Yazdani, D, Omidvar, MN, Cheng, R, Branke, J, Nguyen, TT and Yao, X

Benchmarking Continuous Dynamic Optimization: Survey and Generalized Test Suite

http://researchonline.ljmu.ac.uk/id/eprint/13473/

Article

Benchmarking Continuous Dynamic Optimization: Survey and Generalized Test Suite

Danial Yazdani, Mohammad Nabi Omidvar, Ran Cheng, Jürgen Branke, Trung Thanh Nguyen, and Xin Yao

***Abstract*—Dynamic changes are an important and inescapable aspect of many real-world optimization problems. Designing algorithms to find and track desirable solutions while facing challenges of dynamic optimization problems is an active research topic in the field of swarm and evolutionary computation. To evaluate and compare the performance of algorithms, it is imperative to use a suitable benchmark that generates problem instances with different controllable characteristics. In this paper, we give a comprehensive review of existing benchmarks and investigate their shortcomings in capturing different problem features. We then propose a highly configurable benchmark suite, the generalized moving peaks benchmark, capable of generating problem instances whose components have a variety of properties such as different levels of ill-conditioning, variable interactions, shape, and complexity. Moreover, components generated by the proposed benchmark can be highly dynamic with respect to the gradients, heights, optimum locations, condition numbers, shapes, complexities, and variable interactions. Finally, several well-known optimizers and dynamic optimization algorithms are chosen to solve generated problems by the proposed benchmark. The experimental results show the poor performance of the existing methods in facing new challenges posed by the addition of new properties.**

***Index Terms*—Dynamic environments, Dynamic optimization problems, Evolutionary dynamic optimization, Moving peaks benchmark, Tracking moving optimum, Survey.**

I. Introduction

SEARCH spaces of many real-world optimization problems are dynamic in terms of the objective function, the number of variables, and/or constraints [1]. Optimization problems that change over time and need to be solved online by an optimization method are referred to as dynamic optimization problems (DOPs) [2]. To solve DOPs, algorithms not only need to find desirable solutions but also to react to the environmental changes in order to quickly find a new solution when the previous one becomes suboptimal [3].

D. Yazdani, R. Cheng and X. Yao are with Shenzhen Key Laboratory of Computational Intelligence, University Key Laboratory of Evolving Intelligent Systems of Guangdong Province, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mails: danial.yazdani@gmail.com, chengr@sustc.edu.cn, xiny@sustc.edu.cn). Xin Yao is also with the CERCIA, School of Computer Science, Birmingham B15 2TT, UK.

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

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

T. T. Nguyen is with the Department of Maritime and Mechanical Engineering, Liverpool John Moores University, Liverpool L3 3AF, United Kingdom (e-mail: T.T.Nguyen@ljmu.ac.uk).

Corresponding author: Ran Cheng.

To comprehensively evaluate the effectiveness of algorithms designed for DOPs, a suitable benchmark generator is crucial. A DOP benchmark generator should have the following major features [1], [2]:

- Easy to implement and analyze; Researchers should have access to information about the location of the global optimum and its fitness value, and characteristics of components that form the landscape, such as their variable interaction structure, shape, and change intensities. Although this information should not be used for designing/tuning algorithms and the problem must remain a black-box for algorithms, it allows researchers to analyze and measure the behavior and performance of the algorithms. Additionally, a good benchmark should not be excessively difficult to implement.
- Flexibility: the benchmark generator should be highly configurable with respect to different aspects such as the number of components, shape of components, dimension, change frequency, and change severity. In addition, the configuration of any feature should be independent of other features. This independence allows the researchers to investigate the effect of each single property on the behavior of algorithms. Moreover, a good benchmark should be capable of generating problem instances with varying degrees of complexity. Many real-world problems are very hard and complex [1], [2], [4].
- Variety: the benchmark generator should be capable of generating problems instances with a variety of characteristics such as modularity (fully separable, fully non-separable, and partially separable), components with different condition numbers (ill-conditioning), different intensity of local and global modality (unimodal to highly multimodal), heterogeneity, balanced to highly imbalanced subfunctions, low to high-dimensional (large-scale), different levels of irregularities (smooth to highly irregular), and symmetric components to highly asymmetric ones.

Several DOP benchmarks have been proposed in the literature for different DOP domains [2], [5] such as combinatorial [6], [7], continuous [4], [8], multi-objective [9]–[11], and constrained DOPs [12]–[14]. This paper focuses on dynamic continuous unconstrained single-objective optimization problems. For brevity, we use the term DOP to refer to the considered DOPs in this paper. Commonly used and well-known benchmark generators in this domain use the idea of having several components that form the landscape. In most existing DOP benchmarks, the width, height, and location of these components change over time [5]. One of the benchmark generators that is designed based on this idea is the Moving Peaks Benchmark (MPB) [8], which is the most widely used synthetic problem in the DOP field [1], [3], [5]. Each component in MPB is formed by usually a simple peak whose basin of attraction is determined using a $\max(\cdot)$ function. MPB and its variations have been used in different classes of DOPs such as tracking moving optimum (TMO) in which the algorithms search for the best solution in each environment [2], robust optimization over time (ROOT) in which the algorithms search for solutions whose quality remain acceptable after environmental changes [15], [16], multi-objective DOPs in which there are multiple conflicting objectives [17], large-scale DOPs, where algorithms faces the scalability issues due to high-dimensionality of the problem [18], [19], and DOPs with dynamic constraints in which there are several moving feasible regions in the landscape [13].

Despite the popularity of the standard MPB, landscapes generated by MPB consist of components that are smooth, symmetric, unimodal, separable [18], and easy to optimize, which may not be the case in many real-world problems. Note that MPB is capable of using basic functions such as Rastrigin, Griewank, Ackley, and Rosenbrock as its components instead of the simple peaks. However, practically it is very challenging to use the aforementioned basic functions due to their different characteristics such as shape, gradient, the ratio between the fitness value drop to the distance to the optimum, and standard search range. In fact, many basic functions can not be used as the components of MPB, and for each of the ones that can be utilized, all the MPB parameters need to be specifically tuned. On the other hand, using different basic functions in the MPB to have components with different characteristics is more challenging and almost impossible in many cases.

In this paper, we propose a *generalized* MPB (GMPB) which is capable of generating problems with a variety of characteristics that can range from fully non-separable to fully separable structure, from homogeneous to highly heterogeneous sub-functions, from balanced to highly imbalanced sub-functions, from unimodal to highly multimodal components, from symmetric to highly asymmetric components, and from smooth to highly irregular components. Therefore, GMPB is a benchmark generator with fully controllable features that helps researchers to analyze DOP algorithms and investigate their efficiencies in facing a variety of different problem characteristics. Although the aforementioned problem characteristics have been vastly investigated in other fields of evolutionary computations such as static optimization problems [20]–[22], and large-scale optimization problems [23], [24], little related work has been dedicated to the DOP literature.

The major contributions of this paper can be summarized as follows:

- A comprehensive survey on continuous unconstrained single-objective DOP benchmarks that investigates their baselines, dynamics, characteristics, and behaviors. This survey helps the reader to understand the shortcomings of the existing benchmarks.
- A new benchmark with the capability of generating a variety of problem characteristics that have not been considered in the existing DOP benchmarks. All the characteristics are mathematically formulated into a baseline function that can be easily configured and controlled through its parameters. GMPB is capable of generating problems with any desired combination of characteristics that can also change over time.
- A detailed analysis of the behavior of several well-known algorithms on different problem characteristics.

Background information, including the definitions of dynamic optimization problems, variable interaction, modularity, imbalance, heterogeneity, and ill-conditioning, are provided in Section S-I of the supplementary document. The organization of the rest of this paper is as follows. Section II provides a survey of continuous unconstrained single-objective DOP benchmarks. Section III proposes the generalized MPB. Section S-II conducts a comprehensive empirical analysis of the GMPB. Finally, Section IV concludes the paper and outlines possible future directions.

II. A SURVEY ON DOP BENCHMARKS

In this section, we review the commonly used and well-known DOP benchmarks, which are continuous, single-objective, and unconstrained. In this paper, DOP benchmarks are categorized as follows:

1) DOP benchmarks whose landscapes are formed by a single component; and
2) DOP benchmarks whose landscapes are made by jointing several components. The benchmarks belonging to this category handle multiple components in two major ways:
   a) Using the $\max(\cdot)$ function to control the basin of attraction among multiple components.
   b) Partitioning the search space into hypercube based components.

A. DOP benchmarks whose landscape is formed by a single component

The AB benchmark [25] is the first in this category, which uses two 2-dimensional landscapes called A and B, each with different characteristics. The A landscape consists of 14 sinusoidally shaped peaks, and B consists of 20 peaks, whose shapes are determined by a sine function, and triangular and Gaussian probability distributions. Three types of environmental changes are defined as follows: 1) linear translation of peaks in A; 2) changing the location of the maximum peak randomly while the rest of the search space remains unchanged; and 3) switching between landscapes A and B.

Moving parabola (MP) [26] applies linear, circular, and random dynamics to a simple 3-dimensional sphere function to shift its location. Given a severity parameter $\tilde{\phi}$, the aforementioned three dynamics are formulated as follows:

$$\phi_i^{(t+1)} = \phi_i^{(t)} + \tilde{\phi}, \tag{1}$$

$$\phi_i^{(t+1)} = \begin{cases} \phi_i^{(t)} + \tilde{\phi}\sin\dfrac{2\pi t}{25} & i = 1,3 \\ \phi_i^{(t)} + \tilde{\phi}\cos\dfrac{2\pi t}{25} & i = 2, \end{cases} \tag{2}$$

$$\phi_i^{(t+1)} = \phi_i^{(t)} + \tilde{\phi}\mathcal{N}(0,1), \tag{3}$$

where $\phi_i^{(t)}$ is the offset of the ith dimension in tth environment for $i \in \{1,2,3\}$, $\mathcal{N}(0,1)$ is a random number drawn from a Gaussian distribution with mean 0 and variance 1, and $\tilde{\phi}$ is the change severity.

Dynamic rotation (DR) [27] combines the landscape with a visibility mask, which only allows the fitness of certain regions of the landscape to be evaluated. The visibility mask forces the remaining regions to return a predefined undesirable fitness value. DR uses five types of dynamics: 1) rotating the objective function around the optimum where there is no visibility mask; 2) rotating the visibility mask around the optimum where the objective function is stationary; 3) rotating the objective function around its optimum while the visibility mask is static; 4) rotating both the objective function and the visibility mask around the optimum with the same rotation angle; and 5) rotating both the objective function and the visibility mask around the optimum with different rotation angles.

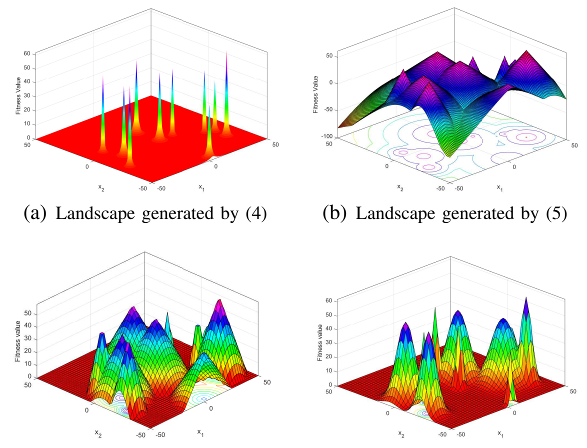
Composition dynamic benchmark generator (CDBG) [28] is extended from the static composition functions in [21], [22]. Since the landscape of CDBG is made by composing several single-component subfunctions, we cannot categorize it as a benchmark with joining components. Moreover, since it uses some standard static basic functions such as Ackley and Rastrigin in its structure, the number of peaks cannot be controlled by the user. CDBG uses the environmental dynamics of the generalized dynamic benchmark generator (GDBG) [28] for shifting each basic function, which will be discussed later in this section.

A general framework for inducing artificial changes (IAC) in optimization problems is proposed in [29]. The utilized landscapes in IAC contain some basic functions such as Rosenbrock, Griewank, Ackley, and Rastrigin. IAC contains four groups of dynamics that can be applied to the landscapes to generate environmental changes. These dynamics include: 1) rotation; 2) shift; 3) decision variables permutation; and 4) duplication based dynamics where a predefined number of decision variables are randomly copied to other positions. The aforementioned dynamics can be applied to the entire landscape, or to some regions of the search space. Consequently, IAC can generate both local or global environmental changes.

B. DOP benchmarks whose landscape is formed by joining several components

This group of DOP benchmarks is further categorized into the ones whose landscape is divided using the $\max(\cdot)$ function or partitioning by hypercubes.

1) DOP benchmarks based on the $\max(\cdot)$ *function:* This group of DOP benchmarks uses a $\max(\cdot)$ function to define the basin of attraction of components, which are usually single peaks. Branke's moving peaks benchmark (MPB) [8] is the most popular benchmark suite in DOPs [1], [5]. MPB generates a landscape containing several components in which each component contains a peak whose height, width, and location change every time the environment changes. MPB is flexible and can be used to generate scalable functions with a



(a) Landscape generated by (4) (b) Landscape generated by (5)

(c) Landscape generated by (10) (d) Landscape generated by (12)

Fig. 1. Landscapes with 10 components generated by different baselines. All shared parameter settings, such as component centers' locations, heights, and widths, are equal. In Figure 1(c), β is set to zero. Note that in contrary to the flat regions in Figure 1(c) whose gradient is zero, the gradients in the flat looking regions in Figures 1(a) and 1(d) have positive non-zero values.

configurable number of components. Each component has the potential to become the global optimum after an environmental change. In the first version of MPB [8], the baseline function was defined as follows:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \frac{h_i^{(t)}}{1 + w_i^{(t)} \sum_{j=1}^{d} (\mathbf{x}_j - \mathbf{c}_{i,j}^{(t)})^2}, \tag{4}$$

where m is the number of components (peaks), $\mathbf{x}$ is a solution in the d-dimensional problem space, $h_i^{(t)}$, $w_i^{(t)}$, and $\mathbf{c}_i^{(t)}$ are the height, width, and the center of the ith peak in the tth environment, respectively. Later, a new scenario of MPB was proposed (Scenario 2), which replaces the *peak* function in (4) with the *cone* function [30] (as in the DF1 benchmark generator [31], [32]). Scenario 2 has become the standard configuration of MPB in many studies [33]–[41].

The Scenario 2 baseline function of MPB is as follows:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \mathbf{x} - \mathbf{c}_i^{(t)} \right\| \right\}. \tag{5}$$

Figures 1(a) and 1(b) illustrate MPB landscapes generated by (4) and (5), respectively. In MPB, the height, width, and center of all components change from one environment to the next based on the following update rules:

$$h_i^{(t+1)} = h_i^{(t)} + \tilde{h}\mathcal{N}(0,1), \tag{6}$$

$$w_i^{(t+1)} = w_i^{(t)} + \tilde{w}\mathcal{N}(0,1), \tag{7}$$

$$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \tag{8}$$

$$\mathbf{v}_i^{(t+1)} = \tilde{s} \cdot \frac{(1-\lambda)\cdot \mathbf{u} + \lambda \cdot \mathbf{v}_i^{(t)}}{\left\| (1-\lambda)\cdot \mathbf{u} + \lambda \cdot \mathbf{v}_i^{(t)} \right\|}, \tag{9}$$

where $\mathcal{N}(0,1)$ is a random number drawn from a Gaussian distribution with mean 0 and variance 1, $\tilde{h}$ is the height severity, $\tilde{w}$ is the width severity, $\tilde{s}$ is the shift severity, $\mathbf{u}$ is a vector of uniformly distributed numbers in range $[-0.5, 0.5]$, and $\lambda \in (0,1)$ is the correlation coefficient. $\lambda = 0$ implies that the peak relocations are uncorrelated whereas they are

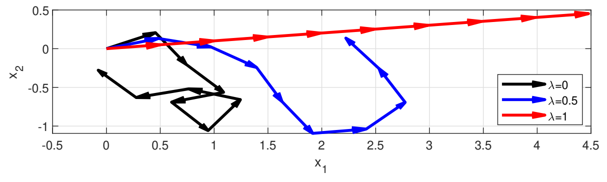


Fig. 2. The movement of the peak center through a 2-dimensional subspace over 10 environmental changes where $\tilde{s} = 1$. The initial position of peak is [0,0], and the same random seed is used for each sequence of movements.

fully correlated for $\lambda = 1$. The step size of peak movements is constant and equal to $\tilde{s}$. Figure 2 shows 10 relocations of a peak center using different values of λ where $\tilde{s} = 1$. According to (6) and (7), the default environmental dynamic for a peak's height and width in MPB is *random change* [26] based on (3). At each environmental change, all components' parameters change using (6), (7), and (8). Consequently, the entire search space changes after each environmental change. Considering this characteristic, an environmental change can be detected by re-evaluating any solution across the search space by comparing its current and previous fitness values. If the values are different, an environmental change can be detected.

