An Open Framework for Constructing Continuous Optimization Problems

Changhe Li, *Member, IEEE*, Trung Thanh Nguyen, Sanyou Zeng, Ming Yang, Min Wu, *Senior Member, IEEE*

***Abstract*—Many artificial benchmark problems have been proposed for different kinds of continuous optimization, e.g., global optimization, multi-modal optimization, multi-objective optimization, dynamic optimization, and constrained optimization. However, there is no unified framework for constructing these types of problems and possible properties of many problems are not fully tunable. This will cause difficulties for researchers to analyze strengths and weaknesses of an algorithm. To address these issues, this paper proposes a simple and intuitive framework, which is able to construct different kinds of problems for continuous optimization. The framework utilizes the k-d tree to partition the search space and sets a certain number of simple functions in each subspace. The framework is implemented into global/multi-modal optimization, dynamic single objective optimization, multi-objective optimization, and dynamic multi-objective optimization, respectively. Properties of the proposed framework are discussed and verified with traditional evolutionary algorithms.**

***Index Terms*—Continuous optimization, global optimization, multi-modal optimization, multi-objective optimization, dynamic optimization, free peaks.**

I. Introduction

A general continuous optimization problem can be mathematically defined as follows:

$$\begin{array}{ll} maximize & \mathbf{f}(\mathbf{x},t) = \{f_1(\mathbf{x},t),\ldots,f_O(\mathbf{x},t)\} \\ subject\ to & g_i(\mathbf{x}) \leq 0, i = 1,\ldots,I \\ & h_i(\mathbf{x}) = 0, i = 1,\ldots,E \end{array} \quad (1)$$

where $\mathbf{x} = (x_1,\ldots,x_D) \in \mathcal{R}^D$ is a decision vector of D dimensions in continuous space, x_i is within a boundary $l_i \leq x_i \leq u_i$ $(1 \leq i \leq D)$, t is real-world time, and $\mathbf{f}$, $\mathbf{g}$, and $\mathbf{h}$ are objective functions, inequality constraints and equality constraints, respectively. In this paper, we aim to provide a method for constructing artificial problems by taking properties of real-world problems into account, such as non-linear, multi-modal, discontinuous, multi-objective, dynamic, and constrained properties.

Manuscript received April 7, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61673355, in part by the Hubei Provincial Natural Science Foundation of China under Grant 2015CFA010, in part by the 111 project under Grant B17040, in part by the British Council under a Newton Institutional Links grant, and in part by the Royal Academy of Engineering under the Newton Research Collaboration Programme (3).

C. Li and M. Wu are with the School of Automation and also with the Hubei key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, China University of Geosciences, Wuhan 430074, China (email: changhe.lw@gmail.com; wumin@cug.edu.cn).

T. T. Nguyen is with the School of Engineering, Technology and Maritime Operations, Liverpool John Moores University, Liverpool L3 3AF, U. K. (e-mail: T.T.Nguyen@ljmu.ac.uk).

M. Yang is with the School of Computer Science, China University of Geosciences, Wuhan 430074, China (email: yangming0702@gmail.com).

S. Zeng is with the School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan 430074, China (email: sanyouzeng@gmail.com).

In the literature of continuous evolutionary optimization, there exist several types of optimization problems, e.g., global optimization problems (GOPs), multi-modal optimization problems (MMOPs), dynamic single objective optimization problems (DSOOPs), multi-objective optimization problems (MOOPs), and dynamic multi-objective optimization problems (DMOOPs), and many benchmark problems have been developed in each type. These problems have been widely employed to test and compare an algorithm's performance [11], [32], [63]. However, for benchmarking problems we still believe that several aspects should be enhanced in terms of the convenience for researchers to carry out experimental studies.

First, there is no framework that comprises different classes of problems. Existing frameworks are designed only for one specific class. This makes it difficult to 1) test algorithms that are able to solve multiple types of problems such as [51] and 2) transfer the problem difficulties from one type of problem to another, e.g., from SOOPs to MOOPs. We believe it would be useful to have a unified open-source framework, which specifies common terminating criteria, scale, multimodality, ruggedness etc. for different types of problems.

Second, the function expressions of some benchmark problems, like the composition test functions series from the IEEE CEC [56] or the MOOP benchmarks in [8], [33], [65], do not offer good indications on what are the geometric properties of the functions. For example, by looking at the problem expressions, it is hard to know the properties of the global structure, the number of local optima, locations of local optima, the relationship between the number of optima and dimension (for GOPs); or the shape of the Pareto optima front, and the distribution of the Pareto optimal set (for MOOPs).

Third, in some popular benchmark problems [17], [56], the problem properties are not fully tunable or controllable or the properties are non-separable, meaning that tuning one property would affect other properties. This would make it difficult to analyse the impact of one single property on an algorithm's behaviour. For example, increasing the number of dimensions will exponentially increase the number of optima for most multi-modal benchmark problems. This would be difficult for researchers to analyze the multi-modal difficulty for their algorithms on problems with different dimensions.

Therefore, we would like to propose an intuitive framework that can enhance these aspects. This paper proposes a unified framework for generating different types of continuous optimization problems. The framework is designed to be simple and intuitive yet highly configurable. The following features are taken into account.

- The framework should be easy to understand and analyze

in the following aspects: the position of an optimum, the shape of an optimum, the basin of attraction of an optimum, the local structure of an optimum, the shape of the Pareto optimal front (POF), the distribution of the Pareto optimal set (POS), and the dominating relationship between any solutions.
- The framework should be independently configurable in the following aspects: the dimensionality, the modality, the number of objectives, the position of an optimum, the shape of an optimum, the size of the basin of attraction of an optimum, the local/global structure, the shape of the POF, the distribution of the POS, the size of a countable POS, the inter-relationship between decision variables, the size of feasible areas, and domino convergence [58].
- The framework should be computationally efficient for fitness evaluations in comparison with existing problems.
- The framework should be able to show common characteristics of existing problems, e.g., the common shapes of the POF.
- The framework should be flexible and extendable, i.e., users are able to easily add new components to the framework.
- The framework should be able to easily manipulate any optimum with different transformations, e.g., rotation, shift, irregularities, and break of the symmetry of symmetric peaks.
- The framework should be compatible with existing problems. Existing problems can be integrated into the framework without changing their structures.

To construct a framework that has all the features mentioned above, this research utilizes the k-d tree [1] to partition the solution space into subspaces, then in each subspace proper functions are set depending on desired features. The framework is named Free Peaks (FPs). Four types of optimization problems are constructed: GOPs/MMOPs, DSOOPs, MOOPs, and DMOOPs. Properties of these problems are analyzed with the support of experiments.

The rest of this paper is organized as follows. Sect. II briefly reviews related work. The details of the FPs are gigen in Sect. III. Sect. IV presents the methods for generating different types of problems. Sect.V carries out experimental studies. Finally, conclusions are given in Sect. VI.

II. Related Work

This section briefly reviews related work to benchmarking problems and discusses some issues.

1) Global/Multi-Modal Optimization Problems: Due to the simplicity of traditional benchmarks, many researchers proposed complicated versions based on traditional benchmarks. There are mainly three types of modifications. The first type is the composition of benchmarks with different properties in the form $F(\mathbf{x}) = \max / \min(f_i(\mathbf{x}))$. In [36], Liang *et al*. proposed a composition test suite, where each problem is composed of ten functions selected from a basic set of five traditional functions. Gallagher *et al*. proposed a parameterized landscape generator [10], where a set of Gaussian functions are used to form peaks. The generator [10] was recently extended with a linear ridge structure [42]. Two simple problems [37], namely "double-Sphere" and "double-Rastrigin", were proposed to study the impact of the global structure on the behavior of evolutionary search.

Recently, Qu *et al*. proposed a set of benchmarks for multi-modal optimization, where seven of them were constructed based on sixteen traditional functions using the idea of [36]. This kind of benchmarks have been widely used in the related IEEE CEC competitions in recent years [56].

The second type is the application of different kinds of transformations to traditional benchmarks, e.g., the orthogonal rotation matrix for rotating the fitness landscape randomly around various axes [17], [36], [56], the shift vectors for shifting the global optima, and the diagonal matrix for creating ill-conditioned fitness landscape [17]. The rotation matrix and shift vector were also used to create difficult MOOPs in [23]. These transformations make benchmark functions much more difficult than the original versions. An early study [50] showed that a rotation of the coordinate system causes a severe performance loss to genetic algorithms (GAs). Non-separability and asymmetry issues were highlighted in [62] for the design of test suites. The BBOB test suite proposed by Hansen *et al*. [17] collects 24 noise free single-objective benchmark functions. These benchmarks have also been widely used for evaluating the performance of algorithms regarding several typical aforementioned difficulties, such as ill-conditioning, irregularities, non-separability, asymmetry, irregularity, and deceptiveness, etc. The difficulties of different features were studied on several existing EAs [16], particularly the ill-conditioned and non-separable features.

