- Krzysztof Trojanowski and Zbigniew Michalewicz. 1999. Searching for optima in non-stationary environments. In *Congress on Evolutionary Computation*, Vol. 3. 1843–1850.
- Hongfeng Wang, Dingwei Wang, and Shengxiang Yang. 2007. Triggered Memory-Based Swarm Optimization in Dynamic Environments. In *Applications of Evolutionary Computing*, Mario Giacobini (Ed.). Springer Berlin Heidelberg, 637–646.
- Shengxiang Yang and Changhe Li. 2010. A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments. *IEEE Transactions on Evolutionary Computation* 14, 6 (2010), 959–974.
- Danial Yazdani. 2018. *Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost*. Ph. D. Dissertation. Liverpool John Moores University, Liverpool, UK.
- Danial Yazdani, Juergen Branke, Mohammad Sadegh Khorshidi, Mohammad Nabi Omidvar, Xiaodong Li, Amir H Gandomi, and Xin Yao. 2024a. Clustering in dynamic environments: a framework for benchmark dataset generation with heterogeneous changes. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 50–58.
- Danial Yazdani, Juergen Branke, Mohammad Nabi Omidvar, Xiaodong Li, Changhe Li, Michalis Mavrovouniotis, Trung Thanh Nguyen, Shengxiang Yang, and Xin Yao. 2021a. IEEE CEC 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark. *arXiv*

- preprint *arXiv:2106.06174* (2021).
- Danial Yazdani, Juergen Branke, Mohammad Nabi Omidvar, Trung Thanh Nguyen, and Xin Yao. 2018a. Changing or Keeping Solutions in Dynamic Optimization Problems with Switching Costs. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 1095–1102.
- Danial Yazdani, Ran Cheng, Cheng He, and Juergen Branke. 2020a. Adaptive control of subpopulations in evolutionary dynamic optimization. *IEEE Transactions on Cybernetics* 52, 7 (2020), 6476–6489.
- Danial Yazdani, Ran Cheng, Donya Yazdani, Jürgen Branke, Yaochu Jin, and Xin Yao. 2021b. A Survey of Evolutionary Continuous Dynamic Optimization Over Two Decades – Part A. *IEEE Transactions on Evolutionary Computation* 25, 4 (2021), 609–629.
- Danial Yazdani, Ran Cheng, Donya Yazdani, Jürgen Branke, Yaochu Jin, and Xin Yao. 2021c. A Survey of Evolutionary Continuous Dynamic Optimization Over Two Decades – Part B. *IEEE Transactions on Evolutionary Computation* 25, 4 (2021), 630–650.
- Danial Yazdani, Babak Nasiri, Alireza Sepas-Moghaddam, and Mohammad Reza Meybodi. 2013. A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing* 13, 4 (2013), 2144–2158.
- Danial Yazdani, Trung Thanh Nguyen, and Juergen Branke. 2018b. Robust optimization over time by learning problem space characteristics. *IEEE Transactions on Evolutionary Computation* 23, 1 (2018), 143–155.
- Danial Yazdani, Trung Thanh Nguyen, Juergen Branke, and Jin Wang. 2017. A New Multi-swarm Particle Swarm Optimization for Robust Optimization Over Time. In *Applications of Evolutionary Computation*, Giovanni Squillero and Kevin Sim (Eds.). Springer International Publishing, 99–109.
- Danial Yazdani, Mohammad Nabi Omidvar, Jürgen Branke, Trung Thanh Nguyen, and Xin Yao. 2019. Scaling up dynamic optimization problems: A divide-and-conquer approach. *IEEE Transactions on Evolutionary Computation* 24, 1 (2019), 1–15.
- Danial Yazdani, Mohammad Nabi Omidvar, Ran Cheng, Juergen Branke, Trung Thanh Nguyen, and Xin Yao. 2020b. Benchmarking Continuous Dynamic Optimization: Survey and Generalized Test Suite. *IEEE Transactions on Cybernetics* (2020), 1 – 14.
- Danial Yazdani, Mohammad Nabi Omidvar, Donya Yazdani, Jürgen Branke, Trung Thanh Nguyen, Amir H Gandomi, Yaochu Jin, and Xin Yao. 2024b. Robust Optimization Over Time: A Critical Review. *IEEE Transactions on Evolutionary Computation* 28, 5 (2024), 1265–1285.
- Delaram Yazdani, Danial Yazdani, Eduardo Blanco-Davis, and Trung Thanh Nguyen. 2024c. A survey of multi-population optimization algorithms for tracking the moving optimum in dynamic environments. *Journal of Membrane Computing* (2024), 1–23.
- Danial Yazdani, Donya Yazdani, Jürgen Branke, Mohammad Nabi Omidvar, Amir Hossein Gandomi, and Xin Yao. 2022. Robust optimization over time by estimating robustness of promising regions. *IEEE Transactions on Evolutionary Computation* 27, 3 (2022), 657–670.
- Delaram Yazdani, Danial Yazdani, Donya Yazdani, Mohammad Nabi Omidvar, Amir H. Gandomi, and Xin Yao. 2023. A Species-Based Particle Swarm Optimization with Adaptive Population Size and Deactivation of Species for Dynamic Optimization Problems. *ACM Transactions on Evolutionary Learning and Optimization* 3, 4 (2023), 1–25.