In [42], a control parameter is added to MPB to determine the percentage of changing components at each environmental change. As a result, a predefined number of randomly chosen components will change using (6), (7), and (8), while the remaining ones will remain unchanged. Therefore, change detection becomes more challenging for some methods as only some parts of the environment change [40]. Furthermore, in order to increase the size of the regions that remains unchanged after environmental changes, a threshold β is used:

$$f^{(t)}(\mathbf{x}) = \max\left\{\beta, \max_{i\in\{1,\dots,m\}}\left\{h_i^{(t)} - w_i^{(t)}\left\|\mathbf{x} - \mathbf{c}_i^{(t)}\right\|\right\}\right\}, \tag{10}$$

where β determines the minimum fitness value of the problem and creates flat regions. The fitness value of any solution in these flat regions is equal to β. These flat regions make the problem more challenging since the change detection has become harder. Figure 1(c) shows a generated landscape using (10).

In the field of robust optimization over time (ROOT) [15], the MPB with the baseline (5) is used in many studies [16], [43]–[47]. However, the values of $\tilde{h}$, $\tilde{w}$, and $\tilde{s}$ are different for each component. The reason for having different height, width and shift severities for each component is to create components with different levels of robustness. Equations (3), (14), and (18) are used as dynamics in [47] while default MPB dynamics, i.e. (3) are used in [16], [43]–[46].

The MPB has one global optimum in each environment. However, most multi-population methods [2] try to cover and track multiple optima. Note that, although multiple optima are covered by many DOP algorithms [2], most DOPs only have single global optimum, despite that there may also exist local optima. This is different from mutlmodal optimization where there exist multiple global optima inside the same fitness landscape [48], [49]. In many DOP algorithms, especially multi-population methods, covering multiple optima is done in order to increase and maintain overall diversity, and accelerate the process of finding new global optimum after environmental changes. In [50], a multimodal version of MPB with multiple global optima is proposed. In this benchmark, a predefined number of components are global optima whose heights are constant over time, and only their locations and widths are time varying. There are other local optima whose heights change over time. However, the upper bound of height value of the local optima is less than that of the global optima.

Pendulum MPB (PMPB) [51] is a modified version of MPB in which the environments reappear periodically over time. PMPB works based on a pendulum length parameter pl and a direction parameter dir. First, using the baseline (5) and the dynamics from (6), (7), and (8), pl environments $\left[f^{(1)}, f^{(2)}, \cdots, f^{(pl)}\right]$ are constructed. Then, the environmental parameters of the constructed environments are stored in an archive to be used for future reappearances. dir shows the direction of fetching the environmental parameters from the archive, to retrieve the next environment. The value of dir changes every pl environmental changes by switching between $right$ and $left$. The environments are retrieved from the archive based on their order and the value of dir. An example of the pendulum-based environmental changes is:

$$\left[f^{(1)}, f^{(2)}, \cdots, f^{(pl-1)}, f^{(pl)}, f^{(pl-1)}, \cdots, f^{(2)}, f^{(1)}, f^{(2)}, \cdots\right]. \tag{11}$$

Similar to MPB, DF1 [31], [32] generates problem instances in which the width, height, and location of components change over time. The baseline function of DF1 is similar to (5), however, the dynamics are different and a logistic function is used to generate dynamics. Another benchmark whose landscape consists of several components is the Gaussian peaks benchmark (GPB) [52] which uses the following baseline:

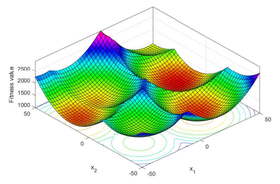
$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}}\left\{h_i^{(t)}\exp\left(-\frac{\left\|\mathbf{x} - \mathbf{c}_i^{(t)}\right\|^2}{2\left(w_i^{(t)}\right)^2}\right)\right\}. \tag{12}$$

An example of a generated 2-dimensional landscape by this benchmark is shown in Figure 1(d). In GPB, the locations of peaks change in random directions, and the step sizes are uniformly distributed over an interval controlled by two levels of severity called abrupt and gradual.

In contrast to the aforementioned landscapes, Blackwell [53] proposes a minimization benchmark generator denoted as moving valleys benchmark (MVB). The generated landscapes by MVB are constructed by several spheric components whose basins of attractions are determined by a $\min(\cdot)$ function. MVB is constructed using the following baseline function:

$$f^{(t)}(\mathbf{x}) = \min_{i\in\{1,\dots,m\}}\left\{\sum_{j=1}^{d}\left(\mathbf{x}_j - \mathbf{c}_{i,j}^{(t)}\right)^2 + \left(h_i^{(t)}\right)^2\right\}. \tag{13}$$

As can be seen in (13), MVB does not use width parameter w which leads to construct components with identical gradients. Figure 3(a) illustrates a 2-dimensional landscape generated by this benchmark.


(a) A landscape generated by (13) with $m = 10$ and $m' = 6$.

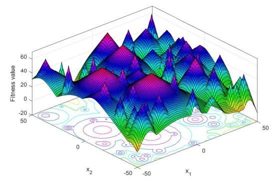
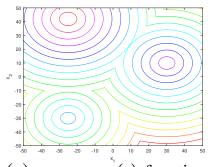

(b) A landscape generated by (5) with $m = 100$ and $m' = 68$.

Fig. 3. Two examples in which some components are covered by larger ones.


(a) $\max(\cdot)$-function based basin of attraction.

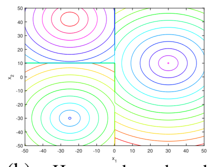

(b) Hyper-cube based basin of attraction.

Fig. 4. Two landscapes with 3 peaks (components) whose basin of attraction is determined based on their hypercube, or $\max(\cdot)$ function. The peak function is $s_1 = h - \|\mathbf{x} - \mathbf{c}\|$ from [4] which is a *cone* function from (5) with $w = 1$.

One main property of all aforementioned benchmarks in this part is the dynamic number of visible components m' in the landscape. A component is visible if it is not covered by any other larger component. When components are wider, the possibility of decreasing m' increases. Among the investigated benchmarks, MVB has very broad components, such that it is probable that the smaller components become hidden under larger ones. For example, as can be seen in Figure 3(a), the number of components m is set to ten while the number of visible components m' is smaller. Furthermore, the number of visible components drops by increasing m. In such circumstances, the density of components is increased such that larger components can cover several smaller close components. Figure 3(b) shows a generated landscape by (5) where the number of components m is set to 100. However, almost a third of them are invisible. Changing the number of visible components is a challenging property for algorithms that try to cover and track multiple moving components. This is due to the fact that the algorithm will lose its track when a component becomes hidden under a larger component in an environmental change. In addition to the property of the varying number of visible component m', a modified version of MPB is proposed in [54] where the number of components m is changing over time.

The generalized dynamic benchmark generator (GDBG) [28] contains six different types of dynamics including small step, large step, random, chaotic, recurrent, and recurrent with noise which are formulated respectively as follows:

$$\Delta\phi = \gamma r \tilde{\phi} \|\phi\|, \tag{14}$$

$$\Delta\phi = \tilde{\phi} \|\phi\| (\gamma \mathrm{sign}(r) + r(\gamma_{\max} - \gamma)), \tag{15}$$

$$\Delta\phi = \tilde{\phi} \mathcal{N}(0,1), \tag{16}$$

$$\phi^{(t+1)} = A\phi^{(t)} \frac{1-\phi^{(t)}}{\|\phi\|}, \tag{17}$$

$$\phi^{(t+1)} = \phi_{\min} + \frac{\|\phi\|}{2}(\sin(\frac{2\pi}{p}t + \varphi) + 1), \tag{18}$$

$$\phi^{(t+1)} = \phi_{\min} + \frac{\|\phi\|}{2}(\sin(\frac{2\pi}{p}t + \varphi) + 1) + \tilde{n}\mathcal{N}(0,1), \tag{19}$$

where $\Delta\phi$ is a deviation from the current control parameters, $\|\phi\|$ is the change range of ϕ, $\phi^{(t)}$ is the offset in tth environment, $\tilde{\phi} \in (0,1)$ is change severity of ϕ, $\phi_{\min}$ is the minimum value of ϕ, $\tilde{n} \in (0,1)$ is noise severity, $\gamma, \gamma_{\max} \in (0,1)$ and A are constant values, $r \in (-1,1)$ is a random number drawn from a uniform distribution, p is the period number, and φ is the initial phase. It is worth mentioning that the random change in (16) is the same as the dynamic that was used in MPB in (6) and (7), and in moving parabola in (3). Moreover, the utilized logistic function for chaotic change in (17) is similar to the logistic function, which is used in DF1. In [55], [56], other dynamics are added to GDBG including change in the number of components and dimensions. Two further DOP benchmarks use the GDBG dynamics, including CDBG which was described in Section II-A, and Rotation DBG (RDBG) [28]. RDBG is constructed based on the baseline of MPB in (4). The height and width of components change using a dynamic from GDBG. Furthermore, the location of components changes using rotation matrices introduced in DR [27]. Since the relocations of components by the rotation procedure are not controllable in RDBG, it is not capable of constructing cyclic environments. In [57], the procedure of rotation in RDGB is modified in order to control the components' relocation steps. This benchmark is capable of constructing cyclic environments with a predefined cycle length.

Although most DOP benchmarks are scalable, they cannot produce modular problems whose components contain multiple moving optima. To address this shortcoming, some studies used composition methods to create modular problems. High-dimensional MPB (HDMPB) [19] is the first modular DOP benchmark with multiple component subfunctions, which is built by the summation of several MPBs to create large-scale DOPs. CMPB [3], [18] is another benchmark that composes several MPBs to create modular problem instances. Unlike HDMPB, CMPB can generate partially separable problems that could contain fully separable dimensions. In addition, CMPB is capable of generating modular problems that are imbalanced and heterogeneous.

2) DOP benchmarks based on hypercube partitioning: The first DOP benchmark of this category is dividing search space (DSS) [58] that partitions the search space into subspaces. Each subspace contains a simple peak function whose summit is at the center of the subspace. For dividing the d-dimensional search space, each dimension is equally divided into k segments resulting in k^d subspaces.

Another DOP benchmark in which the idea of space partitioning by hypercubes is Free Peaks (FPs) [4]. FPs is a complicated benchmark generator whose search space is divided into components using a k-d tree [59]. In each component, there is only one component that can move inside the hypercube. In FPs, the basin of attraction of components are determined by the hypercubes, which is different from other benchmarks such as MPB, DF1 and GPB. An illustrative example is provided

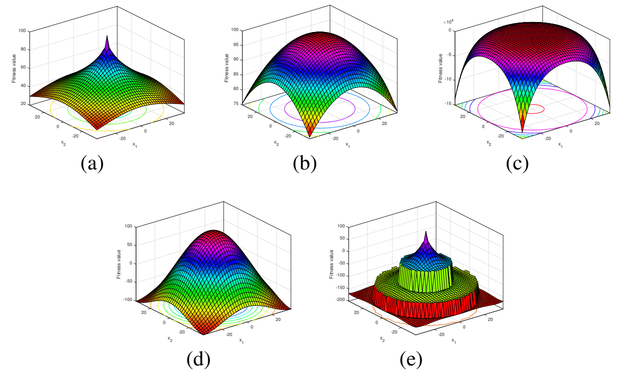


Fig. 5. Five types of components that are used in FPs [4].

in Figure 4 to show the difference between the two types of the basin of attractions. As can be seen in Figure 4(b), the landscape generated by hypercube partitioning has sharp ridges around components. The shape of each component in FPs is chosen from eight different *unimodal* basic functions. From these eight basic functions, three of them are similar to the components (with $w = 1$) from (5), (12), and (4). The shape of the remaining five basic functions of FPs are shown in Figure 5.

To change the shape of each component, a random shape function is chosen from the eight basic functions for each environment. In FPs, contrary to MPB, DF1, and GPB, components do not have a width parameter, and their gradients are determined based on their basic functions. Additionally, the sizes and locations of hypercubes change over time, which leads to changing the size of components. Moreover, component locations inside their respective hypercubes and their height change over time using a random dynamic step. In benchmarks such as FPs whose basin of attractions of components is determined by dividing the landscape into hypercubes, there is no invisible component ($m = m'$). In FPs, a dynamic for changing the number of components/hypercubes is used to cover problems in which the number of components changes over time.

Several transformations are used in FPs. An important and related to single objective DOP transformation of FPs is 'setup of dependencies', which determines the variable interactions by redefining the distance in the eight basic functions. However, the study did not provide any analysis to show the variable interactions when the Euclidean distance (default distance) is used in comparison to the redefined version. According to our analysis using DG2 [60], although it is claimed that the redefinition of distance transformation can determine variable interactions and produce partially separable components, it only can generate fully separable and fully non-separable cases. However, it should be mentioned that this can potentially change the intensity of variable interactions and thus affects the hardness of the problem for optimizers. Furthermore, redefinition of distance transformation can only affect the complexity of exploring components inside their hypercubes and it is not capable of changing the separability of the search space. In fact, the generated landscapes by FPs are

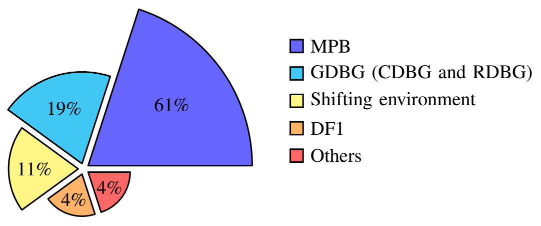


Fig. 6. Distribution of the DOP benchmarks utilization since 1999.

fully non-separable when there is more than one component. Another transformation used in FPs is 'setup of domino convergence' in which the contribution of each dimension will be different from other dimensions by redefining distance in the eight basic functions. Moreover, two transformations from [20] are used in FPs to add smooth irregularities (without changing modality condition) and symmetry breaking.

C. Popularity of DOP benchmarks

In this part, we investigate the popularity of DOP benchmarks in the literature. The search has been conducted using different engines, including Scopus, Google Scholar, ScienceDirect, IEEE Xplore, ACM Digital Library, and Springer. As the searching keywords, we used different terms such as dynamic optimization problems, dynamic environments, evolutionary dynamic optimization, and uncertain environments. Then, we have selected references in which DOP algorithms/frameworks were tested on the unconstrained single-objective continuous DOP benchmarks since 1999. This process was done in August 2019, and 176 journal papers, book chapters, and conference papers have been selected for our investigation.

Figure 6 illustrates the distribution of the DOP benchmarks utilization. In this part, we have categorized DOP benchmarks into five groups; 1) MPB and its modified versions, 2) GDBG [28], [55], [56] which contains all versions of CDBG and RDBG, 3) DF1, 4) shifting environments in which shift functions from MP [26] are utilized for moving static functions such as Sphere, Rastrigin, Griewank, Ackley and Rosenbrock, and 5) other DOP benchmarks. If, in a reference, more than one of the above groups of benchmarks is used, we have considered them in the counter of each used group. As can be seen in Figure 6, MPB and its modified versions are the most commonly used benchmark in continuous DOP.

D. Discussion

Table I summarizes the characteristics of problems and components that are generated by each reviewed DOP benchmark in this section alongside the proposed GMPB. As can be seen, although all the previous DOP benchmarks can generate some problem characteristics, we cannot find a benchmark generator exhibiting different combinations of problem characteristics. One important limitation of the previous benchmarks is that each of their components consists of a single peak, which is easy to optimize because they are smooth, symmetric, and unimodal, which may not be the case of many real-world problems.

TABLE I
CHARACTERISTICS OF THE REVIEWED DOP BENCHMARKS IN SECTION II AND THE PROPOSED GMPB.