The third type is modifications based on the structure of existing functions, e.g., the Hump and Common families problems in a tunable test suite [48]. In [47], four types of interactions between components, namely, the fully separable, partially separable, overlapping, and fully non-separable, were suggested for the buildup of interactions between variable components for large scale optimization.

Besides the above three types of modifications based on existing functions, several new problems have also been proposed. A HappyCat problem with special properties of the fitness landscape structure near the global optimum was proposed in [2]. Genetic programming was used to evolve problem landscapes [30] to analyze the performance of existing EAs. Thereafter, the test suite [30] was extended to MOOPs using the combination of three types of real-coded crossovers [53].

2) Dynamic Single Objective Optimization Problems: For DSOOPs, researchers are interested in characteristics/types of changes, such as the predictability–whether changes are predictable in a regular pattern (e.g., recurrent changes in the objective value and/or location of an optimum, the location of an optimum moves on a certain path), time-linkage–whether future changes depend on the current/previous solutions found by optimizers [3], [45], detectability–whether changes are detectable, severity– determines the magnitude of a change (e.g., the distance of the location of an optimum moves, the objective value of an optimum changes), and change factors (objective functions, the number of dimensions, constraints, domain of

the search space, and function parameters). A comprehensive review of characteristics of existing DSOOPs can be found in [44]. In this paper, we review existing benchmarks from the way of the construction point of view as follows.

One of the most popular benchmarks is the MPB [4]. The MPB consists of a set of peaks, which change in height, width, and location. The fitness landscape is constructed by

$$F(\mathbf{x},t)=\max_{i=1,\ldots,N}\frac{H_i(t)}{1+W_i(t)\sum_{j=1}^{D}(x_j(t)-X_{ij}(t))^2}, \quad (2)$$

where $W_i(t)$ and $H_i(t)$ are the height and width of peak i at time t, respectively, and $X_{ij}(t)$ is the j-th element of the location of peak i at time t. The DF1 [43] and the rotation DBG (RDBG) [31] used a similar way to construct dynamic environments. However, their mechanisms of generating the dynamism are different from the MPB. The DF1 used the logistical function to generate dynamism, and the RDBG rotates the coordinates to generate dynamism. A challenging dynamic landscape (composition DBG, CDBG) [31] was constructed, where a set of composition functions are shifted in the fitness landscape.

3) Multi-Objective Optimization Problems: The aforementioned difficulties for single objective optimization problems (SOOPs) still apply to multi-objective optimization problems [7], [24]. In addition, for MOOPs researchers may be more interested in the properties of the POS and POF. Many benchmarks have been proposed to construct the POS and POF with different properties.

Deb stated that several features of the POF are hard for optimizers to maintain the diversity [7], such as non-linearity, discontinuity, and non-uniform distribution. In [7], Deb proposed a two-objective test problem toolkit, in which three functions with different purposes are involved. A distribution function f_1 is used to test the performance of an algorithm on maintaining diversity on the POF. A distance g is used to test the convergence performance to the POF. A shape function h determines the shape of the POF.

Based on Deb's toolkit [7], Zitzler *et al*. proposed a test suite (ZDT) [65]. The ZDT test suite is very popular due to its simplicity. However, both test suites are not scalable in terms of the number of objectives. Meanwhile, many-objective optimization problems (problems with more than three objectives) have attracted many researchers. Therefore, Deb *et al.* proposed the DTLZ test suite in [8] with a scalable number of objectives, which is relatively fixed with the number of dimensions. Due to the simplicity of the ZDT and DTLZ, the OKA test suite was proposed in [46] with complicated properties. However, the OKA test suite only has two objectives and two dimensions. To construct a test suite with a scalable number of dimensions, Li and Zhang proposed nine problems with complicated POS shapes [33]. Like the ZDT, all of these benchmark problems are not scalable with the number of objectives. A test suite called SYM-PART was proposed with a controllable number of POSs in [49]. In [24], a test suite called WFG was proposed with several difficult features, such as flat regions, variable dependency, multi-modal, and deceptiveness.

Besides the above test suites, researchers are also interested in problems that can be visually examined for many-objective optimization. A distance minimization test suite was proposed where each vertex (an objective) of a polygon is minimized in the decision space [25]. A rectangle problem was proposed in [35] to help the visual investigation of many-objective search, where the POS lies in a rectangle in a two-dimensional decision space. The problem [35] was recently updated in [34].

4) Dynamic Multi-Objective Optimization Problems: For DMOOPs, benchmarks with various hard features as mentioned for MOOPs and various types of changes are needed. In addition to the characteristics of changes of DSOOPs, researchers are also interested in characteristics of changes in the POS and POF for DMOOPs. Changes for DMOOPs were categorized into four types in [9], namely:

- Type I, the POS changes but the POF does not change.
- Type II, both the POS and the POF change.
- Type III, the POF changes but the POS does not.
- Type IV, neither the POS nor the POF changes but environmental changes occur.

A mixed type was recently proposed in [26]. Goh proposed another way to categorize DMOOPs in [12], where DMOOPs were classified based on spatial and temporal features, which are closely related to the change types of DSOOPs.

Several test suites [9], [18], [60] were proposed based on the ZDT [65] and DTLZ [8] test suites. Among these test suites, the one, named FDA, proposed by Farina *et al.* is a popular suite, and two other test suites [40], [64] were proposed based on it. In addition, there exist several other test suites [14], [27], [22], [29]. Considering the characteristics of these test suites as well as the features of several MOOPs [24], [19], [33], recently Helbig and Engelbrecht [20] proposed a test suite with three hard properties: deceptiveness, isolated POF, and complicated POS. Thereafter, Helbig and Engelbrecht introduced a test suite of problems collected from [9], [18], [19], [20], [29], [13] on a special session and a competition on IEEE CEC15 [21].

5) Summary and Discussions: The aforementioned studies focus on developing benchmark problems, there are also many studies that focus on developing methods for analyzing existing benchmark problems based on features of the fitness landscape. A length scale was introduced to measure the ratio of changes in the objective function value to steps between points in the search space [41]. A random increasing walk method was proposed to measure the ruggedness of the fitness landscape [38]. A nearest-better clustering method was proposed to detect if a problem is funnel or random [28]. A notion of evolvability was introduced based on the fitness distribution of the offspring of sampled solutions to describe features of the fitness landscape [54]. A recent survey of existing techniques for characterising continuous problems can be found in [39].

Many benchmark problems have been developed for each type of numerical optimization. However, we think some of them have disadvantages. Firstly, they are not intuitive in terms of geometric properties. For example, it is very hard to comprehend the geometric properties of Liang's composition test suite. In the ZDT [65] and DTLZ [8] suites, objective functions need external or intermediate functions as input parameters, which also causes difficulties to imagine the POF

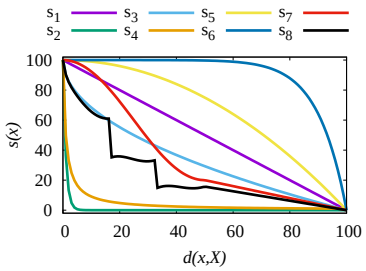


Fig. 1. Shapes of the eight functions with D=1, where h=100, r=50, η=5.5, and m=3 (objective values of all functions are standardized within $[0, 100]$.

and the POS of these test suites. Secondly, properties are not tunable or separately tunable, which causes two difficulties: 1) it is hard to investigate how each property influences the behavior of an algorithm and 2) it is not easy to choose which problems to use for experimental studies since there are many problems and they may share common properties. Therefore, we would like to propose a fully tunable and highly intuitive generator, which is able to generate different kinds of numerical optimization problems.

III. Free Peaks

This section gives the detailed descriptions of our Free Peaks framework. Without any loss of generality, maximization optimization problems are considered in this paper.

A. One Peak Problems

Before constructing the problem, we need to prepare a set of simple problems and a way to partition the solution space. Eight simple symmetrical unimodal functions are constructed as follows:

$$
\begin{aligned}
s_1(\mathbf{x}; h, \mathbf{X}) &= h - d(\mathbf{x}, \mathbf{X}),\\
s_2(\mathbf{x}; h, \mathbf{X}) &= h\exp(-d(\mathbf{x}, \mathbf{X})),\\
s_3(\mathbf{x}; h, \mathbf{X}) &= h - \sqrt{h \cdot d(\mathbf{x}, \mathbf{X})},\\
s_4(\mathbf{x}; h, \mathbf{X}) &= h/(1 + d(\mathbf{x}, \mathbf{X})),\\
s_5(\mathbf{x}; h, \mathbf{X}) &= h - d^2(\mathbf{x}, \mathbf{X})/h,\\
s_6(\mathbf{x}; h, \mathbf{X}) &= h - \exp(2\sqrt{d(\mathbf{x}, \mathbf{X})/\sqrt{D}}) + 1,\\
s_7(\mathbf{x}; h, \mathbf{X}, r) &= \begin{cases} h\cos(\pi d(\mathbf{x}, \mathbf{X})/r) & d(\mathbf{x}, \mathbf{X}) \le r \\ -h - d(\mathbf{x}, \mathbf{X}) + r & d(\mathbf{x}, \mathbf{X}) > r \end{cases},\\
s_8(\mathbf{x}; h, \mathbf{X}, r) &= \begin{cases} \frac{h(cos(m\pi\frac{d(\mathbf{x},\mathbf{X})-mnr}{r})-\eta n)}{\sqrt{d(\mathbf{x},\mathbf{X})+1}} & d(\mathbf{x}, \mathbf{X}) \le r \\ h - \eta m\sqrt{r+1} - d(\mathbf{x}, \mathbf{X}) + r & d(\mathbf{x}, \mathbf{X}) > r, \end{cases}
\end{aligned}
$$

where $d(\mathbf{x}, \mathbf{X}) = \sqrt{\sum_i^D (x_i - X_i)^2}$ is the *Euclidean* distance from $\mathbf{x}$ to the peak, which is located at a location $\mathbf{X}^{s_v} = \mathbf{0}$ with a height of $h > 0$ for each shape function s_v $(v = 1, \ldots, 8)$ in the solution space with D dimensions, r is a parameter of value in $[0, \Omega]$ $(\Omega = \sqrt{\sum_i^D (u_i^{s_v} - l_i^{s_v})^2}, u_i^{s_v} = 100, l_i^{s_v} = -100)$, m and η determine the number of segments and the gap between two neighbor segments of s_8, respectively, and n=$\lfloor md(\mathbf{x}, \mathbf{X})/r \rfloor$. The default values of $\mathbf{X}^{s_v}$=$\mathbf{0}$,h=100,r=50,m=3, and η=5.5 are used in this paper.

Fig 1 shows the shapes of the eight functions. As shown in the figure, we have a linear function s_1, three convex functions s_2-s_4, two concave functions s_5 and s_6, a disconnected function s_8, and a hybrid function s_7 which is partially convex, partially concave, and partially linear (see Appendix B for more shapes). Every function is symmetrical at its peak.

B. Additional Properties

To make the one-peak problems with enriched properties, several groups of transformations are introduced.

1) Group One – Shift of location $\mathbf{X}$*:*

- *T1: Random shift.* $\mathbf{X}$ is shifted by

$$X_i = \mathrm{R}(l_i, u_i), 1 \le i \le D, \tag{4}$$

  where $\mathrm{R}(a, b)$ returns a random number of the uniform distribution within $[a, b]$, l_i and u_i are the lower and upper boundary of the ith dimension, respectively.
- *T2: Corner shift.* The location of $\mathbf{X}$ is shifted to a corner of the decision space by

$$X_i = l_i, 1 \le i \le D, \tag{5}$$

  Note that, the corner shift may be easy for some EAs if they set out-of-range components of a solution to boundary values.

2) Group Two – Redefinition of distance :

- *T3: Setup of dependencies.* Variables are randomly put into K $(1 \le K < D)$ groups $(\mathbf{I}_j, j \in [1, K]$, e.g., $\mathbf{I}_1$=$\{4, 8, 1\}$, $\mathbf{I}_2$=$\{3, 5\}, \ldots)$. Variables in one group interact with each other as described in Eq. (6) and variables in different groups have no dependence. The dependence of variables in a group I_j is set by

$$d_1(\mathbf{x}, \mathbf{X}) = \sqrt{\sum_{j=1}^{K} (\sum_{i=1}^{|I_j|} (\sum_{k=1}^{i} (x_{I_{j,k}} - X_{I_{j,k}}))^2)}. \tag{6}$$

  where $I_{j,k}$ is the kth element of group I_j. Note that, the problem is fully non-separable if K=1; otherwise, the problem is partially separable. The higher the degree of non-separability, the harder a function will normally become [57]. Although T3 will introduce ill-conditioning as T6 introduced later, we still keep it in order to study the effect of varying the degree of dependencies.
- *T4: Setup of domino convergence.* The domino convergence [58] occurs when variables have contributions of a significantly different degree to the objective value. Domino convergence could cause difficulties for optimizers because domino variables would dominate the search of an algorithm. As a result, the other variables may be ignored regardless of whatever values they have. The domino convergence of variables is set by

$$d_2(\mathbf{x}, \mathbf{X}) = \sqrt{\sum_i^D w_i \cdot (x_i - X_i)^2}, \tag{7}$$

  where w_i is a random weight generated by $w_i = 10^{3\mathrm{N}(0,1)}$, $\mathrm{N}(0, 1)$ returns a random number of the standard normal distribution. Note that, T4 is simply to make variables ill-scaled, which is a specific form of T6 introduced later.
- *T5: Flat border:* To have the same objective value at the space boundary, the distance is reformed by

$$d_3(\mathbf{x}, \mathbf{X}) = (1 - \prod q_i)\Omega, q_i = \begin{cases} 1 - \frac{x_i - X_i}{u_i - X_i} & x_i \ge X_i \\ 1 - \frac{X_i - x_i}{X_i - l_i} & x_i < X_i \end{cases} \tag{8}$$

This property is to address the disconnection issue when a problem consists of multiple peaks, and it will be further discussed in Sect. IV-A1 later.

3) Group Three – Transformations in the decision space:

- *T6: Setup of ill-conditioning.* To have an ill-conditioned fitness landscape, a linear transformation matrix $\mathbf{R}$ is applied to $\mathbf{x}$

$$\mathbf{x} = \mathbf{R} \cdot \mathbf{x}, \tag{9}$$

where $\mathbf{R}$ is obtained by $\mathbf{R}$=$\mathbf{QNP}$, $\mathbf{Q}$ and $\mathbf{P}$ are orthogonal matrixes generated by the classical Gram-Schmidt method, $\mathbf{N}$ is a diagonal matrix with a default condition number of 1000.

- *T7: Setup of irregularities.* To generate irregularities, the transformation [17] below is applied to x_i

$$x_i = sign(x_i) exp(\tilde{x}_i + 0.049(\sin(c_1\tilde{x}_i) + sin(c_2\tilde{x}_i))), \tag{10}$$

where $\tilde{x}_i = \begin{cases} log(|x_i|) & x_i \neq 0 \\ 0 & x_i = 0 \end{cases}$, $sign(x_i) = \begin{cases} -1 & x_i < 0 \\ 0 & x_i = 0 \\ 1 & x_i > 0 \end{cases}$,

$c_1 = \begin{cases} 10 & x_i > 0 \\ 5.5 & x_i \leq 0 \end{cases}$, $c_2 = \begin{cases} 7.9 & x_i > 0 \\ 3.1 & x_i \leq 0. \end{cases}$

- *T8: Setup of asymmetry.* To break the symmetry of a symmetric function, the following transformation [17] is applied to x_i

$$x_i = \begin{cases} x_i^{1+0.2\frac{i-1}{D-1}\sqrt{x_i}} & x_i > 0 \\ x_i & x_i \leq 0. \end{cases} \tag{11}$$

4) Group Four – Transformations in the objective space:

- *T9:* Noise is added to the objective value by

$$s(\mathbf{x}) = s(\mathbf{x}) + |\mathrm{N}(0,1)|. \tag{12}$$

- *T10:* Mapping the objective value in a range $[\underline{h}, h]$ by

$$s(\mathbf{x}) = \underline{h} + (h - \underline{h})\frac{s(\mathbf{x}) - s(\underline{x})}{h - s(\underline{x})}, \tag{13}$$

where $\underline{h} \in (0, h)$ is a user given value, $\underline{x}$ is a solution that has the lowest object function's value. Note that, this is a linear mapping, non-linear mapping can also be applied.

5) Group Five — Constraints:

- The peak is within the feasible area.

$$g_1(\mathbf{x}) = d(\mathbf{x}) - r_{g_1}, \tag{14}$$

where $r_{g_1} > 0$ is a parameter with a default value of 50.