	Characteristic	DOP benchmark														
		AB[b]	MP	DR[c]	DSS	MPB	DF1	GPB	MVB	RDBG	CDBG[c]	IAC[c]	HDMPB	CMPB	FPs	GMPB
Landscape properties	Controllable number of components	✗	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
	Modularity	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	✓
	Heterogeneity	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✓
	Imbalance	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✓
Component properties	Separable[a]	-	✓	-	✓	✓	✓	✓	✓	✓	-	-	✓	✓	✓	✓
	Non-separable[a]	-	✗	-	✗	✗	✗	✗	✗	✗	-	-	✗	✗	✓	✓
	Ill-conditioning	-	✗	-	✗	✗	✗	✗	✗	✗	-	-	✗	✗	✓	✓
	Smooth	-	✓	-	✓	✓	✓	✓	✓	✓	-	-	✓	✓	✓	✓
	Irregular	-	✗	-	✗	✗	✗	✗	✗	✗	-	-	✗	✗	✓	✓
	Unimodal	-	✓	-	✓	✓	✓	✓	✓	✓	-	-	✓	✓	✓	✓
	Multimodal	-	✗	-	✗	✗	✗	✗	✗	✗	-	-	✗	✗	✗	✓
	Symmetric	-	✓	-	✓	✓	✓	✓	✓	✓	-	-	✓	✓	✓	✓
	Asymmetric	-	✗	-	✗	✗	✗	✗	✗	✗	-	-	✗	✗	✓	✓
	Basin of attraction[d]	-	-	-	HC	MF	MF	MF	MF[e]	MF	-	-	MF	MF	HC	MF
Change	Location of components	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Size of components	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
	Shapes of components	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓
	Rotation	✗	✗	✓	✗	✗	✗	✗	✗	✗[f]	✓	✓	✗	✗	✗	✓
	Number of visible optima	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
	Condition number of components	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Complexity of components	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓

[a] Determined according to empirical analysis on a dimension-wise manner.
[b] Component formula has not been provided in [25].
[c] DR, CDBG, and IAC do not use distance-based component functions to produce multiple components on the landscape.
[d] Hypercube (HC) and $\max(\cdot)$ function (MF) based basin of attractions.
[e] In MVB, a $\min(\cdot)$ function is used to determine the basin of attractions of components.
[f] The rotation is used to relocate components' centers.

According to Table I, among all previous DOP benchmarks, the state-of-the-art FPs is capable of generating components with a variety of problem properties by using some transformations. However, it still has some considerable limitations. FPs is not easy to understand and hard to implement. Additionally, some of its transformations cannot be used simultaneously (such as setup of dependencies and domino convergence). Moreover, ill-conditioning, irregularities, and symmetry breaking transformations are static. Furthermore, components generated in FPs are unimodal because the transformations do not change local modality. Finally, FPs cannot produce modular, heterogeneous, and imbalanced problems. In terms of modularity, CMPB [18] is the only benchmark capable of generating modular, heterogeneous, and imbalanced problems. In CMPB, an automated weight is assigned to each subfunction to avoid the situations in which partially separable subfunctions become dominated by fully-separable dimensions in terms of contribution to overall fitness value. However, this circumstance can happen when there are several low dimensional (such as 2 and 3-dimensional) subfunctions. Therefore, CMPB is not suitable to generate partially separable problem instances whose number of low-dimensional subfunctions are more than the ones with higher dimensions. In the next section, GMPB is introduced, which addresses all the above issues.

III. GENERALIZED MOVING PEAKS BENCHMARK

In this section, the details of the GMPB are given. We start with the baseline of the traditional MPB (5) and a series of modifications are proposed in order to incorporate the following features into the new design:

1) Modularity: Most DOP benchmarks whose generated landscape contains multiple components (such as MPB, FPs, and DF1 whose components are single local peaks) are non-separable in nature [18]; therefore, they are not capable of generating modular test instances. This limitation motivates the design of a new benchmark capable of producing modular test instances with heterogeneity and imbalance characteristics. In GMPB, subfunctions can vary in size, shape, change intensity, and contribution to the overall fitness value.

2) Component Complexity: Although most DOP benchmarks have a multimodal landscape, each component is unimodal, smooth, regular, separable, and symmetric, which makes them easy to optimize. GMPB is capable of generating a wide range of easy to hard to optimize components by adding the following properties to each component with varying degree of intensity:

a) Ill-conditioning: Ill-conditioning is an important property of many real-world problems [20], [24], which cannot be found in the components generated by most DOP benchmarks due to the circular nature of the level curves of each peak. This motivates the design of a new benchmark capable of generating ill-conditioned peaks whose condition number *changes over time*.

b) Asymmetry: In almost all DOP benchmarks, components are symmetric which is undesirable for the following reasons: 1) symmetry of the peaks may be in favor of some special operators that use Gaussian distributions to generate new solutions [20], [24]; and 2) they are easy to optimize on a dimension-wise basis due to their symmetric nature. The GMPB is capable of generating symmetric to highly asymmetric components.

c) Component separability: In addition, as shown analytically [18], an n-dimensional peak generated by MPB and DF1 by (5) is separable which makes it easier to optimize. As demonstrated in Table I, most other DOP benchmarks generate components (peaks) that can be optimized in a dimension-wise manner. These empirical observations suggest that they are separable. Rotating a component is a way of changing its variable interaction and making it *fully non-separable*, which is more difficult to optimize [20], [24], [61]. In GMPB, components can be rotated using rotation matrices to have different degrees of variable interactions. Using rotation matrices to transform problem spaces has been used in many benchmarks for static [20], [22], [24], [62] and dynamic [4], [27], [28], [56], [63] optimization problems. Moreover, by rotating a component over time, the degree of variable interactions changes dynamically. In the rest of this section, the procedure of transforming MPB into GMPB is described.

d) Component modality: As mentioned in Table I, generated components of all existing benchmarks are unimodal which makes them easy to optimize. Components in GMPB can be unimodal to highly multimodal.

A. Components with varying condition number

According to (5), the width of each component (peak) is the same in all dimensions, which makes the shape of components cone-like with circular contour lines (see Figure 7(a)). To alleviate this, the width of a component is changed from a scalar variable to a d-dimensional vector. We know that the Euclidean distance can be shown as square root of a dot product, i.e., $\|\mathbf{a}\| = \sqrt{\mathbf{a}^T\mathbf{a}}$. Therefore, MPB can be rewritten as follows:

$$\begin{aligned} f^{(t)}(\mathbf{x}) &= \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \mathbf{x} - \mathbf{c}_i^{(t)} \right\| \right\} \\ &= \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - \sqrt{\hat{w}_i^{(t)} \left(\mathbf{x} - \mathbf{c}_i^{(t)}\right)^\top \left(\mathbf{x} - \mathbf{c}_i^{(t)}\right)} \right\}, \end{aligned} \tag{20}$$

where $\sqrt{\hat{w}_i^{(t)}} = w_i^{(t)}$. Instead of factoring the width we can represent it as a diagonal matrix $\mathbf{W}_i^{(t)} = \hat{w}_i^{(t)}\mathbf{I}$:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - \sqrt{\left(\mathbf{x} - \mathbf{c}_i^{(t)}\right)^\top \mathbf{W}_i^{(t)} \left(\mathbf{x} - \mathbf{c}_i^{(t)}\right)} \right\}, \tag{21}$$

By allowing the main diagonal elements of $\mathbf{W}$ to be set arbitrarily, the contour lines of the function become ellipses which are aligned with the coordinate axes. Figure 7(b) shows an example of the MPB made using (21) with three components. One of the components (the one with circular contour) has the same width in each direction which makes it similar to the components made by (5) . The other two components have different width values for each dimension resulting in elliptical contour lines.

By calculating the condition number of the diagonal matrix $\mathbf{W}$, the ill-conditioning degree of each component can be determined. The condition number of a matrix is generally calculated through singular value decomposition of the matrix and finding the ratio between the largest and smallest diagonal elements of its singular value matrix. Since $\mathbf{W}$ is a diagonal matrix, its singular value matrix has the same diagonal values but in a sorted order; therefore, the condition number of $\mathbf{W}$ can be calculated without any singular decomposition. Since the diagonal values of $\mathbf{W}$ are squared values of the width vector, the condition number is equal to the squared condition number of the component. Consequently, the condition number of the generated hyper-ellipsoid by the (21) is the square root of the condition value of $\mathbf{W}$. If all the diagonal values of $\mathbf{W}$ are equal, then the problem is the same as the original MPB by (5) and the condition number of $\mathbf{W}$ will be 1. This condition number shows that the component does not have ill-conditioning. However, when the width values are different in different dimensions, the condition number of $\mathbf{W}$ grows. From a geometric perspective, the condition number of a hyper-ellipsoid is the ratio of its largest diameter to its smallest one. Figure 8 illustrates three components with different ill-conditioning degrees from the geometric point of view. As demonstrated in Figure 8, by increasing the difference between width values, the ratio between the largest and smallest diameters of the ellipsoid component grows. Consequently, the level of ill-conditioning is increased.

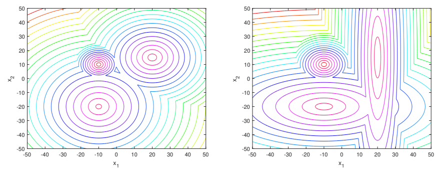

(a) Standard MPB generated by (5) (b) Ill-conditioned MPB generated by (21)

Fig. 7. Comparing two landscapes with and without ill-conditioning in components.

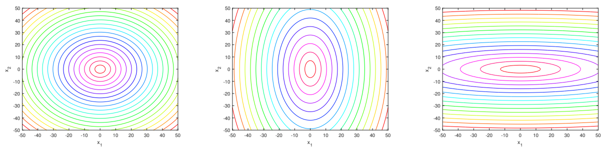
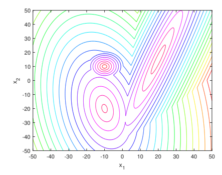

(a) w=[1,1] and $\mathfrak{c}$=1 (b) w=[2,4] and $\mathfrak{c}$=2 (c) w=[8,2] and $\mathfrak{c}$=4

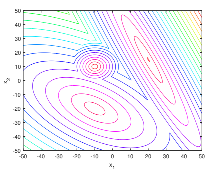
Fig. 8. Three different 2-dimensional components with different ill-conditioning degrees. The values of widths (w) of each peak and their corresponding condition number ($\mathfrak{c}$) are shown.

B. Components with varying variable interaction degree

As mentioned before, components of the MPB are circular peaks, separable, and easy to optimize. However, even after changing the MPB baseline to (21), peaks remain separable due to their symmetric nature. To change the principal axes of the elliptic contour lines, an orthonormal matrix $\mathbf{R}$ can be included as follows:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - \sqrt{\left(\mathbf{x} - \mathbf{c}_i^{(t)}\right)^\top {\mathbf{R}_i^{(t)}}^\top \mathbf{W}_i^{(t)} \mathbf{R}_i^{(t)} \left(\mathbf{x} - \mathbf{c}_i^{(t)}\right)} \right\}, \tag{22}$$


(a) Landscape generated by by Eq.(22)


(b) After rotating 45 degrees

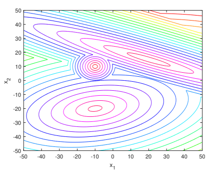

(c) After rotating 90 degrees

Fig. 9. Clockwise rotating a landscape generated by (27) in 2-dimensional space

where $\mathbf{R}_i^{(t)}$ is the rotation matrix of component i in tth environment. Equation (22) is capable of generating components with elliptic contour lines whose principal axes are defined by the rotation matrix $\mathbf{R}$. In (22), the ill-conditioning intensity of a component can be obtained by calculating the condition number of $\mathbf{R}^\top\mathbf{W}\mathbf{R}$ which is equal to the condition number of $\mathbf{W}$ because $\mathbf{R}$ is an orthonormal matrix and does not change the condition number of $\mathbf{W}$ after multiplication. Contrary to the components generated by (5) and (21), each generated component i by (22) is fully non-separable whose degree of variable interactions depends on $\mathbf{R}_i$. Figure 9(a) shows the landscape generated by (22) with three components (same component centroids as in Figure 7(b)).

The rotation matrix $\mathbf{R}_i$ is obtained by rotating the projection of $\mathbf{x}$ onto all x_p-x_q planes by a given angle θ. The total number of unique planes which will be rotated is $\binom{d}{2} = \frac{d(d-1)}{2}$. For rotating each x_p-x_q plane by a certain angle (θ), a *Givens rotation matrix* $\mathbf{G}_{(p,q)}$ must be constructed. To this end, first, $\mathbf{G}_{(p,q)}$ is initialized to an identity matrix $\mathbf{I}_{d\times d}$; then, four elements of $\mathbf{G}_{(p,q)}$ are altered as:

$$\mathbf{G}_{(p,q)}(p,p) = \cos\left(\theta^{(t)}\right), \tag{23}$$

$$\mathbf{G}_{(p,q)}(q,q) = \cos\left(\theta^{(t)}\right), \tag{24}$$

$$\mathbf{G}_{(p,q)}(p,q) = -\sin\left(\theta^{(t)}\right), \tag{25}$$

$$\mathbf{G}_{(p,q)}(q,p) = \sin\left(\theta^{(t)}\right), \tag{26}$$

where $\mathbf{G}_{(p,q)}(i,j)$ is the element at ith row and jth column of $\mathbf{G}_{(p,q)}$. $\mathbf{R}_i$ in tth environment is calculated by:

$$\mathbf{R}_i^{(t)} = \prod_{(p,q)\in\mathcal{P}} \mathbf{G}_{(p,q)} \times \mathbf{R}_i^{(t-1)}, \tag{27}$$

where $\mathcal{P}$ contains all unique pairs of dimensions defining all possible planes in a d-dimensional space. The order of the multiplications of the *Givens rotation matrices* is random. The reason behind using (27) for calculating $\mathbf{R}$ is that we aim to have control on the rotation matrix based on an angle severity $\tilde{\theta}$. Note that the initial $\mathbf{R}_i^{(0)}$ for problem instances with rotation property is obtained by using the Gram-Schmidt orthogonalization method on a matrix with normally distributed entries. Figure 9 shows an example of component rotation in the proposed benchmark in which by changing the angle of rotation, the degree of variable interactions changes.

C. Asymmetric components with Irregularities

Although optimizing a rotated component is harder for many algorithms due to its nonseparable nature [61], each component can still be considered *easy to optimize* since the components generated by (22) are smooth, regular, and unimodal. To add irregularity and local optima to the components, a transformation function is added to (22):

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}}\left\{h_i^{(t)} - \sqrt{\mathbb{T}\left(\left(\mathbf{x}-\mathbf{c}_i^{(t)}\right)^\top \mathbf{R}^{(t)\top}, i\right)\mathbf{W}^{(t)}\mathbb{T}\left(\mathbf{R}^{(t)}\left(\mathbf{x}-\mathbf{c}_i^{(t)}\right), i\right)}\right\}, \tag{28}$$

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

$$\mathbb{T}(y_j, i) = \begin{cases} \exp\left(\log(y_j) + \tau_i^{(t)}\left(\sin\left(\eta_{i,1}^{(t)}\log(y_j)\right) + \sin\left(\eta_{i,2}^{(t)}\log(y_j)\right)\right)\right) & \text{if } y_j > 0 \\ 0 & \text{if } y_j = 0 \\ -\exp\left(\log(|y_j|) + \tau_i^{(t)}\left(\sin\left(\eta_{i,3}^{(t)}\log(|y_j|)\right) + \sin\left(\eta_{i,4}^{(t)}\log(|y_j|)\right)\right)\right) & \text{if } y_j < 0 \end{cases} \tag{29}$$

where y_j is jth dimension of $\mathbf{y}$, and $\tau_i^{(t)}$ and $\eta_{i,k}^{(t)}$ for $k = 1, \cdots, 4$ are five dynamic parameters that determine the intensity of irregularities and modality of the ith component. Equation (28) is designed in a way that the $\mathbf{c}$ positions have the best fitness values among all optima in their component and the global optimum is still the $\mathbf{c}$ of the component with the largest height. In GMPB, the term ith *component* means the area containing all the points whose Euclidean distances to the $\mathbf{c}_i$ is less than those to all $\mathbf{c}_j$ where $i \neq j$.