- The peak is at the border of the feasible area.

$$g_2(\mathbf{x}) = \sqrt{\sum_i^D (x_i - X_i^c)^2} - \sqrt{\sum_i^D (X_i - X_i^c)^2}, \tag{15}$$

where $X_i^c = \begin{cases} X_i & i = j|\underset{j}{\operatorname{argmin}}|X_j - l_j| \\ l_i & else \end{cases}$.

- The problem has multiple feasible areas.

$$g_3(\mathbf{x}) = |sin(log(1 + d^3(\mathbf{x})))| - r_{g_3}, \tag{16}$$

where $0 < r_{g_3} < 1$ is a parameter with a default value of 0.5.

Fig. 2 shows the effect of the transformations on function s_1 with D=2, where Fig. 2-(a) is the original landscape of s_1, Fig. 2-(b)-(i) are the landscapes of s_1 after using transformations T1-T8, respectively, and Fig. 2-(j)-(l) are the landscapes of s_1 after adding the constraints in Eqs.(14)-(16), respectively. In Fig. 2-(j)-(l), the black areas are infeasible areas. Note that, the aforementioned transformations and constraints can be mixed together to make the fitness landscape more complex except for the transformations in groups one and two, because these transformations are mutually exclusive with each other.

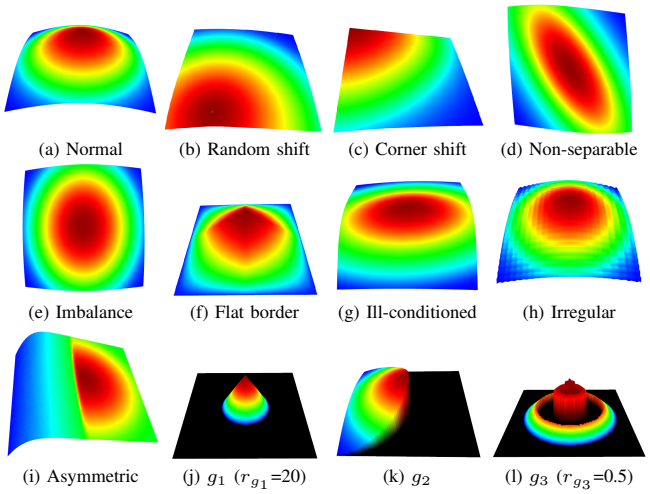


Fig. 2. Visual effect of different transformations on the fitness landscape of s_1 with D=2, where the black areas are infeasible areas.

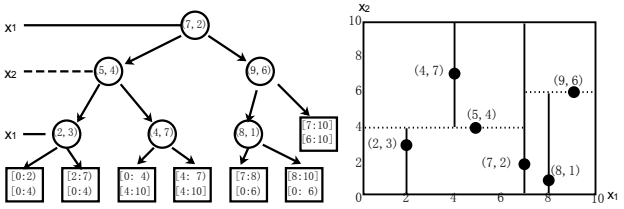


Fig. 3. An example of the k-d tree for the division of a 2-D space with ranges ([0:10],[0:10]) by a set of six points.

C. Partition the Search Space

The k-d tree [1] is a binary tree in which every node is a k-dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts. Points to the left of this hyperplane are represented by the left subtree of that node and points to the right of the hyperplane are represented by the right subtree. Every leaf node denotes a subspace of the solution space. Fig. 3 shows a k-d tree (Fig. 3-left) for the decomposition of a 2-D solution space (Fig. 3-right) with six points.

To construct a balanced tree, the canonical method [1] is used, where a median point is selected with the cutting axis. Algorithms 1 and 2 present the space partition process and the inquiry of a solution, respectively. Note that, for the range of a subspace, in each dimension it takes the close form for the lower bound and the open form for the upper bound (see subspace ([0:2),[0:4)) of the left child of node (2,3) in the left graph of Fig. 3). However, for a subspace containing the upper bound of the whole solution space, it takes the close form for both its lower and upper bounds (see subspace ([7:10],[6:10]) of the right child of node (9,6) in the left graph of Fig. 3).

In this paper, the solution space is divided by default into

Algorithm 1 kdtree($list$, $depth$)

```
axis ← depth%D + 1; ▷ Select a cutting plane axis based on depth
Select the median by axis from list
if ||list||=1 then                                   ▷ A leaf node
    Create a subspace;
else
    Create a node node with data of the median point;
    node.left ← kdtree(points in list before the median, depth+1);
    node.right← kdtree(points in list after the median, depth+1);
    return node;
end if
```

Algorithm 2 inquire($\mathbf{x}$, $node$,$depth$)

```
i ← depth%D + 1;
if node is a leaf node then return the subspace; end if
if x_i < node_i then
    inquire(x, node.left,depth+1);
else
    inquire(x, node.right,depth+1);
end if
```

N subspaces with random sizes. The solution space can be divided into subspaces with user-defined sizes (see the method in Appendix C).

D. Setup of Subspaces

After dividing the search space into N subspaces, we can set them as follows. An O-objective function $\mathbf{f}(\mathbf{x})$ with N subspaces can be defined by

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \mathbf{f}^{b_1}(\mathbf{x}) = \{f_1^{b_1}(\mathbf{x}), f_2^{b_1}(\mathbf{x}), \ldots, f_O^{b_1}(\mathbf{x})\}, \mathbf{x} \in [\mathbf{l}^{b_1}, \mathbf{u}^{b_1}] \\ \mathbf{f}^{b_2}(\mathbf{x}) = \{f_1^{b_2}(\mathbf{x}), f_2^{b_2}(\mathbf{x}), \ldots, f_O^{b_2}(\mathbf{x})\}, \mathbf{x} \in [\mathbf{l}^{b_2}, \mathbf{u}^{b_2}] \\ \ldots \\ \mathbf{f}^{b_N}(\mathbf{x}) = \{f_1^{b_N}(\mathbf{x}), f_2^{b_N}(\mathbf{x}), \ldots, f_O^{b_N}(\mathbf{x})\}, \mathbf{x} \in [\mathbf{l}^{b_N}, \mathbf{u}^{b_N}] \end{cases} \tag{17}$$

where each subspace b_k contains O component functions and each component function $f_j^{b_k}$ is associated with a shape function s_v (j=1, 2, ... O, k=1, 2, ... N, v=1, 2, ..., 8). The whole search space of $\mathbf{f}$ ($[\mathbf{l}, \mathbf{u}]$) is divided into N subspaces: $[\mathbf{l}, \mathbf{u}]$=$\{[\mathbf{l}^{b_k}, \mathbf{u}^{b_k}], \ldots\}$, k=1, 2, ... N.

To compute the jth objective of $\mathbf{x}$ ($f_j(\mathbf{x})$), we need to find the subspace b_k where $\mathbf{x}$ is (i.e., $\mathbf{l}^{b_k} \leq \mathbf{x} < \mathbf{u}^{b_k}$) by Algorithm 2, then map $\mathbf{x}$ to a solution $\mathbf{x}^{s_v}$ in the search space of s_v associated with f_j in subspace b_k by

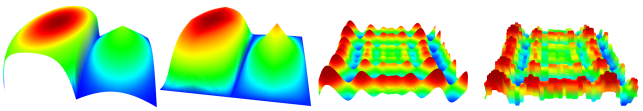
$$map(x_i) = x_i^{s_v} = l_i^{s_v} + (u_i^{s_v} - l_i^{s_v}) \frac{x_i - l_i^{b_k}}{u_i^{b_k} - l_i^{b_k}}, i = 1, 2, \ldots D, \tag{18}$$

Eventually, we set the objective $f_j(\mathbf{x})$ by

$$f_j(\mathbf{x}) = f_j^{b_k}(\mathbf{x}) = s_v(\mathbf{x}^{s_v}). \tag{19}$$

Note that, the fitness landscape of s_v will be stretched/squeezed if the aspect ratio of the subspace is different from that of the search space of s_v.

For single objective problems, i.e., O=1, the shape function of each subspace is not necessarily the same and any transformation introduced in Sec. III-B can be applied. For multi-objective problems, however, the shape function associated with a particular objective in all subspaces is normally the same and these transformations are not applied for simplicity. This would be easy to analyze the dominant relationships between any two solutions in the search space. The detailed setup of subspaces will be given later for each type of optimization problems.



(a) Without $T5$ (b) With $T5$ (c) Schwefel (d) Simulated Schwefel

Fig. 4. Visual effect of transformation $T5$ and a simulated fitness landscape, where graphs (a) and (b) consist of two peaks with shapes s_1 (the right peak) and s_6 (the left peak) in each graph and graph (d) is a simulated landscape of the Schwefel's landscape shown in graph (c).