Figure 10 shows the effect of the different values of τ and $\eta_{1,2,3,4}$ on the shape of a component from Figure 10(a). As can be seen, higher values of τ and $\eta_{1,2,3,4}$ increase the irregularities and number of local optima in the component. Additionally, according to 10(b) and 10(c), when all four values of $\eta_{1,2,3,4}$ are equal, the component shape is symmetric. On the other hand, according to Figure 10(d), 10(e) and 10(f), when $\eta_{1,2,3,4}$ are unequal, the component shape is asymmetric. By changing τ and $\eta_{1,2,3,4}$ over time for each component, their intensity of irregularities, number of local optima, and asymmetry degree change over time. Figure 11 shows an example for how the shape of a component changes as the values of τ and $\eta_{1,2,3,4}$ change over time. Furthermore, Figure 12 shows examples of landscapes with the components from Figure 7(a) and 9(c) that are made irregular by different values of τ and $\eta_{1,2,3,4}$ for each peak. Equation (28) is the baseline of GMPB.

D. Modularity, heterogeneity, and imbalance

The modularity can be obtained by composing several GMPB:

$$F^{(t)}(\mathbf{x}) = \sum_{i=1}^{k} f_i^{(t)}(\mathbf{x}) \tag{30}$$

where $f_i^{(t)}$ is the ith baseline, and k is the number of subfunctions in the composition function. Since the baseline of GMPB is nonseparable, 1-dimensional subfunctions will be used for generating fully separable dimensions. Each subfunction in (30) can have a different number of components, dimensions, and/or change intensities. Consequently, since each subfunction can have different landscapes with different

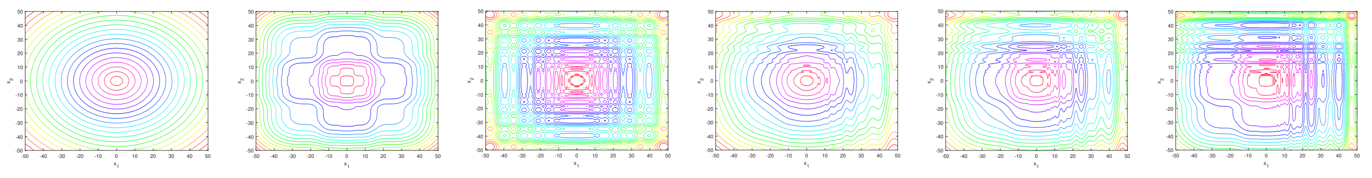

(a) $\tau = 0$, $\eta_{1,2,3,4} = 0$ (b) $\tau = 0.1$, $\eta_{1,2,3,4} = 5$ (c) $\tau = 0.1$, $\eta_{1,2,3,4} = 20$ (d) $\tau = 0.05$, $\eta_1 = 0$, $\eta_2 = 5$, $\eta_3 = 15$, and $\eta_4 = 25$ (e) $\tau = 0.1$, $\eta_1 = 0$, $\eta_2 = 5$, $\eta_3 = 15$, and $\eta_4 = 25$ (f) $\tau = 0.2$, $\eta_1 = 0$, $\eta_2 = 5$, $\eta_3 = 15$, and $\eta_4 = 25$

Fig. 10. Illustrating the effect of different values of τ and $\eta_{1,2,3,4}$ in Eq. (29) on a component.

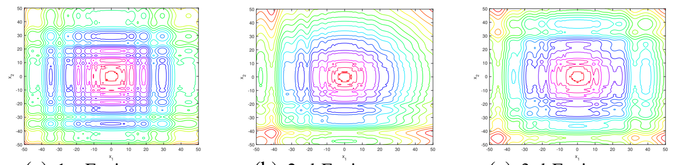

(a) 1st Environment. (b) 2nd Environment. (c) 3rd Environment.

Fig. 11. Illustrating the effect of dynamic τ and $\eta_{1,2,3,4}$ on shape of a component.

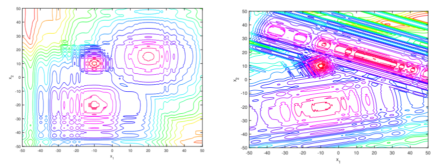

(a) Transformed version of the landscape from Figure 7(a) (b) Transformed version of the landscape from Figure 9(c)

Fig. 12. Transforming a landscape with three components using (29) with $\tau \in \mathcal{U}[0, 0.2]$ and $\eta_{1,2,3,4} \in \mathcal{U}[0, 25]$ for each main peak.

features and challenges, the generated problems by GMPB can possess the heterogeneity characteristic. A natural result of composing GMPB baselines in (30) is an exponential growth in the total number of optima in the landscape that can turn to the global optimum after environmental changes. To demonstrate this challenging characteristic, we provide an illustrative example in Fig 13. In Figure 13(a) and 13(b), two 1-dimensional landscapes with 2 and 5 optima, respectively, are shown (for simplicity, τ and $\eta_{1,2,3,4}$ are set to zero). The 2-dimensional modular landscape constructed by (30) results in a total of 10 optima as shown in Figure 13(c). To be specific, the number of optima in the modular landscapes constructed by (30) is equal to a product of the numbers of optima in the subfunctions.

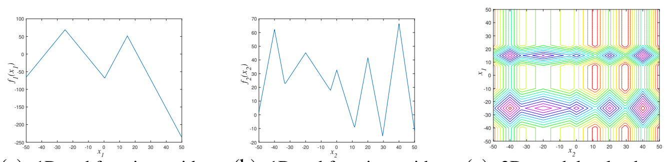
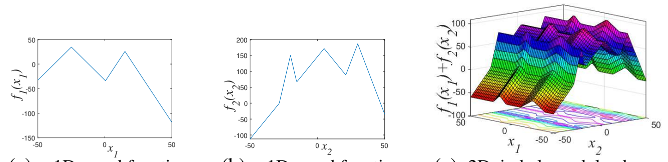

(a) 1D subfunction with two optima. (b) 1D subfunction with five optima. (c) 2D modular landscape with ten optima by composing (a) and (b).

Fig. 13. Exponentially growing number of optima by composing subfunctions in (30).



(a) 1D subfunction with ω=0.5. (b) 1D subfunction with ω=3. (c) 2D imbalanced landscape by composing (a) and (b).

Fig. 14. Imbalanced landscape generated by composing two subfunctions with different weights ω in (32).

According to the nature of baseline in (28), the fitness value is independent of the dimension number, and it is dependent on the highest h. Therefore, there is a balance between the contribution of subfunctions f in (30), which is not common in many real-world problems [3], [24], [64]. To address this issue, the contribution of each MPB is normalized according to its number of dimensions relative to the dimension of the problem:

$$F^{(t)}(\mathbf{x}) = d^{-1}\sum_{i=1}^{k} d_i f_i^{(t)}(\mathbf{x}) \tag{31}$$

where d is the dimension of the problem, and d_i is the dimension of the subfunction f_i. By using (31), the shortcoming of CMPB [18] regarding dominating contribution of higher dimensional subfunctions by low-dimensional ones is addressed. After normalizing the contribution of each subfunction, an imbalance coefficient ω will be added to the composition, which can be utilized to generate different imbalance patterns.

$$F^{(t)}(\mathbf{x}) = d^{-1}\sum_{i=1}^{k} \omega_i d_i f_i^{(t)}(\mathbf{x}) \tag{32}$$

where ω_i controls the contribution of subfunction f_i for generating imbalance. In an imbalanced problem, the contributions of subfunctions on the overall fitness value are different. Consequently, some subfunctions become more significant for optimizers. Figure 14(c) illustrates a 2-dimensional modular landscape constructed by composing two 1-dimensional subfunctions shown in Figures 14(a) and 14(b). The range of fitness values in the subfunction from Figure 14(b) is considerably larger than that of the subfunction from Figure 14(a). Consequently, the composed problem shown in Figure 14(c) is an imbalanced problem. According to the illustrated landscape in Figure 14(c), the progress of optimization by moving toward the direction of x_2 axis is more important than moving in x_1.

E. Dynamics

For each subfunction f_i in (32), the height, width vector, angle, irregularity parameters in (29) and center of the jth component change from one environment to the next according to the following update rules:

$$\mathbf{c}_{i,j}^{(t+1)} = \mathbf{c}_{i,j}^{(t)} + \tilde{s}_i \frac{\mathbf{r}}{\|\mathbf{r}\|}, \tag{33}$$

$$h_{i,j}^{(t+1)} = h_{i,j}^{(t)} + \tilde{h}_i \mathcal{N}(0,1), \tag{34}$$

$$w_{i,j,k}^{(t+1)} = w_{i,j,k}^{(t)} + \tilde{w}_i \mathcal{N}(0,1), \tag{35}$$

$$\theta_{i,j}^{(t+1)} = \theta_{i,j}^{(t)} + \tilde{\theta}_i \mathcal{N}(0,1), \tag{36}$$

$$\eta_{i,j,l}^{(t+1)} = \eta_{i,j,l}^{(t)} + \tilde{\eta}_i \mathcal{N}(0,1), l \in \{1,2,3,4\}, \tag{37}$$

$$\tau_{i,j}^{(t+1)} = \tau_{i,j}^{(t)} + \tilde{\tau}_i \mathcal{N}(0,1), \tag{38}$$

where $\mathcal{N}(0,1)$ is a random number drawn from a Gaussian distribution with mean 0 and variance 1, $\mathbf{r}$ is a vector of randomly generated numbers by $\mathcal{N}(0,1)$, $\tilde{h}_i$, $\tilde{w}_i$, $\tilde{s}_i$, $\tilde{\theta}_i$, $\tilde{\eta}_i$ and $\tilde{\tau}_i$ are height, width, shift, angle, and irregularity parameters' change severities of components in ih subfunction, respectively. $w_{i,j,k}$ shows the width of jth component in kth dimension of the ith subfunction. In addition, $h_{i,j}$, $\theta_{i,j}$, and $\mathbf{c}_{i,j}$ show height, angle, and position of jth component in ith subfunction f, respectively.

Outputs of equations (33) to (38) are bounded as follows: $h_{i,j} \in [h_{\min}, h_{\max}]$, $w_{i,j,k} \in [w_{\min}, w_{\max}]$, $\mathbf{c}_{i,j} \in [Lb, Ub]^d$, $\tau \in [\tau_{\min}, \tau_{\max}]$ and $\eta_{1,2,3,4} \in [\eta_{\min}, \eta_{\max}]$, and $\theta_{i,j} \in [\theta_{\min}, \theta_{\max}]$, where Lb and Ub are maximum and minimum problem space bounds. For keeping the above mentioned values in their bounds, a *Reflect* method is utilized. Assume $a^{(t+1)} = a^{(t)} + b$ represents one of the equations (34) to (38). Then, the output based on the reflect method is

$$a^{(t+1)} = \begin{cases} a^{(t)} + b & \text{if } a^{(t)} + b \in [a_{\min}, a_{\max}] \\ 2 \times a_{\min} - a^{(t)} - b & \text{if } a^{(t)} + b < a_{\min} \\ 2 \times a_{\max} - a^{(t)} - b & \text{if } a^{(t)} + b > a_{\max} \end{cases} \tag{39}$$

The dynamics which are used for GMPB in (34) to (38) are based on *random step*, which is also used in MPB. Any dynamics in (14) to (19) can also be used in GMPB. For example, researchers who are interested in periodical environmental changes can use the pendulum-based environmental changes similar to (11), or use the cyclic center relocations from [57]. To have predictable component relocations, (33) can be replaced with (8) and (9) (set $\lambda = 1$). To have undetectable environmental changes, a simple noise generator can be added to the fitness function. In addition, inspired by (10), the β threshold can be added to the benchmark to create vast flat regions whose fitness remain unchanged in environmental changes. By applying the aforementioned modifications, most change detection methods that work by re-evaluating solutions [2] cannot work properly. However, it should be mentioned that in most real-world DOPs, the algorithm will be informed about the environmental changes, and it is not necessary to use a change detection mechanism [1], [16]. Moreover, having partial environmental changes can be obtained by assigning different change frequencies to different subfunctions or different components. This can be done using a simple probability based method in which when an environmental change happens, components or subfunctions with a predefined probability can be involved in change. Finally, dynamic constraints can be easily added to GMPB. In Section S-III of the supplementary document, we have provide a discussion on the existing dynamic constrained benchmarks that can be incorporated to the GMPB.

TABLE II
SUMMARY OF SCENARIOS' CHARACTERISTICS INCLUDING SEPARABILITY (FULLY NON-SEPARABLE (N) AND PARTIALLY SEPARABLE (P)), COMPONENT MODALITY (MULTIMODAL (M) AND UNIMODAL (U)) AND ILL-CONDITIONING PLUS ROTATION

Feature	Scenario							
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
Separability	N	N	N	N	P	P	P	P
Component modality	U	U	M	M	U	U	M	M
Ill-conditioning and rotation	×	✓	×	✓	×	✓	×	✓

F. Proposed scenarios

We propose eight problem instances exhibiting different combinations of problem characteristics using the GMPB benchmark generator. To design these scenarios, the characteristics of components and problems are categorized into three groups: separability, local-modality, and ill-conditioning/rotation. The first group is defined according to the modularity of the problems based on (32). Two groups of problems are designed according to separability, including fully nonseparable and partially separable ones. For component modality, problems are categorized based on the value assigned to τ and $\eta_{1,2,3,4}$. If the values are set to zero, components will be smooth, regular, and unimodal. Otherwise, the problem contains components with different shapes and various features. Finally, if the rotation matrices are set to the identity matrix, angle severity is set to zero, and the width vector has the same value on all dimensions (with the same random number for all dimensions at environmental changes), components will have circular contour and become fully separable. Else, components will be ill-conditioned with dynamic condition number and variable interactions. The properties of the eight scenarios and the categories they belong to are summarized in Table II. The parameter settings of the problem instances and component characteristics are shown in Table III. To investigate the efficiency of DOP algorithms on each scenario with different levels of difficulty, two different parameter settings are considered for change frequency (ϑ), shift severity ($\tilde{s}$), and the number of components (m), which are shown in Table IV. The first being the default settings while the second group generates more challenging scenarios.

G. Discussion

To study the effect of different component characteristics, several well-known optimizers are selected and tested on the generated components by GMPB. The results and the

TABLE III
PARAMETER SETTINGS (PART 1) OF SCENARIOS f_1 TO f_8 FROM TABLE II.

Parameter	Symbol	Scenario							
		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
Number of subfunctions	k	1				6			
Dimension	d	10							
Number of fully separable dimensions	d'	0				2			
Number of Nonseparable subfunctions	d''	$\{10\}$				$\{4, 2, 2\}$			
Angle severities	$\tilde{\theta}_i$	0	$\pi/9$	0	$\pi/9$	0	$\mathcal{U}[\pi/12, \pi/6]$	0	$\mathcal{U}[\pi/12, \pi/6]$
Height severities	$\tilde{h}_i$	7				$\mathcal{U}[5, 9]$			
Width severities	$\tilde{w}_i$	1				$\mathcal{U}[0.5, 1.5]$			
Irregularity parameter τ severities	$\tilde{\tau}_i$	0		0.05		0		$\mathcal{U}[0.025, 0.075]$	
Irregularity parameter η severities	$\tilde{\eta}_i$	0		2		0		$\mathcal{U}[1, 3]$	
Weight of subfunction i	ω_i	1				$\mathcal{U}[0.5, 3]$			
Search range	$[Lb, Ub]^d$	$[-50, 50]^d$							
Height range	$[h_{\min}, h_{\max}]$	$[30, 70]$							
Width range	$[w_{\min}, w_{\max}]^d$	$[1, 12]^d$							
Angle range	$[\theta_{\min}, \theta_{\max}]$	-	$[-\pi, \pi]$	-	$[-\pi, \pi]$	-	$[-\pi, \pi]$	-	$[-\pi, \pi]$
Irregularity parameter τ range	$[\tau_{\min}, \tau_{\max}]$	-	-	$[0, 0.4]$	$[0, 0.4]$	-	-	$[0, 0.4]$	$[0, 0.4]$
Irregularity parameter η range	$[\eta_{\min}, \eta_{\max}]$	-	-	$[10, 25]$	$[10, 25]$	-	-	$[10, 25]$	$[10, 25]$
Number of Environments	T	100							

TABLE IV
TWO GROUPS OF PARAMETER SETTINGS (PART 2) OF SCENARIOS f_1-f_8.