E. Time Complexity

To evaluate a solution $\mathbf{x}$ we need to perform the following three steps: 1) to find out the subspace (b_k) where $\mathbf{x}$ is; 2) to map $\mathbf{x}$ to a location ($\mathbf{x}^{s_v}$) in the search space of f^{b_k}; 3) to compute the objective of ($\mathbf{x}^{s_v}$) by one of the eight shape functions. Identifying to which subspace a point belongs has a complexity of $O(log(N))$ by Algorithm 2. Both the second and the third steps run in $O(D)$. Therefore, the total time complexity of evaluating a solution is $O(log(N)) + 2O(D)$.

IV. Benchmark Problems

In this section, we will construct GOPs/MMOPs, DSOOPs, MOPs, and DMOOPs with the FPs framework.

A. Global/Multi-Modal Optimization Problems

Based on the FPs, we can construct a GOP or MMOP with independently manageable properties, such as the number of optima, the number of dimensions, the shape of each peak, the size of the basin of attraction of each peak, the height of each peak, and the location of each peak. Given these basic manageable properties, the FPs is capable of generating a fitness landscape which has enriched properties. Together with the transformations introduced in Sect.III-B, we are even able to construct a very complex landscape with fully independently configurable properties.

1) Stitching Borders: The objective values of solutions at the border of two neighbour subspaces may be different. This may cause disconnections at the borders of neighbour subspaces. This issue can be addressed by the transformation $T5$. Fig. 4-(b) shows the effect of $T5$ on a fitness landscape with two peaks. The transformation $T5$ stitches borders of neighbour subspaces.

2) Defining Global Structures: The whole fitness landscape may have a very weak global structure if the height of a peak is randomly set. To address this issue, we could simply set peak heights based on any existing problem. Fig. 4-(d) shows a simulated fitness landscape of the Schwefel function (Fig. 4-(c)) with 1,000 subspaces, where each subspace is associated with s_6. To achieve this, we set the height of a peak to the objective value of a solution, which maps to the peak location, in a traditional problem. Note that, the global optimum of the simulated problem would change if the location of the global optimum of the original problem is unknown.

The basin of attraction of a peak, i.e., the subspace where the peak is, is the main factor which impacts a gradient-based local search method. However, for an EC algorithm, the main factor,

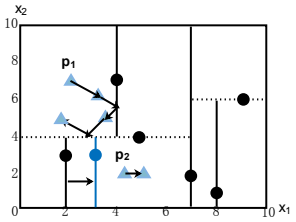


Fig. 5. Location change of a peak, where triangle points are peaks' locations and circle points are division points.

strictly speaking, would be the effective basin of attraction of an optimum. It is very hard to estimate the size of the effective basin of attraction of an optimum. In the FPs, for a peak, subspaces, which are around the peak and with peak heights monotonically decreases as the distance to the peak increases, are regarded as the effective basin of attraction of the peak.

B. Dynamic Single-Objective Optimization Problems

In this section, changes in DSOOPs are categorized into two types: physical changes and non-physical changes. Physical changes are changes that can be seen in the fitness landscape, including changes in peak location, peak height, the size of the basin of attraction, and the number of peaks. Non-physical changes are categorised based on the characteristics of changes, including detectability, predictability, time-linkage, and noise. The physical changes are listed as follows.

1) The change in a peak's location within the peak's basin: To change a peak's location ($\mathbf{X}^{b_k}(t)$) within its basin b_k at time t, we change its mapping location $\mathbf{X}^{s_v}(t)$ (see Eq. (18)) in the search space of the associated function s_v by

$$\mathbf{X}^{s_v}(t+1) = (\mathbf{X}^{s_v}(t) - \mathbf{X}^{s_v}(t-1))\lambda + \nu(1-\lambda)\mathrm{N}(0, \sigma^{s_v}), \tag{20}$$

where ν is a normalized vector with a random direction; $\mathrm{N}(0, \sigma^{s_v})$ returns a random number of the normal distribution with mean 0 and variance σ^{s_v} (the shift severity with a default value of 1); $\lambda \in [0, 1]$ is a parameter to determine the correlation between the direction of the current movement and the previous movement. $\lambda = 1$ indicates the direction of a peak's movement is predictable, and $\lambda = 0$ indicates the movement of a peak is completely in a random direction. The ith dimension of $\mathbf{X}^{s_v}(t)$ will be re-mapped to a valid location if it moves out of the range of the component function by (see the movement of peak $p1$ in Fig. 5)

$$X_i^{s_v} = \begin{cases} l_i^{s_v} + (u_i^{s_v} - l_i^{s_v})\frac{(l_i^{s_v} - X_i^{s_v})}{(u_i^{s_v} - X_i^{s_v})} & X_i^{s_v} < l_i^{s_v}, \\ l_i^{s_v} + \frac{(u_i^{s_v} - l_i^{s_v})^2}{(X_i^{s_v} - l_i^{s_v})} & X_i^{s_v} > u_i^{s_v} \end{cases} \tag{21}$$

2) The change in a peak's shape: To change a peak's shape in subspace b_k, a random shape function is chosen from the eight basic functions by

$$f^{b_k} = s_{\mathrm{R}(1,8)}, \tag{22}$$

where $\mathrm{R}(a, b)$ returns a random number of the uniform distribution within $[a, b]$.

3) The change in the size of a peak's basin of attraction: To vary the size of the basin of attraction of a peak, we just need to change the value of the cutting hyper-plane constructed with the dimension c of a division point $\mathbf{dp}$ (point $\mathbf{dp}$ should be a parent node of a leaf node in the kd-tree, e.g., node (2,3) in Fig. 3) by

$$dp_c = dp_c + \mathrm{R}(-\sigma_c, \sigma_c), i = 1, 2, \ldots D, \tag{23}$$

where σ_c is a constant related to the range of the cth dimension of the hyper-rectangle cut by the cutting hyper-plane for the generation of two neighbour subspaces. Note that, two neighbour subspaces will change if we change the value of a cutting dimension (see the movement of peak p_2 in Fig. 5).

4) The change in a peak's height: The height of a peak at time t is changed by

$$H_i(t+1) = \begin{cases} H_i(t) - \delta_{h_i} & H_i(t+1) < H_{min} || H_i(t+1) > H_{max}, \\ H_i(t) + \delta_{h_i} & Otherwise, \end{cases} \tag{24}$$

where δ_{h_i}=$\mathrm{N}(0, \sigma_{h_i})$, σ_{h_i} is the height severity of peak p_i, σ_{h_i} is set to a random value in $[0, 7]$; H_{min} and H_{max} are the minimum and maximum heights, which are set to 0 and 100, respectively, in this paper.

5) The change in the number of peaks: The number of peaks follows a recurrent change by

$$N(t+1) = \begin{cases} \sigma_N(N(0) + t)\%T + N_{min} & (N(0) + t)\%T = 0, \\ \sigma_N(T - (N(0) + t)\%T) + N_{min} & Otherwise, \end{cases} \tag{25}$$

where $N(t)$ is the number of peaks at time t ($N(0)$ is the initial number of peaks); $\sigma_N = 2$ is the change step; $T = 25$ is the time period; N_{min}=1 is the minimum number of peaks. If the number of peaks increases, σ_N random division points are added to the division set; Otherwise, σ_N points are randomly removed from the division set.

In addition to the predictable change in a peak's location and the recurrent change in the number of peaks, three other non-physical features are introduced: a time-linkage change, a partial change, and noisy environments. In the time-linkage change, a peak changes only when it is found by an optimizer. For the partial change, a part of peaks change when an environmental change occurs. In the noisy environment, noise is added to a solution when it is to be evaluated by

$$x_i = x_i + \sigma_{noi} BR_{b_k} \mathrm{N}(0, 1), \tag{26}$$

where $i = 1, \ldots, D, b_k = inquire(\mathbf{x})$, σ_{noi}=0.01 is the noise severity; BR_{b_k} is the basin ratio of the subspace b_k where $\mathbf{x}$ is located.

Table I summarizes the feature comparison with other four popular dynamic benchmarks. From the table, the FPs provides many more features than the other four benchmarks.