Parameter	Symbol	Default setting		Challenging setting	
		f_{1-4}	f_{5-8}	f_{1-4}	f_{5-8}
Shift severities	$\tilde{s}_i$	2	$\mathcal{U}[1, 3]$	4	$\mathcal{U}[3, 5]$
Numbers of components in each subfunction i	m_i	10	$\mathcal{U}[5, 15]$	25	$\mathcal{U}[15, 35]$
Change frequency	ϑ	5000	5000	2500	2500

correspondence analysis can be found in Sections S-II-C1 and S-II-C2 of the supplementary document. To investigate the performance of existing DOP algorithms on the proposed GMPB scenarios, a set of 11 different DOP algorithms is chosen (see Section S-II-B of the supplementary document). The experimental results on the eight GMPB scenarios can be found in Sections S-II-D and S-II-E of the supplementary document. The experimental results clearly indicate the poor efficiencies of the existing algorithms in handling the challenges posed by GMPB.

IV. CONCLUSION

In this paper, we have presented a comprehensive review of continuous single-objective unconstrained dynamic optimization problem (DOP) benchmarks. The critical review of the existing benchmark suites showed that the landscape of most well-known DOP benchmarks is made up of one or more smooth, separable, symmetric, unimodal components with neutral condition numbers, which are easy to optimize and not representative of many real-world problems. We also have observed that there is no DOP benchmark in the literature capable of generating problems with different combinations of characteristics. To address the above mentioned shortcomings, we have proposed a new generalized moving peaks benchmark (GMPB). GMPB is capable of generating problems and components with varieties of properties. GMPB is configurable and can generate problems ranging from simple unimodal, smooth, regular, separable, symmetric, balanced, and homogeneous to highly multimodal, imbalanced, heterogeneous, partially separable problems with ill-conditioned, highly asymmetric, irregular, nonseparable components. Moreover, all the aforementioned features are dynamic and change over time in GMPB. Several well-known optimizers and DOP algorithms have been used to investigate the performance of the existing algorithms on the generated problem instances by GMPB. The experimental results have indicated the poor efficiencies of the existing algorithms in handling the challenges posed by GMPB. In the future, some other important characteristics can also be considered to be added to GMPB, such as dynamic constraints or multiple conflicting objectives, which are important classes of DOPs [10]–[12]. In addition, due to the importance of the discrete and combinatorial DOPs, investing existing problems and benchmark generators in this domain is an important topic for future work [6], [7].

REFERENCES

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

[2] T. T. Nguyen, S. Yang, and J. Branke, “Evolutionary dynamic optimization: A survey of the state of the art,” *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

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

[4] C. Li, T. T. Nguyen, S. Zeng, M. Yang, and M. Wu, “An open framework for constructing continuous optimization problems,” *IEEE Transactions on Cybernetics*, pp. 1–15, 2018.

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

[6] S. Yang, Y. Jiang, and T. T. Nguyen, “Metaheuristics for dynamic combinatorial optimization problems,” *IMA Journal of Management Mathematics*, vol. 24, no. 4, pp. 451–480, 2013.

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

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

[9] S. B. Gee, K. C. Tan, and H. A. Abbass, “A benchmark test suite for dynamic evolutionary multiobjective optimization,” *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 461–472, 2017.

[10] S. Jiang, M. Kaiser, S. Yang, S. Kollias, and N. Krasnogor, “A scalable test suite for continuous dynamic multiobjective optimization,” *IEEE Transactions on Cybernetics*, pp. 1–13, 2019.

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

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

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

[14] Y. Wang, J. Yu, S. Yang, S. Jiang, and S. Zhao, "Evolutionary dynamic constrained optimization: Test suite construction and algorithm comparisons," *Swarm and Evolutionary Computation*, vol. 50, p. 100559, 2019.

[15] X. Yu, Y. Jin, K. Tang, and X. Yao, "Robust optimization over time - a new perspective on dynamic optimization problems," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2010, pp. 1–6.

[16] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 01, pp. 143–155, 2019.

[17] D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, "A multi-objective time-linkage approach for dynamic optimization problems with previous-solution displacement restriction," in *European Conference on the Applications of Evolutionary Computation*, K. Sim and P. Kaufmann, Eds. Lecture Notes in Computer Science, 2018, vol. 10784, pp. 864–878.

[18] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," *IEEE Transaction on Evolutionary Computation*, 2019.

[19] W. Luo, B. Yang, C. Bu, and X. Lin, "A hybrid particle swarm optimization for high-dimensional dynamic optimization," in *Simulated Evolution and Learning*, Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin, Eds. Springer Lecture Notes in Computer Science, 2017, vol. 10593, pp. 981–993.

[20] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2010.

[21] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," Nanyang Technological University, Tech. Rep., 2005.

[22] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Swarm Intelligence Symposium*, 2005, pp. 68–75.

[23] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization," Nature Inspired Computation and Applications Laboratory, Tech. Rep., 2009.

[24] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Information Sciences*, vol. 316, pp. 419–436, 2015.

[25] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 1993, pp. 523–530.

[26] P. Angeline, "Tracking extrema in dynamic environments," in *Evolutionary Programming VI*, P. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart, Eds. Springer Lecture Notes in Computer Science, 1997, vol. 1213, pp. 335–345.

[27] K. Weicker and N. Weicker, "Dynamic rotation and partial visibility," in *Proceedings of the Congress on Evolutionary Computation*, 2000, p. 1125–1131.

[28] C. Li and S. Yang, "A generalized approach to construct benchmark problems for dynamic optimization," in *Simulated Evolution and Learning*, X. L. et al., Ed. Springer Lecture Notes in Computer Science, 2013, vol. 5361, pp. 391–400.

[29] R. Tinos and S. Yang, "A framework for inducing artificial changes in optimization problems," *Information Sciences*, vol. 485, pp. 486 – 504, 2019.

[30] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. Springer Natural Computing Series, 2003, pp. 239–262.

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

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

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

[34] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," in *Congress on Evolutionary Computation*, vol. 3, 2005, pp. 2808–2815.

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

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

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

[38] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, "A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," *Applied Soft Computing*, vol. 13, no. 04, pp. 2144–2158, 2013.

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

[40] C. Li and S. Yang, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 556–577, 2012.

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

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

[43] D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, "A new multi-swarm particle swarm optimization for robust optimization over time," in *Applications of Evolutionary Computation*, G. Squillero and K. Sim, Eds. Springer Lecture Notes in Computer Science, 2017, vol. 10200, pp. 99–109.

[44] D. Yazdani, J. Branke, M. N. Omidvar, T. T. Nguyen, and X. Yao, "Changing or keeping solutions in dynamic optimization problems with switching costs," in *Genetic and Evolutionary Computation Conference*, 2018, pp. 1095–1102.

[45] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Finding robust solutions to dynamic optimization problems," in *Applications of Evolutionary Computation*, vol. 7835. Lecture Notes in Computer Science, 2013, pp. 616–625.

[46] Y. nan Guo, M. Chen, H. Fu, and Y. Liu, "Find robust solutions over time by two-layer multi-objective optimization method," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1528—-1535.

[47] Y. Huang, Y. Ding, K. Hao, and Y. Jin, "A multi-objective approach to robust optimization over time considering switching cost," *Information Sciences*, vol. 394-395, pp. 183–197, 2017.

[48] S. Das, S. Maity, B.-Y. Qu, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization—a survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 71–88, 2011.

[49] R. Cheng, M. Li, K. Li, and X. Yao, "Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 692–706, 2017.

[50] W. Luo, X. Lin, T. Zhu, and P. Xu, "A clonal selection algorithm for dynamic multimodal function optimization," *Swarm and Evolutionary Computation*, vol. 50, p. 100459, 2019.

[51] B. Nasiri, M. Meybodi, and M. Ebadzadeh, "History-driven particle swarm optimization in dynamic and uncertain environments," *Neurocomputing*, vol. 172, pp. 356 – 370, 2016.

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

[53] T. M. Blackwell, "Swarms in dynamic environments," in *Conference on Genetic and Evolutionary Computation*, vol. 2723. Lecture Notes in Computer Science, Springer, 2003, pp. 19–26.

[54] C. Li, S. Yang, and M. Yang, "An adaptive multi-swarm optimizer for dynamic optimization problems," *Evolutionary Computation*, vol. 22, no. 4, pp. 559–594, 2014.

[55] C. Li, M. Mavrovouniotis, S. Yang, and X. Yao, "Benchmark generator for the ieee wcci-2014 competition on evolutionary computation for dynamic optimization problems," De Montfort University, Tech. Rep., 2013.

[56] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for cec'2009 competition

on dynamic optimization," Center for Computational Intelligence, Tech. Rep., 2008.

[57] T. Zhu, W. Luo, and L. Yue, "Dynamic optimization facilitated by the memory tree," *Soft Computing*, vol. 19, no. 3, pp. 547–566, 2014.

[58] K. Trojanowski and Z. Michalewicz, "Searching for optima in non-stationary environments," in *Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1843–1850.

[59] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[60] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929–942, 2017.

[61] A. M. Sutton, M. Lunacek, and L. D. Whitley, "Differential evolution and non-separability: Using selective pressure to focus search," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 1428–1435.

[62] X. Li, K. Tang, M. N. Omidvar, Z. Yang, , and K. Qin, "Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization," RMIT University, Tech. Rep., 2013.

[63] R. Tinos and S. Yang, "Continuous dynamic problem generators for evolutionary algorithms," in *Congress on Evolutionary Computation*, 2007, pp. 236–243.

[64] B. Kazimipour, M. N. Omidvar, A. Qin, X. Li, and X. Yao, "Bandit-based cooperative coevolution for tackling contribution imbalance in large-scale optimization problems," *Applied Soft Computing*, vol. 76, pp. 265–281, 2019.

Supplementary Document of ‘Benchmarking Continuous Dynamic Optimization: Survey and Generalized Test Suite’

Danial Yazdani, Mohammad Nabi Omidvar, Ran Cheng, Jürgen Branke, Trung Thanh Nguyen, and Xin Yao

CONTENTS

S-I. BACKGROUND

This section covers basic definitions such as dynamic optimization problems, variable interaction, imbalance, heterogeneity, and ill-conditioning.

a) Dynamic optimization problems: DOPs are usually represented as follows:

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

where f is the objective function, $\vec{x}$ is a design vector, $\vec{\alpha}^{(t)}$ are environmental parameters which change over time, t is the time index with $t \in [0, T]$, and T is the problem life cycle. In this paper, like most previous studies in the DOP domain, we consider DOPs that change discretely over time, i.e., $t \in \{1, \dots, T\}$ with stationary periods between changes. For a DOP with T environmental states, there is a sequence of T static environments:

$$F(\vec{x}) = \left[f(\vec{x}, \vec{\alpha}^{(1)}), f(\vec{x}, \vec{\alpha}^{(2)}), \dots, f(\vec{x}, \vec{\alpha}^{(T)})\right]. \tag{S-2}$$

b) Modularity and Variable Interaction: Real-world problems often have a modular structure [1]. The modularity is caused by the interaction structure of the decision variables resulting in a wide range of structures from fully separable functions to fully nonseparable ones. Variable interaction or linkage refers to the extent to which the optimum of a variable depends on the values taken by other decision variables. For continuous optimization problems, variable interaction is defined as follows [2]:

Definition 1. *[2] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function. Decision variables x_i and x_j interact if a candidate solution $\mathbf{x}^\star$ exists, such that*

$$\frac{\partial^2 f(\mathbf{x}^\star)}{\partial x_i \partial x_j} \neq 0. \tag{S-3}$$

Some functions exhibit an underlying interaction structure such that groups of decision variables can be optimized independently. These functions, which are called *partially separable*, are defined as follows:

Definition 2. *[1] A function $f(\mathbf{x})$ is partially separable with m independent components iff:*

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \left(\arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \dots), \dots, \arg\min_{\mathbf{x}_m} f(\dots, \mathbf{x}_m)\right), \tag{S-4}$$

where $\mathbf{x} = (x_1, \dots, x_n)^\top$ is a decision vector of n dimensions, $\mathbf{x}_1, \dots, \mathbf{x}_m$ are disjoint sub-vectors of $\mathbf{x}$, and $2 \leq m \leq n$. The function is called fully separable when $m = n$.

Additive separability is a special type of partial separability, which is defined as follows:

Definition 3. *[1] A function is additively separable if it has the following general form:*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_i),\ m > 1, \tag{S-5}$$

where $f_i(\cdot)$ is a nonseparable subfunction, and m is the number of nonseparable components of f. The definition of $\mathbf{x}$ and $\mathbf{x}_i$ is identical to what was given in Def. 2.

Definition 4. *[1] A function $f(\mathbf{x})$ is fully nonseparable if every pair of its decision variables interact.*

c) Imbalance and heterogeneity: Real-world problems often exhibit a modular structure with nonuniform imbalance among the contribution of its constituent parts to the objective value, commonly known as the *imbalance issue* [1], [3]–[5]. A partially separable problem is heterogeneous when its subfunctions have various characteristics and landscapes [1], [6]. Additionally, in DOPs, heterogeneity arises when subfunctions have different number of components, dimensions, and/or change intensities [7].

The imbalance property can be caused as a by-product of modularity or due to the heterogeneous nature of the input variables and their domains. For example, model predictive control (MPC) is a dynamic optimization problem with a wide range of applications in chemical power plants, robotics, and power systems, that exhibits modularity, heterogeneity, and imbalance [8].

d) Ill-conditioning: If the width value of a component is stretched in the direction of one axis more than the other axes, it is said that the component is ill-conditioned [9]. Ill-conditioning is an important property of many real-world problems [1], [9], [10].

S-II. Experimental Studies

In this section, we use several well-known optimizers and DOP algorithms to solve the generated problem instances by GMPB with various combinations of characteristics. By investigating the behavior and performance of the algorithms on the problems with the new properties, we can observe the shortcomings of the existing DOP methods in facing the new challenges posed by GMPB.

A. Evaluation metrics

To evaluate the performance and effectiveness of algorithms, three different evaluation metrics are used in this paper. First, the offline-error [11] (E_O) which is the average error of the best found position over all fitness evaluations:

$$E_O = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta}\left(f^{(t)}\left(\mathbf{x}^{\star(t)}\right) - f^{(t)}\left(\mathbf{x}^{*((t-1)\vartheta+c)}\right)\right), \tag{S-6}$$

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

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

where $\mathbf{x}^{*(t)}$ is the best found position in tth environment. The aforementioned evaluation metrics are performance-based measurements [12]. The third evaluation metric that we use in this paper is the average distance to optimum over time [13]:

$$E_D = \frac{1}{T\vartheta}\sum_{t=1}^{T}\sum_{c=1}^{\vartheta}\left\|\mathbf{x}^{\star(t)} - \mathbf{x}^{\circ((t-1)\vartheta+c)}\right\|, \tag{S-8}$$

where $\mathbf{x}^{\circ((t-1)\vartheta+c)}$ is the closest found position to the global optimum at the cth fitness evaluation in the tth environment. E_D is an efficiency-based evaluation metric that indicates the global optimum tracking effectiveness. As shown in [14], E_D values are not related to E_O and E_B. For example, a position may be fit, but it resides on a component whose distance to the global optimum component is far. In addition, when the component with the highest height is narrow, the fitness of solutions on its basin of attraction, whose distances to the global optimum are low, can be very poor in terms of fitness values.

B. Benchmark Algorithms

To study the effect of different component characteristics generated by GMPB, three well-known optimizers are selected: particle swarm optimization (PSO) [15], and two versions of differential evolution (DE): jDE [16], [17] bDE [18], [19], and covariance matrix adaptation evolution strategy (CMA-ES) [20].

For PSO, we use the global-best neighbourhood topology with constriction factor [21]. jDE is a well-known DE version that uses $DE/rand/1/bin$ strategy and self adaptive scaling factor (F) and crossover rate (Cr) [16]. bDE uses $DE/best/2/bin$ strategy and Brownian particles to improve the exploitation ability. In addition, bDE uses uniformly randomized F and Cr in each iteration. For CMA-ES, the version provided in [22] is used.