TABLE I
FEATURE COMPARISON WITH PEER BENCHMARKS

Physical changes/ Non-physical changes	MPB[4]	DF1 [43]	RDBG [31]	CDBG [31]	FPs
Peak location	✓	✓	✓	✓	✓
Peak height	✓	✓	✓	✓	✓
Peak width	✓	✓	✓	✓	✓
Peak shape	×	×	×	×	✓
Movement within the basin	×	×	×	×	✓
Manageable basin size	×	×	×	×	✓
Number of peaks	×	×	✓	×	✓
Recurrent	×	×	✓	✓	✓
Partial	×	×	✓	×	✓
Time-linkage	×	×	×	×	✓
Noise	×	×	✓	×	✓
Predictable	✓	×	×	×	✓

C. Multi-Objective Optimization Problems

In an MOOP for maximum optimization, a solution $\mathbf{a}$ is said to dominate another solution $\mathbf{b}$, denoted as $\mathbf{a} \succ \mathbf{b}$, *iff* $\forall i \in \{1, \ldots, O\}, f_i(\mathbf{a}) \geq f_i(\mathbf{b})$ and $\mathbf{f}(\mathbf{a}) \neq \mathbf{f}(\mathbf{b})$. A solution $\mathbf{x}^*$ is called a *Pareto optimal solution*, *iff* $\nexists\ \mathbf{y}$ that $\mathbf{y} \succ \mathbf{x}^*$. The set of all Pareto optimal solutions is called the Pareto optimal set (POS) and the image of the POS in the objective space is called the *Pareto optimal front* (POF).

In this subsection, we aim to provide a flexible framework for constructing MOOPs, which are easy to analyze the POS and POF but can have hard properties for optimizers. To construct a problem with O objectives under the FPs, we set O component functions in every subspace. Each component function is associated with an objective. In this paper, component functions are chosen from s_1-s_8 defined in Sect. III-A.

1) Problems with A Finite Countable POS : To construct an MOOP where its POS is finite and countable, we just simply set peaks of all component functions within a subspace at a same location, i.e., in subspace b_k, we have $X_1^{b_k}$=$X_2^{b_k}$,...,=$X_O^{b_k}$, where $X_j^{b_k}, j = 1, \ldots, O, k = 1, \ldots, N$, is the peak of the component function associated with objective f_j in subspace b_k. For solutions in each subspace, the solution at the peak location dominates all other solutions since the objective values of all component functions monotonically decrease as the distance to the peak increases. Therefore, the peak solution has the largest objective values on all objectives. The POS comprises all the peak solutions, which are all non-dominated. Points on the POF of this type of problems are countable and completely disconnected with each other.

2) Problems with an Infinite Non-Countable POS and without Space Division: Here we consider the case where there is no space division. To construct an MOOP with an infinite POS, we just simply set peaks of component functions at different locations in the solution space. For simplicity, in this paper the peak associated with f_1 is located at the center of a subspace and peaks associated with remaining objectives are located on a sphere, which is centered at the center peak with a radius $100r^b$ ($r^b \in [0, 1]$).

Here we explain how to identify the POS for such a problem. As mentioned above, functions s_1-s_8 have rotational symmetry with respect to their peaks and the objective value of each function monotonically decreases as the distance to its peak increases. For any point $\mathbf{x}$ in the solution space, we draw O spheres, of which each sphere is centered at a particular peak and passes through point $\mathbf{x}$. Solutions within a sphere have greater objective values than point $\mathbf{x}$ with respect to the objective associated with the sphere center. Similarly, solutions outside a sphere have smaller objective values than point $\mathbf{x}$ with respect to the associated objective. Therefore, solutions in the intersection of the inside of all spheres dominate point $\mathbf{x}$ and solutions in the intersection of the outside of all spheres are dominated by point $\mathbf{x}$.

Fig. 6 shows a two-objective case, where $\mathbf{X}_1$ and $\mathbf{X}_2$ are locations of peaks associated with objective functions f_1 and f_2, respectively, and function s_1 is used for both f_1 and f_2. According to the above analysis, solution $\mathbf{b}$ dominates solutions in the two red dotted areas and is dominated by solutions in the red solid area as shown in the middle graph.

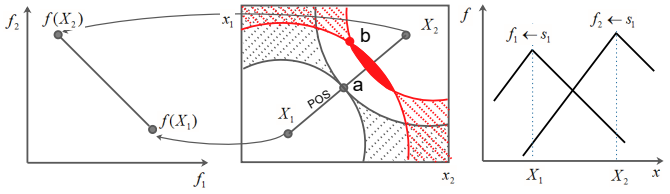


Fig. 6. Illustration of a two-objective problem with D=2, where $\mathbf{X}_1$ and $\mathbf{X}_2$ are locations of the peaks associated with f_1 and f_2, respectively, the left graph is the POF, the middle graph shows the search space, and the right graph shows functions f_1 and f_2.

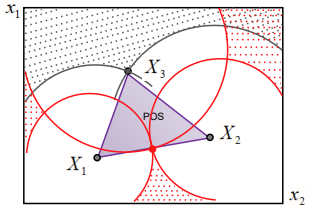
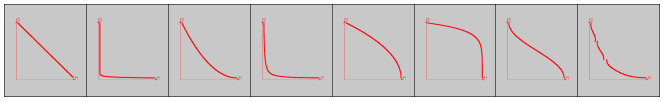


Fig. 7. Illustration of a three-objective problem with D=2, where $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ are locations of peaks associated with three objectives, the POS is the solutions in the triangle, solutions in the red dotted areas are dominated by the red point, and solutions in areas with grey dots are dominated by $\mathbf{X}_3$.

For a solution $\mathbf{a}$ on the line segment $\overline{\mathbf{X}_1\mathbf{X}_2}$, there is no solution which dominates it. Therefore, $\mathbf{a}$ is a solution in the POS. Together with the above analysis, it can be seen that the POS of the problem is composed of all solutions on the line segment between $\mathbf{X}_1$ and $\mathbf{X}_2$. Fig. 7 shows a three-objective case. For point $\mathbf{X}_3$ and the red point between $\mathbf{X}_1$ and $\mathbf{X}_2$, there is no point that dominates them and they dominate all the points in grey and red color, respectively. The POS for this case, hence comprises solutions in the triangle $\triangle\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3$. In the general case with O objectives and D dimensions, the POS is the set of points in a closed convex hyper-polygon expanded by $\mathbf{X}_1, \cdots, \mathbf{X}_O$, where $\mathbf{X}_j$ denotes the peak of the component function associated with objective f_j (see the mathematical proof in Appendix E).

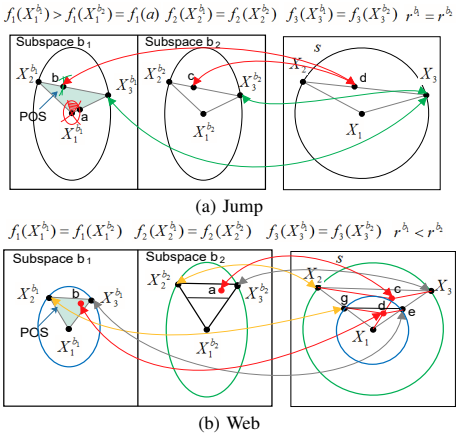
Different shapes of the POF can be constructed if we set component functions with different shapes. Fig. 8 shows eight shapes of the POF of problems with two objectives, where component functions s_1-s_8 are used as f_1, respectively, for the eight problems and s_1 is used as f_2 for all problems. Among these shapes, one is linear, three are convex, two are concave, one is both concave and convex, and one is disconnected (see Appendix F for the three-objective version of each problem).

3) Problems with an Infinite POS and Space Division: Problems without space division may have simple properties and they may be easy to solve. Problems with space division, on the other hand, can have complicated properties, which make them hard to solve. In this subsection, the same component function is used for all peaks associated with a particular objective in all subspaces. In this section two types of problems are constructed, namely Jump and Web. In a solution space $\mathcal{S}$ that is equal to the solution space of any component function s_v, $v = 1, 2, \ldots, 8$, we first draw N spheres centered at the center of $\mathcal{S}$, where the radius of each sphere is the same for the Jump type and is different for the Web type (note that, one sphere is sufficient in the Jump case, but we still draw N spheres for the consistency of the



(a)f_1:s_1 (b)f_1:s_2 (c)f_1:s_3 (d)f_1:s_4 (e)f_1:s_5 (f)f_1:s_6 (g)f_1:s_7 (h)f_1:s_8

Fig. 8. The POF of two-objective problems, where f_2 : s_1, r_b=0.8, m=3 and η=2.5 for the problem shown in graph (h).



(a) Jump

(b) Web

Fig. 9. Illustration of dominating relationships for two types of problems, where each has two subspaces and curved arrows denote the mappings between points in the subspaces of $\mathbf{f}$ and points in a space $\mathcal{S}$.

description of the two types of problems). Then we draw $O-1$ rays that start from the center of $\mathcal{S}$ by random directions. These rays intersect each sphere at $O-1$ intersection points. Finally, to construct an MOOP with O objectives and N subspaces, the intersection points on each sphere and the center point of $\mathcal{S}$ are mapped to the peak points in a different subspace $b_k, k \in [1, N]$ according to Eq. (18) (see the peak mapping relationship for both types of problems in Fig. 9 introduced later). Note that, the peak associated with f_1 is located at the center of each subspace. For both types of problems, all peaks associated with a particular objective have the same height except peaks associated with f_1, which have different heights, in the Jump case. The illustration of the POS of these two types of problems is explained as follows.

Fig. 9 shows two examples of three-objective problems with two subspaces where curved arrows denote the mappings defined by Eq. (18) between points in subspaces of function $\mathbf{f}$ to points in the solution space $\mathcal{S}$. Note that, a sphere in the solution space $\mathcal{S}$ with a cube shape will map to an ellipse in a subspace b_k with a rectangle shape. We still use the term "sphere" instead of "ellipse" for consistency in this section.

The problem in Fig. 9-(a) belongs to the Jump type, where $f_1(\mathbf{X}_1^{b_1}) > f_1(\mathbf{X}_1^{b_2})$, $f_2(\mathbf{X}_2^{b_1}) = f_2(\mathbf{X}_2^{b_2})$, $f_3(\mathbf{X}_3^{b_1}) = f_3(\mathbf{X}_3^{b_2})$, r^{b_1}=r^{b_2}, X_j, $j = 1, 2, 3$, is the jth peak in the solution space $\mathcal{S}$ and $X_j^{b_k}$, $j = 1, 2, 3, k = 1, 2$ is the mapping peak of X_j in subspace b_k. In the graph, any point $\mathbf{x}^{b_2}$ in the triangle $\triangle \mathbf{X}_1^{b_2}\mathbf{X}_2^{b_2}\mathbf{X}_3^{b_2}$ is dominated by at least one point in $\triangle \mathbf{X}_1^{b_1}\mathbf{X}_2^{b_1}\mathbf{X}_3^{b_1}$. This is because, for any point $\mathbf{x}^{b_2}$ there always exists one point $\mathbf{x}^{b_1}$ in $\triangle \mathbf{X}_1^{b_1}\mathbf{X}_2^{b_1}\mathbf{X}_3^{b_1}$ so that $\mathbf{x}^{b_1}$ has the same mapping point as $\mathbf{x}^{b_2}$ in the search space of $\mathcal{S}$. It is easy to see that point $\mathbf{x}^{b_1}$ has the same objective value as point $\mathbf{x}^{b_2}$ on f_2 and f_3, respectively, and a greater objective value than point $\mathbf{x}^{b_2}$ on f_1 due to $f_1(\mathbf{X}_1^{b_1}) > f_1(\mathbf{X}_1^{b_2})$, $f_2(\mathbf{X}_2^{b_1}) = f_2(\mathbf{X}_2^{b_2})$, and $f_3(\mathbf{X}_3^{b_1}) = f_3(\mathbf{X}_3^{b_2})$.

Here we explain how to find all points which dominate a point $\mathbf{x}^{b_2}$. Take the case where $\mathbf{x}^{b_2}$ is also $\mathbf{X}_1^{b_2}$ as an example. Due to $f_1(\mathbf{X}_1^{b_1}) > f_1(\mathbf{X}_1^{b_2})$, we assume that there exists a point $\mathbf{a}$ on the line segment $\overline{\mathbf{X}_1^{b_1}\mathbf{X}_3^{b_1}}$ so that $f_1(\mathbf{X}_1^{b_2})$=$f_1(\mathbf{a})$. Note that, this assumption holds if $f_1(\mathbf{X}_1^{b_2}) > f_1(\mathbf{X}_3^{b_1})$. If $f_1(\mathbf{X}_1^{b_2}) \leq f_1(\mathbf{X}_3^{b_1})$, then point $\mathbf{a}$ will be on the extension of $\overline{\mathbf{X}_1^{b_1}\mathbf{X}_3^{b_1}}$ starting from $\mathbf{X}_3^{b_1}$. However, this will not affect the analysis below. Given the above, we can have three observations for the three objectives f_1, f_2, and f_3: Firstly, because $f_1(\mathbf{X}_1^{b_1}) > f_1(\mathbf{X}_1^{b_2})$=$f_1(\mathbf{a})$, $f_1(\mathbf{X}_1^{b_2})$ will always be less than the objective values of all solutions within a sphere which is centered at $\mathbf{X}_1^{b_1}$ and goes through $\mathbf{a}$. Secondly, because $f_2(\mathbf{X}_2^{b_1}) = f_2(\mathbf{X}_2^{b_2})$ and r^{b_1}=r^{b_2}, $f_2(\mathbf{X}_1^{b_2})$ will always be less than the objective values of all solutions within a sphere which is centered at $\mathbf{X}_2^{b_1}$ and goes through $\mathbf{X}_1^{b_1}$. Thirdly, because $f_3(\mathbf{X}_3^{b_1}) = f_3(\mathbf{X}_3^{b_2})$ and r^{b_1}=r^{b_2}, $f_3(\mathbf{X}_1^{b_2})$ will always be less than the objective values of all solutions within a sphere which is centered at $\mathbf{X}_3^{b_1}$ and goes through $\mathbf{X}_1^{b_1}$. Combining the three observations above, we can conclude that $\mathbf{X}_1^{b_2}$ is dominated by solutions in the intersection of the three aforementioned spheres (the red shaded area shown in Fig. 9-(a)).

Interestingly, as shown in Fig. 9-(a) for any point $\mathbf{c}$ on the line segment $\overline{\mathbf{X}_2^{b_2}\mathbf{X}_3^{b_2}}$ in the subspace b_2, there exists only one point $\mathbf{b}$, which dominates point $\mathbf{c}$, on the line segment $\overline{\mathbf{X}_2^{b_1}\mathbf{X}_3^{b_1}}$ in the subspace b_1. If we draw three spheres as above, it is easy to see that the intersection of the inside of the three spheres only has one point $\mathbf{b}$ as shown in Fig. 9-(a) since two spheres centered at $\mathbf{X}_2^{b_1}$ and $\mathbf{X}_3^{b_1}$, respectively, are tangential at point $\mathbf{b}$. Points $\mathbf{b}$ and $\mathbf{c}$ have the same objective values on f_2 and f_3, respectively. However, point $\mathbf{b}$ has a greater objective value than point $\mathbf{c}$ due to $f_1(\mathbf{X}_1^{b_1}) > f_1(\mathbf{X}_1^{b_2})$ and $|\overline{\mathbf{X}_1^{b_1}\mathbf{b}}| = |\overline{\mathbf{X}_1^{b_2}\mathbf{c}}|$ (points $\mathbf{b}$ and $\mathbf{c}$ map to the same point in space $\mathcal{S}$, i.e., point $\mathbf{d}$ in the right graph). Therefore, we can deduce that $\mathbf{b}$ dominates $\mathbf{c}$. This will cause difficulties for optimizers to give up solutions on $\overline{\mathbf{X}_2^{b_2}\mathbf{X}_3^{b_2}}$ to explore the true POS (see the results in Appendix A.D). This is because, for any point $\mathbf{c}$ on $\overline{\mathbf{X}_2^{b_2}\mathbf{X}_3^{b_2}}$ obtained by an algorithm, the algorithm must find an exact point $\mathbf{b}$ so that the dominated point $\mathbf{c}$ can be eliminated during the selection. Finally, combining the analysis above with the analysis of problems without space division, we can see that the POS for a Jump problem comprises solutions in the triangle with the highest peak associated with f_1.

Fig. 9-(b) shows a problem of the Web type, where the radius of the sphere in subspace b_2 is greater than that in subspace b_1, i.e., $r^{b_1} < r^{b_2}$ and points $\mathbf{g}$ and $\mathbf{e}$ are the mapping points of $\mathbf{X}_2^{b_1}$ and $\mathbf{X}_3^{b_1}$, respectively, in the space $\mathcal{S}$. Fig. 9-(b) also shows the two mapping spheres of the respective spheres in subspaces b_1 and b_2 (in blue and green). Since the sphere in b_1 is smaller than the sphere in b_2, the mapping sphere of b_1 (in blue) is also smaller than the mapping sphere of b_2 (in green). In the figure, for any point $\mathbf{a}$ in $\triangle \mathbf{X}_1^{b_2}\mathbf{X}_2^{b_2}\mathbf{X}_3^{b_2}$, there exist a point $\mathbf{b}$ in $\triangle \mathbf{X}_1^{b_1}\mathbf{X}_2^{b_1}\mathbf{X}_3^{b_1}$ so that $\mathbf{b}$ dominates $\mathbf{a}$. In the right graph, assuming point $\mathbf{c}$ is the mapping point of point $\mathbf{a}$, it is easy to find a point $\mathbf{d}$ on the line segment $\overline{\mathbf{X}_1\mathbf{c}}$ so that $\frac{|\overline{\mathbf{X}_1\mathbf{g}}|}{|\overline{\mathbf{g}\mathbf{X}_2}|} = \frac{|\overline{\mathbf{X}_1\mathbf{e}}|}{|\overline{\mathbf{e}\mathbf{X}_3}|} = \frac{|\overline{\mathbf{X}_1\mathbf{d}}|}{|\overline{\mathbf{d}\mathbf{c}}|}$ and hence, $|\overline{\mathbf{X}_1\mathbf{d}}| < |\overline{\mathbf{X}_1\mathbf{c}}|$, $|\overline{\mathbf{d}\mathbf{g}}| < |\overline{\mathbf{X}_2\mathbf{c}}|$, and $|\overline{\mathbf{d}\mathbf{e}}| < |\overline{\mathbf{X}_3\mathbf{c}}|$. Now if we call $\mathbf{b}$ the mapping point of $\mathbf{d}$ in subspace b_1, we can see