To investigate the performance of existing DOP algorithms on GMPB, a set of 11 different algorithms is chosen: AmQSO [23], CPSO [24], DynPopDE [25], FTmPSO [26], an improved version of DSPSO [27], mCMA-ES [3], mbDE [3], mjDE [3], mPSO [3], mQSO [28], and RPSO [29]. These algorithms are representative of different research approaches from several point of views [12], [30]: 1) different optimizers with various core procedures are used in the DOP algorithms including PSO, jDE, bDE, and CMA-ES; 2) different population configurations are used among them, for example mQSO is a multi-population algorithm with constant number of sub-populations. AmQSO, DSPSO, and CPSO are using an adaptive number of sub-populations, regrouping approaches, and clustering methods respectively. RPSO is a single population method that uses randomization after environmental changes; and 3) FTmPSO and DynPopDE use resource allocation methods.

For a better comparison, some parts of the algorithms are changed as follows: 1) The procedure of change detection is removed from all methods. It is assumed that algorithms are informed about environmental changes like many real-world cases [31]; 2) All PSO based algorithms use PSO with constriction factor version [21]. For CPSO, the procedure of updating Gbest is kept [24]; 3) All methods that use some knowledge about the shift severity (such as AmQSO and FTmPSO), use the shift severity calculation method from [32]; and 4) For addressing DSPSO shortcomings and improving its performance, the velocity of particles is randomized in (-1,1) after environmental changes. Moreover, the solo particles after determining species form a sub-population to participate in the optimization process. For CPSO, if all sub-populations are converged and deactivated, the converged sub-populations are allowed to do more exploitation until the next environmental change.

C. Empirical analysis

1) The effect of condition number and variable interaction: To investigate the impact of ill-conditioning and variable interactions, PSO, CMA-ES, jDE, and bDE are tested on a static 5-dimensional unimodal smooth problem (single component with $\tau = \eta_{1,2,3,4} = 0$) generated by (28). All tests in this section are repeated 101 times (with different random seeds) and algorithms stop running when the number of fitness evaluations reaches 2000. In this section, we investigate the impact of different component characteristics on the performance of the optimizers by using E_O and E_B as the performance indicators. In addition, since the test instances in this subsection are static, E_D is not considered for evaluating the tracking ability. For each result, the average E_O is calculated by (S-6), and the mean error at 500th (E_{500}), 1000th (E_{1000}) and 2000th (E_{2000}) fitness evaluations are reported (standard error in parenthesis). For each test, uniformly randomized height, width, location, and orthogonal matrix are used. In addition, the parameter settings of PSO, CMA-ES, jDE, and bDE are chosen according to the provided sensitivity analysis in [7].

Table S-I shows the obtained results by algorithms on four different problems based on various combinations of applying rotation (R) and ill-conditioning (I). Across all tests, the best results based on E_o are obtained by PSO because of its high convergence speed. In fact, the problems in this section are smooth and regular; therefore, the PSO which uses constriction factor and global best neighbourhood topology converges very fast towards the optimum which signifies a good exploitation ability. bDE has better local search ability in comparison to jDE based on E_{1000} and E_{2000}. bDE uses the best position in its mutation strategy, which improves its local search ability in comparison to jDE which performs better beyond 1000 fitness evaluation. However, due to the utilized mutation strategy in bDE, its convergence speed is lower than jDE at the first quarter of the optimization process when the population diversity is higher. By decreasing the population diversity, the convergence speed of bDE surpasses that of jDE. CMA-ES ranks last according to E_O due to the nature of the performance indicator, which is calculated by averaging the fitness of the best obtained position over time. The convergence speed of CMA-ES at the initial iterations of the optimization process is low such that the poor results in this part cause a poor averaged value over time. The results obtained by CMA-ES show that it outperforms other optimizers after 500 fitness evaluations.

According to the Table S-I, the obtained results on problems without ill-conditioning are exactly the same, irrespective of whether rotation is present (under the same random seed for both problems). This indicates that components consist of cone peaks without ill-conditioning (condition number of zero) are invariant to rotation. The results show that ill-conditioning alone (i.e., no rotation) can pose a challenge to optimizers.

TABLE S-I

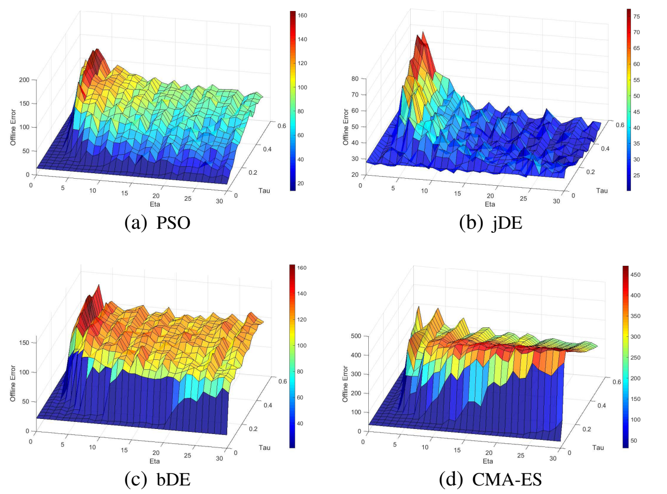
OBTAINED RESULTS BY PSO, JDE, BDE, AND CMA-ES ON A SINGLE COMPONENT PROBLEM UNDER DIFFERENT CONDITIONS BASED ON ROTATION (R) AND ILL-CONDITIONING (I). IN EACH ROW, BEST RESULTS ARE HIGHLIGHTED ACCORDING TO THE FRIEDMAN TEST WITH $\alpha - 0.05$.

R	I	Error	Optimizer			
			PSO	jDE	bDE	CMA-ES
		E_O	14.00(1.05)	22.89(2.04)	22.31(1.89)	35.53(2.21)
×	×	E_{500}	1.03(0.18)	12.86(2.33)	19.37(3.69)	1.04(0.10)
		E_{1000}	0.04(0.018)	6.69(2.03)	0.47(0.27)	9e-4(1e-4)
		E_{2000}	0.001(6e-04)	5.36(1.93)	0.003(3e0-4)	5e0-9(0.0)
		E_O	14.00(1.05)	22.89(2.04)	22.31(1.89)	35.53(2.21)
✓	×	E_{500}	1.03(0.18)	12.86(2.33)	19.37(3.69)	1.04(0.10)
		E_{1000}	0.04(0.018)	6.69(2.03)	0.47(0.27)	9e-4(1e-4)
		E_{2000}	0.001(6e-04)	5.36(1.93)	0.003(3e0-4)	5e0-9(0.0)
		E_O	12.66(0.52)	27.90(2.66)	26.44(1.94)	35.83(2.28)
×	✓	E_{500}	2.45(0.57)	15.82(2.96)	29.58(4.17)	1.20(0.13)
		E_{1000}	0.13(0.06)	9.63(2.40)	4.96(1.18)	0.001(1e-4)
		E_{2000}	0.04(0.02)	6.97(1.97)	0.39(0.28)	5e0-9(0.0)
		E_O	17.84(1.22)	30.18(2.11)	31.39(2.22)	36.18(2.15)
✓	✓	E_{500}	11.87(2.13)	21.34(2.50)	39.20(3.79)	1.18(0.14)
		E_{1000}	3.31(0.79)	13.60(2.20)	11.73(2.02)	0.001(0.002)
		E_{2000}	1.62(0.48)	10.19(1.87)	1.59(0.46)	6e0-8(0.0)

When the rotation is added to an ill-conditioned component, the problem becomes more challenging for PSO, jDE and bDE as a consequence of changing variable interactions from separable to nonseparable. Note that for each run, the orthogonal matrix is generated randomly which leads to test instances with various degrees of variable interaction. Moreover, the results indicate that CMA-ES is invariant to rotations even when ill-conditioning is present.

2) Effect of irregularities of components: To investigate the effect of different levels of irregularity, a 5-dimensional single-subfunction single-component problem (without rotation and ill-conditioning) with various combination of τ and $\eta_{1,2,3,4}$ values is used in this part. In addition, the component for each test is symmetric, i.e., $\eta_1 = \eta_2 = \eta_3 = \eta_4$. Figure S-1 illustrates the effect of different τ and $\eta_{1,2,3,4}$ combinations based on E_O on the performance of PSO, jDE, bDE, and CMA-ES. According to Figures S-1(a), S-1(b), and S-1(b), the performance of the algorithms drop when irregularity is added to the component. According to the results, when $\eta_{1,2,3,4} \in (3, 5)$ and $\tau > 0.2$ the problem is more challenging. Based on our investigations, these parameter settings of τ and $\eta_{1,2,3,4}$ create multiple large local optima in the component, which increases the possibility of premature convergence of the algorithms. By increasing the value of η, the number of local optima increases but their size become smaller, which makes it easier for the algorithms to escape local optima.

By comparing Figures S-1(a), S-1(b), S-1(c), and S-1(d), it is obvious that jDE performs better than the other methods in irregular environments. The reason is that jDE has a good exploration ability helping it to prevent trapping in local optima. However, as shown in Section S-II-C1 and in Figure S-1(b) ($\eta = 0$ and/or $\eta_1 = \eta_2 = \eta_3 = \eta_4 = 0$), due to its inferior convergence speed and exploitation ability, it performs worse than other methods on regular and smooth environments. According to Figures S-1(a) and S-1(c), PSO and bDE significantly suffer from irregularities. The reason is that both PSO and bDE are susceptible to local optima and



(a) PSO (b) jDE

(c) bDE (d) CMA-ES

Fig. S-1. Obtained results (E_O) by PSO, jDE, bDE, and CMA-ES on a 5-dimensional single component with different τ and $\eta_{1,2,3,4}$ combinations.

premature convergence. Comparing Figures S-1(a) and S-1(c) indicates that PSO outperforms bDE in irregular environments, As can be seen in Figure S-1(d), the performance of CMA-ES deteriorates significantly on irregular problems. In fact, CMA-ES shows the worst performance on preventing early convergence among the tested algorithms.

3) Effect of modularity of problem: Finally, to study the effect of modularity, the following two test problems are proposed: 1) a fully non-separable 10-dimensional problem with 10 components (GMPB(10_{10})); and 2) a partially separable problem with two fully nonseparable 5-dimensional subfunctions with 5 components in each (GMPB$(5_5 + 5_5)$); components in both problems are unimodal, smooth, regular, symmetric, separable, and with condition number of zero. Table S-II shows the obtained results by different DOP algorithms on the two cases. Each result in Table S-II is obtained by 31 independent runs on 100 environments and until 500,000 fitness evaluations. The remaining parameters are set the same as scenario 1 (f_1) from Table III.

Both tested problems in this section are 10-dimensional but one of them is constructed by composition of two 5-dimensional subproblems using (31). According to [3], [7], the number of physical peaks in composing MPBs can be up to the product of number of peaks in all subfunctions. Therefore, although GMPB$(5_5 + 5_5)$ has 5 peaks in each subfunction, its overall landscape can contain up to 25 physical peaks. This exponentially growing number of peaks by composing subfunctions from Eq. (29) is the main reason behind the worse results of the algorithms on GMPB$(5_5 + 5_5)$ compared to the obtained results on GMPB(10_{10}). FTmPSO outperforms other methods in both tested problems since it uses a resource allocation method which improves its performance, especially when the number of peaks are higher [26].

D. Investigating the performance of the DOP algorithms on eight GMPB scenarios

In this section, the performance of the DOP algorithms from Section S-II-B are studied on the eight scenarios of GMPB from Table II. To measure the performance of the DOP algorithms, E_O and E_B as two performance-based evaluation metrics, are used in this part. The obtained results on the scenarios with the default parameter settings from Table IV are shown in Table S-III. According to Table S-III, the best results are obtained for f_1 which is the equivalent to the traditional MPB and the easiest of all scenarios due to lack of modularity, and having easy to optimize components. f_2 has regular and smooth components but they are ill-conditioned and non-separable. As expected, mCMA-ES whose performance is invariant to rotations, obtains the best results on f_2. f_3 contains irregular and complex components but they are not rotated and stretched from any direction. Surprisingly, mCMA-ES obtains the best E_O among the algorithms which was not expected according to Figure 1(d). The reason behind this improvement of mCMA-ES in comparison to CMA-ES is benefiting from the multiple-population approach. In fact, using multiple-population and re-initializing the explorer (finder) sub-population after convergence or stagnating, addresses CMA-ES's flaw of being stuck in local optima. According to the results of Table S-III, f_4 is the most challenging non-separable problem because its components are irregular, ill-conditioned, asymmetric, non-separable whose degrees change over time. Although the performance of mCMA-ES dropped considerably in f_4, it outperforms all other methods. For f_1-f_4, rather than CMA-ES, PSO based algorithms with an adaptive multi-population approach – i.e., FTmPSO, AmQSO, and mPSO – outperform other algorithms. One reason is the superior convergence speed of PSO in comparison to DE-

TABLE S-II
OBTAINED RESULTS (MEAN AND STANDARD ERROR) BY DOP ALGORITHMS ON GMPB(10_{10}) AND GMPB$(5_5 + 5_5)$. THE BEST OBTAINED RESULTS ARE IN BOLD FOR E_B AND HIGHLIGHTED FOR E_O BASED ON THE FRIEDMAN TEST WITH $\alpha = 0.05$.

Algorithm	Error	Problem instance	
		GMPB(10_{10})	GMPB$(5_5 + 5_5)$
AmQSO	E_O	4.99(0.17)	6.81(0.21)
	E_B	2.78(0.15)	4.00(0.16)
CPSO	E_O	10.05(0.35)	10.55(0.24)
	E_B	7.29(0.31)	7.05(0.19)
DynPopDE	E_O	13.38(0.41))	17.04(0.51)
	E_B	12.63(0.40)	15.99(0.48)
FTmPSO	E_O	3.25(0.15)	4.57(0.15)
	E_B	**1.99(0.15)**	**2.54(0.12)**
IDSPSO	E_O	6.56(0.26)	6.85(0.2)
	E_B	4.30(0.27)	4.90(0.19)
mCMA-ES	E_O	3.63(0.21)	5.19(0.17)
	E_B	2.51(0.24)	3.34(0.18)
mbDE	E_O	6.13(0.22)	8.07(0.22)
	E_B	3.60(0.21)	4.83(0.18)
mjDE	E_O	9.10(0.31)	10.13(0.32)
	E_B	6.24(0.29)	7.21(0.27)
mPSO	E_O	4.81(0.20)	6.77(0.15)
	E_B	2.77(0.18)	3.97(0.12)
mQSO	E_O	8.11(0.21)	10.04(0.25)
	E_B	6.17(0.18)	7.60(0.22)
RPSO	E_O	20.78(0.71)	16.14(0.43)
	E_B	16.62(0.72)	12.72(0.41)

based methods. Another reason is adaptive multi-population approach that improves the premature convergence issue. FTmPSO outperforms AmQSO and mPSO because of its additional mechanisms such as resource allocation [26].

Scenarios f_5-f_8 are partially separable problems with two fully separable dimensions. In terms of component characteristics, f_5-f_8 are similar to f_1-f_4, respectively. However, the modularity property of f_5-f_8 leads to exponential growth in the number of optima as indicated in Figure 13, which significantly increases their difficulties. According to Table S-III, CPSO obtains the best results in f_5-f_8. As stated before, the overall landscape of these problems are usually highly multimodal due to their modularity and the exponentially growing of number of optima. Therefore, CPSO outperforms other algorithms because of its clustering based multi-population and re-diversification approaches which increase its exploration capability at the beginning of each environment. For algorithms with adaptive sub-population generation mechanism such as FTmPSO, AmQSO, and mPSO, the higher number of peaks in the landscape results in simultaneous creation and running of more sub-populations. Consequently, higher numbers of sub-populations decrease their efficiencies due to higher fitness evaluation consumptions in each iteration.

Tables S-IV, S-V, and S-VI, show the obtained results on the scenarios according to the challenging parameter settings from Table IV where the shift severities, number of components, and change frequencies are higher. By comparing the reported results in Tables S-III and S-IV, it can be seen that the problems with higher shift severities are more challenging. This performance deterioration is a consequence of larger displacement of component centers after an environmental change. Table S-V shows that problems with a higher number of components are more challenging because finding and covering higher number of components consumes more computational resources, which decreases the efficiency of algorithms. Finally, based on the results of Table S-VI, problems with higher change frequencies are more difficult because they give the algorithms less time and computational resource to cover and track optima in each environment.

E. Investigating the tracking efficiency of the DOP algorithms on eight GMPB scenarios

In this section, the tracking efficiencies of the DOP algorithms from Section S-II-B are studied on the eight scenarios of GMPB from Table II. We use E_D in this part to measure the tracking efficiencies of the DOP algorithms. The obtained results on the scenarios with the default parameter settings from Table IV are shown in Table S-VII. The results show that the patterns captured by E_D are different from those of the other performance metrics, i.e., E_O and E_B. This is due to the fact that the fitness values of solutions are not always related to their distance from the global optimum.

According to Table S-VII, the results deteriorate when the problems are modular(f_5-f_8). Additionally, the tracking efficiencies of the DOP algorithms drop in ill-conditioned and irregular problems. Overall, FTmPSO outperforms other methods, and mjDE obtains the best results on majority of cases on f_5-f_8.

S-III. A note on constrained DOP benchmarks and adding constraints to GMPB

Many real-world optimization problems may involve constraints. By adding constraints to GMPB, another class of optimization problems called dynamic constrained optimization problems (DCOPs) can be covered. In DCOPs, both or either of the objective function and/or constraints can be time-varying. In this section, we investigate the existing DCOP benchmark generators and analyze their adaptability and flexibility in order to incorporate constraints into GMPB. DCOP benchmarks can be categorized into two groups:

- DCOP benchmarks with inflexible structure which are constructed by combining time dependent variables with constrained static functions, and
- DCOP benchmarks with controllable structure to build a landscape with multiple moving disjointed feasible regions.

A. DCOP benchmarks with inflexible structure

In [33], a set of three DCOP benchmark problems are conducted. Both objective function and constraint(s) are time-varying in this set of problem instances. A commonly used DCOP benchmark is G24 [34]. The idea is to combine existing static constrained problems with time-dependent parameters to construct dynamic objective functions and constraints. The same idea is used to construct DCOPs from static constrained problem instances in [35]. All the above mentioned DCOP benchmarks are not suitable to be combined with GMPB since they are not scalable and flexible. In fact, their mathematical model is fixed, and the characteristics of their landscapes and feasible regions cannot be controlled.

B. DCOP benchmarks with controllable structure

In [36], several moving disjointed feasible regions with different shapes are added to the modified MPB from (10). Each feasible region is constructed by:

$$g_j^{(t)}(\mathbf{x}) = \left\| b_j^{(t)}\mathbf{x} - \mathbf{c'}_j^{(t)} \right\|_{p_j^{(t)}} - r_j^{(t)} \leq 0,\ j = 1, 2, \cdots, \mathfrak{m}, \tag{S-9}$$

where $g_j^{(t)}$ is jth constraint in tth environment that forms a feasible region, $\mathfrak{m}$ is the number of feasible regions, $\mathbf{c'}_j$ is the center of jth feasible region, r_j defines the size of jth feasible region, b_j defines the relative size based on the different spatial direction of the solution $\mathbf{x}$, and $\|\cdot\|_{p_j}$ is the p_j-norm function. p_j can define the shape of jth feasible region. For example, the feasible region can be diamond like if $p_j = 1$, or it can be a ball shape if $p_j = 2$. All the aforementioned parameters can be time-dependant. Consequently, the location, size, and shape of each feasible region can change over time. In this benchmark, when a solution $\mathbf{x}$ lies inside the defined region by any $g_j^{(t)}$, it is considered as a feasible solution. Therefore, a solution $\mathbf{x}$ in tth environment is feasible if:

$$\{\exists g_j^{(t)}(\mathbf{x}) | g_j^{(t)}(\mathbf{x}) \leq 0,\ j = 1, 2, \cdots, \mathfrak{m}\}. \tag{S-10}$$

This type of feasible regions can be added to GMPB to construct a DCOP benchmark generator.

TABLE S-III
OBTAINED RESULTS (MEAN AND STANDARD ERROR) BY DOP ALGORITHMS ON EIGHT GMPB SCENARIOS FROM TABLE II WITH THE DEFAULT SETTINGS FROM TABLE IV. THE BEST OBTAINED RESULTS ARE IN BOLD FOR E_B AND HIGHLIGHTED FOR E_O BASED ON THE FRIEDMAN TEST WITH $\alpha = 0.05$.

Algorithm	Error	GMPB Scenario							
		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
AmQSO	E_O	4.99(0.17)	7.76(0.13)	8.14(0.25)	13.07(0.21)	17.41(0.72)	21.07(0.98)	17.78(0.69)	21.34(0.92)
	E_B	2.78(0.15)	4.02(0.10)	5.83(0.27)	8.57(0.18)	13.73(0.63)	16.33(0.76)	14.24(0.60)	16.89(0.74)
CPSO	E_O	10.05(0.35)	14.03(0.21)	10.83(0.30)	24.24(0.61)	14.76(0.63)	18.71(0.98)	15.67(0.62)	20.15(0.96)
	E_B	7.29(0.31)	8.61(0.19)	8.04(0.25)	18.71(0.60)	**10.27(0.50)**	**12.74(0.65)**	**11.36(0.47)**	**15.38(0.71)**
DynPopDE	E_O	13.38(0.41)	19.28(0.49)	50.16(2.73)	80.07(4.19)	26.48(1.12)	31.60(1.75)	43.88(2.03)	50.42(2.42)
	E_B	12.63(0.40)	17.66(0.47)	46.64(2.41)	72.77(3.91)	24.05(1.06)	28.32(1.59)	41.17(1.93)	47.06(2.29)
FTmPSO	E_O	3.25(0.15)	5.44(0.12)	7.15(0.29)	12.70(0.34)	15.39(0.65)	19.38(0.94)	16.76(0.59)	20.22(0.84)
	E_B	**1.91(0.15)**	**1.99(0.08)**	**4.95(0.25)**	**7.11(0.26)**	11.42(0.52)	14.12(0.70)	12.88(0.51)	**15.21(0.62)**
IDSPSO	E_O	6.56(0.26)	7.81(0.21)	8.49(0.26)	13.92(0.49)	17.23(0.78)	18.61(0.92)	19.22(0.74)	22.41(1.07)
	E_B	4.30(0.27)	4.76(0.19)	6.67(0.24)	11.09(0.49)	15.04(0.73)	15.88(0.83)	17.17(0.70)	20.07(0.98)
mCMA-ES	E_O	3.63(0.21)	4.23(0.06)	5.65(0.24)	7.59(0.24)	15.29(0.66)	17.18(0.86)	17.7(0.65)	19.74(0.88)
	E_B	2.01(0.24)	1.43(0.06)	3.82(0.24)	5.11(0.22)	12.85(0.59)	14.12(0.75)	15.69(0.61)	17.33(0.78)
mbDE	E_O	6.13(0.22)	9.03(0.20)	10.94(0.38)	19.66(0.49)	19.07(0.73)	23.21(1.20)	21.21(0.84)	25.39(1.15)
	E_B	3.60(0.21)	4.85(0.17)	8.46(0.35)	14.94(0.47)	15.38(0.64)	18.46(0.96)	17.77(0.73)	21.13(0.95)
mjDE	E_O	9.10(0.31)	14.97(0.42)	9.31(0.34)	19.59(1.09)	16.57(0.71)	21.58(1.26)	15.60(0.75)	21.27(1.22)
	E_B	6.24(0.29)	10.73(0.39)	6.55(0.33)	15.20(0.99)	12.55(0.60)	16.89(1.06)	**11.39(0.68)**	16.99(1.08)
mPSO	E_O	4.81(0.20)	7.47(0.12)	8.20(0.27)	12.88(0.27)	17.03(0.65)	20.79(1.08)	18.02(0.67)	21.36(0.93
	E_B	2.77(0.18)	3.77(0.09)	5.72(0.25)	8.51(0.22)	13.37(0.57)	16.10(0.85)	14.30(0.56)	16.97(0.75)
mQSO	E_O	8.11(0.21)	10.75(0.16)	11.52(0.38)	17.05(0.43)	17.43(0.75)	20.32(1.10)	18.54(0.8)	21.72(0.97)
	E_B	6.17(0.18)	7.45(0.15)	9.62(0.34)	13.74(0.38)	14.33(0.67)	16.20(0.91)	15.53(0.73)	17.85(0.79)
RPSO	E_O	20.79(0.71)	21.26(0.75)	21.08(0.64)	24.03(0.69)	19.25(1.07)	23.45(1.49)	19.56(1.07)	22.31(1.17)
	E_B	16.62(0.72)	16.11(0.73)	16.33(0.62)	18.76(0.68)	14.71(0.91)	17.38(1.25)	14.97(0.94)	16.44(0.91)

TABLE S-IV
OBTAINED RESULTS (MEAN AND STANDARD ERROR) BY DOP ALGORITHMS ON EIGHT GMPB SCENARIOS FROM TABLE II WITH THE CHALLENGING SETTINGS FROM TABLE IV FOR THE SHIFT SEVERITIES. THE BEST OBTAINED RESULTS ARE IN BOLD FOR E_B AND HIGHLIGHTED FOR E_O BASED ON THE FRIEDMAN TEST WITH $\alpha = 0.05$.

Algorithm	Error	GMPB Scenario							
		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
AmQSO	E_O	7.58(0.22)	12.08(0.13)	11.4(0.31)	19.22(0.27)	21.06(0.86)	26.61(1.32)	22.01(0.95)	27.79(1.17)
	E_B	4.00(0.19)	6.02(0.08)	7.78(0.24)	12.18(0.20)	15.24(0.70)	19.10(0.98)	16.14(0.72)	20.55(0.85)
CPSO	E_O	13.38(0.36)	19.78(0.24)	13.84(0.49)	32.59(0.66)	17.55(0.75)	22.88(1.18)	18.63(0.90)	24.78(1.19)
	E_B	8.83(0.29)	11.44(0.21)	9.86(0.40)	24.45(0.61)	**11.17(0.50)**	**14.63(0.78)**	**12.68(0.66)**	**17.57(0.85)**
DynPopDE	E_O	23.01(0.52)	38.37(0.62)	69.45(2.81)	111.78(2.91)	38.02(1.54)	45.93(2.46)	55.66(2.78)	65.12(3.01)
	E_B	21.73(0.51)	35.19(0.63)	64.80(2.53)	103.03(2.60)	34.11(1.41)	40.35(2.18)	52.23(2.61)	60.84(2.82)
FTmPSO	E_O	4.82(0.22)	7.84(0.14)	9.16(0.35)	16.06(0.35)	19.26(0.89)	24.96(1.22)	20.57(0.90)	25.51(1.08)
	E_B	**2.77(0.22)**	**2.74(0.10)**	**6.03(0.31)**	**8.70(0.26)**	13.58(0.67)	17.23(0.87)	14.96(0.68)	18.41(0.80)
IDSPSO	E_O	8.54(0.32)	10.92(0.17)	11.67(0.45)	18.18(0.52)	19.33(0.87)	23.51(1.02)	24.99(1.03)	30.3(1.39)
	E_B	5.11(0.30)	6.41(0.16)	9.15(0.42)	14.19(0.52)	15.71(0.78)	19.08(0.85)	21.88(0.93)	26.55(1.24)
mCMA-ES	E_O	4.66(0.2)	6.18(0.08)	7.32(0.4)	11.65(0.28)	20.03(0.79)	23.86(1.25)	26.13(0.99)	30.2(1.23)
	E_B	2.20(0.22)	1.90(0.05)	4.66(0.43)	7.39(0.26)	15.24(0.65)	17.65(0.94)	21.81(0.84)	25.91(1.06)
mbDE	E_O	8.00(0.32)	14.5(0.16)	17.32(0.40)	29.24(0.57)	24.79(0.97)	30.58(1.43)	27.5(1.20)	33.73(1.53)
	E_B	3.67(0.25)	6.84(0.12)	13.45(0.36)	21.42(0.55)	19.12(0.82)	23.51(1.11)	22.48(1.01)	27.20(1.25)
mjDE	E_O	12.5(0.39)	22.76(0.48)	13.32(0.5)	27.44(0.69)	19.16(0.93)	26.14(1.56)	19.23(1.02)	25.43(1.32)
	E_B	8.12(0.32)	15.79(0.47)	8.9(0.38)	20.30(0.67)	13.28(0.77)	19.07(1.24)	13.41(0.80)	18.98(1.08)
mPSO	E_O	7.16(0.22)	12.19(0.16)	11.62(0.33)	18.52(0.26)	21.03(0.83)	26.97(1.36)	22.21(0.96)	27.69(1.26)
	E_B	3.62(0.18)	6.05(0.11)	7.98(0.27)	11.63(0.19)	15.04(0.66)	19.43(1.01)	16.29(0.74)	20.46(0.94)
mQSO	E_O	12.68(0.32)	17.59(0.19)	15.27(0.45)	24.00(0.44)	21.68(0.89)	26.18(1.39)	23.44(1.07)	27.48(1.27)
	E_B	9.37(0.26)	11.75(0.16)	12.35(0.42)	18.41(0.43)	16.73(0.75)	19.43(1.07)	18.65(0.92)	21.42(1.01)
RPSO	E_O	23.10(0.71)	26.81(0.67)	23.3(0.63)	27.73(0.73)	19.73(1.04)	24.39(1.37)	20.18(1.14)	23.45(1.24)
	E_B	16.46(0.67)	18.38(0.63)	16.21(0.65)	19.47(0.69)	14.08(0.83)	15.98(1.01)	14.16(0.94)	15.54(0.95)

In [37], a DCOP benchmark with multiple disjointed moving feasible regions is proposed, where both objective function and constraints are based on multi-component baselines. The baseline in (4) is used as the objective function subjected to:

$$g^{(t)}(\mathbf{x}) = \delta^{(t)} - C^{(t)}(\mathbf{x}) \leq 0 \qquad \text{(S-11)}$$

where $C(\cdot)$ is a multi-component function (maximization), TABLE S-V

Obtained results (mean and standard error) by DOP algorithms on eight GMPB scenarios from Table II with the challenging settings from Table IV for the numbers of components. The best obtained results are in bold for E_B and highlighted for E_O based on the Friedman test with $\alpha = 0.05$.

Algorithm	Error	GMPB Scenario							
		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
AmQSO	E_O	5.36(0.12)	10.00(0.12)	7.80(0.21)	14.16(0.26)	16.88(0.77)	19.96(0.86)	16.63(0.69)	21.09(0.97)
	E_B	3.29(0.11)	5.96(0.10)	5.69(0.18)	9.80(0.21)	13.62(0.64)	15.93(0.71)	13.54(0.58)	17.06(0.81)
CPSO	E_O	8.66(0.21)	15.49(0.21)	9.27(0.23)	24.40(0.42)	13.59(0.58)	17.36(0.80)	13.82(0.58)	19.86(1.00)
	E_B	6.39(0.19)	10.10(0.19)	7.06(0.21)	19.06(0.40)	**9.56(0.45)**	**11.99(0.57)**	**10.13(0.45)**	**14.86(0.79)**
DynPopDE	E_O	11.60(0.34)	18.56(0.54)	48.10(2.08)	67.54(3.21)	24.58(1.04)	27.67(1.22)	35.11(1.60)	42.79(2.11)
	E_B	10.93(0.33)	16.82(0.52)	44.84(1.91)	61.03(2.89)	22.31(0.96)	24.63(1.07)	32.74(1.52)	39.73(1.97)
FTmPSO	E_O	3.80(0.13)	7.42(0.09)	6.53(0.15)	13.48(0.21)	15.64(0.70)	18.59(0.78)	15.88(0.62)	20.73(0.88)
	E_B	**2.41(0.12)**	**3.25(0.06)**	**4.43(0.13)**	**7.99(0.16)**	12.04(0.56)	13.96(0.60)	12.41(0.52)	16.09(0.69)
IDSPSO	E_O	6.06(0.18)	8.79(0.20)	8.73(0.24)	14.36(0.47)	17.43(0.82)	19.04(0.79)	18.88(0.85)	22.83(1.01)
	E_B	4.10(0.16)	6.02(0.19)	7.06(0.24)	11.62(0.45)	15.49(0.80)	16.80(0.72)	17.15(0.82)	20.66(0.95)
mCMA-ES	E_O	4.41(0.15)	6.06(0.06)	5.11(0.13)	8.49(0.16)	15.27(0.68)	16.82(0.66)	16.98(0.76)	19.56(0.89)
	E_B	2.56(0.15)	3.31(0.06)	3.42(0.12)	6.14(0.15)	12.96(0.6)	14.13(0.58)	15.19(0.69)	17.28(0.82)
mbDE	E_O	5.91(0.14)	10.71(0.24)	10.99(0.30)	18.94(0.32)	19.09(0.84)	21.87(0.94)	19.28(0.82)	24.85(1.16)
	E_B	3.54(0.11)	6.32(0.20)	8.74(0.29)	14.16(0.31)	15.76(0.72)	17.86(0.77)	16.36(0.72)	20.83(0.97)
mjDE	E_O	9.11(0.19)	16.66(0.34)	9.55(0.24)	20.95(0.49)	15.66(0.83)	21.00(1.23)	14.49(0.71)	21.93(1.11)
	E_B	6.82(0.18)	12.65(0.33)	7.15(0.22)	16.58(0.46)	12.25(0.75)	17.11(1.09)	11.26(0.62)	18.07(0.97)
mPSO	E_O	5.25(0.12)	9.78(0.11)	7.60(0.20)	13.86(0.21)	16.77(0.68)	19.80(0.86)	16.89(0.68)	21.31(0.97)
	E_B	3.27(0.10)	5.81(0.08)	5.46(0.16)	9.58(0.17)	13.53(0.58)	15.81(0.70)	13.81(0.58)	17.25(0.82)
mQSO	E_O	8.15(0.21)	11.97(0.17)	10.53(0.33)	17.32(0.31)	16.34(0.69)	18.92(0.87)	16.70(0.76)	21.28(1.00)
	E_B	6.53(0.20)	8.75(0.17)	8.92(0.32)	14.15(0.29)	13.45(0.61)	15.22(0.72)	14.01(0.68)	17.71(0.86)
RPSO	E_O	22.75(0.81)	24.07(0.79)	22.36(0.90)	25.10(0.71)	17.05(0.99)	20.05(1.11)	16.15(0.90)	21.18(1.12)
	E_B	18.60(0.73)	18.97(0.77)	18.45(0.88)	19.84(0.71)	12.76(0.87)	14.38(0.92)	11.99(0.76)	14.28(0.90)

TABLE S-VI

Obtained results (mean and standard error) by DOP algorithms on eight GMPB scenarios from Table II with the challenging settings from Table IV for the change frequencies. The best obtained results are in bold for E_B and highlighted for E_O based on the Friedman test with $\alpha = 0.05$.

Algorithm	Error	GMPB Scenario							
		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
AmQSO	E_O	7.30(0.29)	11.46(0.20)	9.58(0.35)	17.62(0.31)	20.92(0.91)	25.36(1.27)	22.25(0.91)	26.37(1.29)
	E_B	4.73(0.27)	6.96(0.15)	7.78(0.29)	12.54(0.27)	17.55(0.81)	21.03(1.10)	18.70(0.80)	22.05(1.10)
CPSO	E_O	12.58(0.50)	22.50(0.27)	13.50(0.40)	33.86(0.58)	18.47(0.78)	25.19(1.34)	19.67(0.86)	26.26(1.29)
	E_B	10.71(0.48)	16.83(0.26)	10.98(0.36)	27.61(0.52)	**13.52(0.58)**	18.40(0.98)	**14.56(0.66)**	20.06(1.01)
DynPopDE	E_O	23.01(0.52)	38.37(0.62)	69.45(2.81)	111.78(2.91)	38.02(1.54)	45.93(2.46)	55.66(2.78)	65.12(3.01)
	E_B	21.73(0.51)	35.19(0.63)	64.80(2.53)	103.03(2.60)	34.11(1.41)	40.35(2.18)	52.23(2.61)	60.84(2.82)
FTmPSO	E_O	4.93(0.25)	9.33(0.16)	9.41(0.23)	18.04(0.39)	18.58(0.79)	23.24(1.14)	20.81(0.83)	24.46(1.06)
	E_B	**2.98(0.26)**	**4.24(0.10)**	**6.64(0.20)**	**11.57(0.32)**	14.30(0.63)	**17.66(0.89)**	16.54(0.70)	**19.03(0.80)**
IDSPSO	E_O	8.94(0.28)	11.98(0.27)	10.93(0.39)	17.06(0.53)	19.10(0.97)	21.39(1.02)	21.42(0.95)	24.61(1.22)
	E_B	6.27(0.25)	7.99(0.25)	8.46(0.36)	13.10(0.47)	16.11(0.88)	**17.79(0.88)**	18.65(0.88)	21.53(1.10)
mCMA-ES	E_O	5.65(0.22)	6.53(0.07)	7.08(0.26)	11.06(0.33)	18.75(0.7)	20.23(1.01)	21.39(0.91)	24.06(1.08)
	E_B	3.43(0.23)	3.46(0.06)	5.25(0.25)	8.56(0.31)	15.95(0.63)	16.77(0.87)	19.27(0.86)	21.54(0.99)
mbDE	E_O	8.33(0.30)	13.59(0.24)	14.50(0.44)	24.97(0.56)	23.01(1.01)	27.85(1.53)	25.33(1.14)	30.14(1.37)
	E_B	5.38(0.25)	8.47(0.21)	11.86(0.42)	19.74(0.50)	19.64(0.89)	23.67(1.34)	21.92(10.00)	25.94(1.20)
mjDE	E_O	12.02(0.30)	23.64(0.97)	13.00(0.33)	30.58(1.62)	18.94(0.86)	27.18(1.72)	19.78(0.89)	27.23(1.33)
	E_B	9.57(0.27)	19.15(0.92)	10.41(0.31)	25.54(1.53)	15.47(0.78)	22.95(1.54)	16.45(0.82)	23.00(1.15)
mPSO	E_O	6.70(0.23)	11.50(0.19)	10.12(0.30)	17.67(0.34)	20.30(0.89)	25.31(1.33)	21.74(0.89)	25.85(1.27)
	E_B	4.20(0.21)	7.00(0.14)	7.51(0.24)	12.68(0.27)	16.96(0.81)	20.81(1.11)	18.22(0.78)	21.65(1.08)
mQSO	E_O	10.84(0.33)	15.60(0.23)	13.94(0.40)	22.48(0.68)	19.88(0.76)	24.30(1.23)	21.92(1.00)	25.42(1.23)
	E_B	8.63(0.27)	11.87(0.22)	11.90(0.38)	18.57(0.64)	17.00(0.69)	20.38(1.08)	19.14(0.93)	21.77(1.07)
RPSO	E_O	23.67(0.85)	28.38(0.70)	26.06(0.92)	29.50(0.78)	23.72(1.26)	28.66(1.67)	23.84(1.15)	28.19(1.38)
	E_B	18.63(0.73)	20.90(0.65)	20.72(0.77)	21.72(0.76)	17.12(0.99)	20.39(1.34)	17.35(0.90)	20.12(1.08)

and δ controls the size of feasible regions. (4) is used as $C(\cdot)$ in [37]. In fact, feasible regions are defined using the constructed components by $C(\cdot)$ whose sizes are defined according to the values of δ and the gradients of each component. Therefore, for a component i in $C(\cdot)$ whose height is larger than δ, all solutions around its summit whose calculated fitness values by $C(\cdot)$ are larger than δ, form a feasible region. Depending on the chosen multi-component function as $C(\cdot)$,

TABLE S-VII
OBTAINED RESULTS (MEAN AND STANDARD ERROR BY USING E_D) BY DOP ALGORITHM ON EIGHT GMPB SCENARIOS FROM TABLE II WITH THE DEFAULT SETTINGS FROM TABLE IV. THE BEST OBTAINED RESULTS ARE HIGHLIGHTED BASED ON THE FRIEDMAN TEST WITH $\alpha = 0.05$.

Algorithm	GMPB Scenario							
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
AmQSO	19.85(1.33)	21.69(1.35)	25.37(2.04)	28.39(1.80)	49.51(0.67)	49.40(0.80)	52.60(0.82)	54.28(0.91)
CPSO	36.91(1.02)	34.39(1.86)	38.90(1.16)	39.43(0.93)	52.32(0.54)	54.25(0.63)	55.67(0.61)	56.75(0.57)
DynPopDE	31.95(1.36)	35.11(1.56)	49.55(1.13)	48.05(1.43)	53.77(0.91)	54.47(0.76)	58.26(0.62)	58.37(0.53)
FTmPSO	18.02(1.53)	19.80(1.30)	22.77(1.24)	24.07(0.98)	46.03(0.81)	45.88(0.84)	48.28(0.69)	49.79(0.84)
IDSPSO	22.66(1.09)	25.47(1.40)	37.61(1.94)	41.08(1.68)	69.01(1.34)	69.93(1.42)	76.06(1.54)	79.58(0.83)
mCMA-ES	20.43(1.24)	20.74(1.73)	25.78(1.60)	25.95(1.60)	55.71(0.98)	56.29(0.91)	59.81(0.86)	59.42(0.73)
mbDE	19.69(1.34)	20.83(1.60)	31.01(1.19)	32.56(0.58)	48.28(0.92)	49.11(0.72)	52.21(0.67)	54.04(0.78)
mjDE	28.92(1.48)	29.95(1.07)	33.62(1.30)	35.04(1.67)	44.14(0.70)	46.05(0.72)	48.02(0.71)	49.07(0.93)
mPSO	19.16(1.20)	22.50(1.35)	24.62(1.79)	26.31(1.39)	49.83(0.81)	49.31(0.76)	53.64(0.98)	55.60(0.66)
mQSO	20.29(1.40)	24.25(1.58)	28.88(1.19)	30.92(1.74)	57.56(0.96)	58.43(1.20)	60.53(0.71)	63.60(1.10)
RPSO	57.35(0.96)	62.28(1.90)	59.67(0.78)	60.52(1.72)	60.30(0.79)	60.63(0.74)	61.65(0.63)	60.13(0.68)

the size, location and shape of feasible regions change over time. In addition, if the value of δ is set to the range of $[h_{\min}, h_{\max}]$ of the components of $C(\cdot)$, the number of feasible regions can change over time. GMPB can be embedded in the proposed benchmark framework in [37] as the objective function and $C(\cdot)$ in (S-11).

In [38], another multi-component DCOP benchmark generator is proposed. In this benchmark generator, the baseline from (4) is used as the objective function subjected to:

$$g_j^{(t)}(\mathbf{x}) = \sum_{k=1}^{d} \left(x_k - c_{a_j,k}^{(t)} \right)^2 - r_j^{(t)^2} \leq 0,\ j = 1, 2, \cdots, \mathfrak{m}, \tag{S-12}$$

where $\mathfrak{m}$ is the number of feasible regions, r_j is the radius of jth feasible region, d is the dimension of the problem, and $\mathbf{c}_{a_j}$ is a center of a component in the objective function which is used as the center of the feasible region as well. Consequently, the feasible regions are located around the components' summits of the objective function. Therefore, their locations change by relocating components' centers. Moreover, the feasible regions can switch their locations to other components' summits in the landscape after environmental changes. In addition, the radius of feasible regions, which are hyper-balls, can change over time. Similar to the proposed benchmark in [36], the feasibility of a solution is determined by (S-10) in this benchmark. This type of feasible regions around the optima can be easily added to GMPB.

In [39], a DCOP benchmark framework based on modified multi-component functions is proposed. The embedded multi-component function needs to be modified based on the 'field of components on a zero plane'. To this end, the function $f(\cdot)$ is modified to $\max(0, f(\cdot))$. This modification is done in order to avoid any circumstance where there is no feasible solution. Note that this idea is inspired by (10) with $\beta = 0$. In this benchmark, the problem is defined as:

$$h_j^{(t)}(\mathbf{x}) = f_j^{(t)}(\mathbf{x}) - g_j^{(t)}(\mathbf{x}), \tag{S-13}$$

where $f(\cdot)$ and $g(\cdot)$ are two multi-component based functions that are modified to the field of components on a zero plane. A solution $\mathbf{x}$ is feasible in the tth environment if $h_j^{(t)}(\mathbf{x}) \geq 0$. By controlling the parameters of the $f(\cdot)$ and $g(\cdot)$, the feasible regions can be controlled. Therefore, there are multiple feasible regions whose sizes, shapes, and locations change over time due to the environmental changes in $f(\cdot)$ and $g(\cdot)$. GMPB can be used in this framework to build DCOP benchmark problems.

REFERENCES

[1] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Information Sciences*, vol. 316, pp. 419–436, 2015.

[2] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Transactions on Mathematical Software*, vol. 42, no. 2, p. 13, 2016.

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

[4] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '11. Association for Computing Machinery, 2011, pp. 1115–1122.

[5] M. Yang, M. N. Omidvar, C. Li, X. Li, Z. Cai, B. Kazimipour, and X. Yao, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 493–505, 2017.

[6] B. Kazimipour, M. N. Omidvar, A. Qin, X. Li, and X. Yao, "Bandit-based cooperative coevolution for tackling contribution imbalance in large-scale optimization problems," *Applied Soft Computing*, vol. 76, pp. 265–281, 2019.

[7] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," *IEEE Transaction on Evolutionary Computation*, 2019.

[8] F. Wang, P. Bahri, P. L. Lee, and I. Cameron, "A multiple model, state feedback strategy for robust control of non-linear processes," *Computers & Chemical Engineering*, vol. 31, no. 5-6, pp. 410–418, 2007.

[9] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2010.

[10] R. Kannan, S. Hendry, N. Higham, and F. Tisseur, "Detecting the causes of ill-conditioning in structural finite element models," The University of Manchester, Tech. Rep., 2013.

[11] J. Branke and H. Schmeck, “Designing evolutionary algorithms for dynamic optimization problems,” in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. Springer Natural Computing Series, 2003, pp. 239–262.

[12] T. T. Nguyen, S. Yang, and J. Branke, “Evolutionary dynamic optimization: A survey of the state of the art,” *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[13] K. Weicker and N. Weicker, “On evolution strategy optimization in dynamic environments,” in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3. IEEE, 1999, pp. 2039–2046.

[14] J. G. O. L. Duhain, “Particle swarm optimisation in dynamically changing environments - an empirical study,” Master’s thesis, University of Pretoria, Pretoria, South Africa, 2012.

[15] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *International Conference on Neural Networks*, vol. 04, 1995, pp. 1942–1948.

[16] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[17] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, “Dynamic optimization using self-adaptive differential evolution,” in *Congress on Evolutionary Computation*, 2009, pp. 415–422.

[18] R. Mendes and A. S. Mohais, “DynDE: a differential evolution for dynamic optimization problems,” in *Congress on Evolutionary Computation*, vol. 3, 2005, pp. 2808–2815.

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

[20] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[21] R. Eberhart and Y. Shi, “Comparing inertia weights and constriction factors in particle swarm optimization,” in *Congress on Evolutionary Computation*, vol. 1, 2001, pp. 84–88.

[22] N. Hansen, “The cma evolution strategy: A tutorial,” INRIA, Tech. Rep. arXiv:1604.00772v1, 2016.

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

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

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

[26] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, “A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization,” *Applied Soft Computing*, vol. 13, no. 04, pp. 2144–2158, 2013.

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

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

[29] X. Hu and R. C. Eberhart, “Adaptive particle swarm optimization: detection and response to dynamic systems,” in *Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1666–1670.

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

[31] T. T. Nguyen, “Continuous dynamic optimisation using evolutionary algorithms,” Ph.D. dissertation, University of Birmingham, 2011.

[32] D. Yazdani, T. T. Nguyen, and J. Branke, “Robust optimization over time by learning problem space characteristics,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 01, pp. 143–155, 2019.

[33] C. Liu, “New dynamic constrained optimization pso algorithm,” in *International Conference on Natural Computation*, vol. 7. IEEE, 2008, pp. 650–653.

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

[35] Z. Zhang, S. Yue, M. Liao, and F. Long, “Danger theory based artificial immune system solving dynamic constrained single-objective optimization,” *Soft Computing*, vol. 18, no. 1, pp. 185–206, 2014.

[36] H. Richter, “Memory design for constrained dynamic optimization problems,” in *Applications of Evolutionary Computation*, C. Di Chio et al., Ed. Springer Berlin Heidelberg, 2010, pp. 552–561.

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

[38] Y. Wang, J. Yu, S. Yang, S. Jiang, and S. Zhao, “Evolutionary dynamic constrained optimization: Test suite construction and algorithm comparisons,” *Swarm and Evolutionary Computation*, vol. 50, p. 100559, 2019.

[39] G. Pamparà and A. P. Engelbrecht, “A generator for dynamically constrained optimization problems,” in *Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, 2019, p. 1441–1448.