that $\mathbf{b}$ dominates $\mathbf{a}$. The proof above means that, for any pair of subspaces (like the example in Fig. 9-(b)), the one with a smaller sphere radius will always have dominating solutions (in the triangle $\triangle \mathbf{X}_1^{b_1}\mathbf{X}_2^{b_1}\mathbf{X}_3^{b_1}$). More generally, among all the subspaces, the one with the smallest sphere radius will be the one that contains dominating solutions, and the POS will be all the points in the triangle in that smallest sphere. Therefore, the POS in Fig. 9-(b) comprises solutions in the triangle in subspace b_1, which has the smallest sphere radius.

D. Dynamic Multi-Objective Optimization Problems

Based on the structure of MOOPs, we can construct a DMOOP with the four change types mentioned in Sect. II-4. The four change types are implemented in six changes in this paper, including changes of rotation, web, jump, basin size, the number of subspaces, the number of dimensions, and the number of objectives. In all the six changes, peaks' locations are set using the same way as described in the previous subsections. All the changes use the same initial setup, where all peaks have the same height and each subspace has a random r^b (determining the radius of the hyper-sphere where peaks locate), except the jump change. In the jump change, all subspaces have the same value for r^b and all on-sphere peaks have the same height.

1) Rotation: The landscape of each subspace $b_k, k \in [1, N(t)]$ is rotated separately by a transformation matrix $\mathbf{A}^{b_k}$, which is a product of a set of rotation matrices. Each rotation matrix $\mathbf{M}_{i_1,i_2}^{\theta^{b_k}}$ is obtained by rotating the projection of the decision space in the plane i_1-i_2, $i_1, i_2 \in [1, D]$ by an angle θ^{b_k} from the i_1-th axis to the i_2-th axis. A point $\mathbf{x}^{b_k}(t)$ in subspace b^k is changed by the following procedures:

- Randomly select L dimensions (L is an even number) from the D dimensions to compose a vector.
- In the vector, for each pair of two neighbor dimensions i_l and i_{l+1}, construct a rotation matrix $\mathbf{M}_{i_l,i_{l+1}}^{\theta^{b_k}}$.
- A transformation matrix $\mathbf{A}_k$ is obtained by

$$\mathbf{A}^{b_k} = \mathbf{M}_{i_1,i_2}^{\theta^{b_k}} \cdot \ldots \cdot \mathbf{M}_{i_{L-1},i_L}^{\theta^{b_k}}, \theta^{b_k} = \mathrm{R}(0, 2\pi\sigma_\theta) \quad (27)$$

  where $\sigma_\theta \in [0, 1]$ is a rotation severity with a default value of 0.1.
- Rotate $\mathbf{x}^{b_k}(t)$ by $\mathbf{x}^{b_k}(t+1) = \mathbf{x}^{b_k}(t) \cdot \mathbf{A}_k(t)$

2) Web: In each subspace b_k, we simply change the value of parameter r^{b_k} as follows:

$$r^{b_k}(t+1) = r^{b_k}(t) + \mathrm{R}(-\sigma_r, \sigma_r), \quad (28)$$

where σ_r is a web severity with a default value of 5. Then, we move each on-sphere peak as follows:

$$\mathbf{X}_j^{b_k}(t+1) = r^{b_k}(t+1) \cdot normalize(\mathbf{X}_j^{b_k} - \mathbf{X}_1^{b_k}), \quad (29)$$

where $j = 2, \ldots, O$ (the number of objectives). Note that, a random initial r^b is generated for each subspace.

3) Jump: In this change, all on-sphere peaks remain unchanged and only center peaks are allowed to change their heights. All subspaces use the same parameter settings except the height for the center peak with objective f_1. Eq. (24) is used to change the height of center peaks.

4) Basin Size: The size of each subspace is allowed to change and the change in Sect. IV-B3 is used.

5) The number of subspaces, the number of dimensions, and the number of objectives: The same method as used in Sect. IV-B5 is used for the three changes, which is shown in Eq. (25), except that the step size for the three changes is one and the minimum value for dimensions and objectives is two. For the three changes, a random subspace/dimension/objective is added/removed when the number is increased/decreased by one. To remove one random objective, each associated peak with that objective in every subspace is removed. Note that, due to the structure of MOOPs it is not allowed to remove the first objective. To add one objective, a new peak with the same component function is added on the sphere in each subspace. Regarding the addition of a new subspace, a subspace with a random r^b is added in the Web case or a subspace with the first peak of a random height is added in the Jump case. Other parameters for the new subspace use the same configurations as other subspaces.

Given the above changes, we can analyze the effect of changes on the POS and POF. For the rotation and basin size changes, the POS changes but the POF does not change. This is because these changes do not alter the distribution of the POS and the relative positions of all peaks in a subspace. The jump and web changes will affect both the POS and the POF. The distribution of the POS will move from one subspace to another subspace if the subspace with the highest center peak in the jump change or the subspace with the smallest r^b in the web change does not remain the highest or smallest. The POF will also change even if the subspace with the highest peak or the smallest r^b remains the same. Take a two-objective problem as an example. If two peaks move close to each other, the largest objective value of each objective will not change, but the smallest value of each objective will increase. As a result, the POF will also change. For the change in the number of subspaces, it may affect both the POS and POF or neither the POS nor the POF, depending on the r^b value of the new subspace. If the r^b of the new subspace is the smallest, then the POF and the POS will change; Otherwise, the POS and POF will not change.

V. Experimental Studies

Due to the space limitation, this section only carries out experiments on the effect of transformations for GOPs. Other experimental studies are moved to Appendix A in the supplementary file.

A. Experimental Setup

For GOPs, we select five traditional algorithms: two of them are the standard particle swarm optimization (PSO) [5] with a local-best model (PSO/L) and the global-best model (PSO/G), respectively, the other two are the differential evolution (DE) [55] with mutation strategies of DE/Rand/1 and DE/Best/2, respectively, and the last choice is the evolution strategy with covariance matrix adaptation (CMA-ES) [15]. For both PSOs, the initial weight [52] version is used. In the PSO/L, an

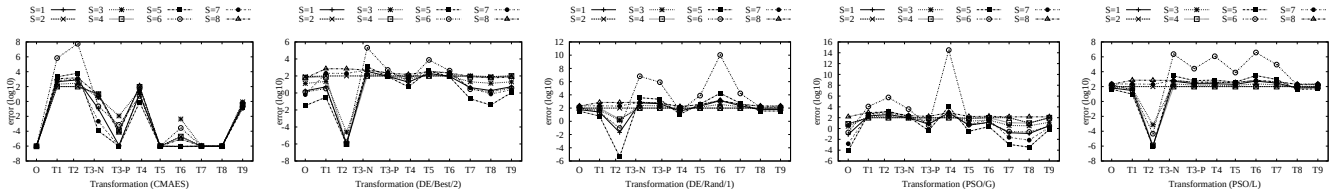


Fig. 10. The errors obtained by the five algorithms on the eight one-peak problems with different transformations in 100 dimensions, where O denotes the original problem, T3-N and T3-P denote non-separable and partially separable transformations, respectively.

adaptive random topology [6] is used, where a particle's neighbourhood is updated with a certain number of random particles when the best particle shows no improvement. The parameters of the PSOs are set to $\omega = 0.7298$ and $\eta_1 = \eta_2 = 1.496$ as suggested by [59] and the parameters F and CR for the DEs are set to 0.5 and 0.6 as suggested by [55]. The latest source code for CMA-ES from the GitHub page was used in this paper and the suggested parameter settings are used. Because the purpose of this experiment is studying the effect of transformations on GOP algorithms rather than tuning algorithms to find the best settings, the population size for all the five algorithms are set to 100.

B. Effect of Transformations On One-peak Problems

In this sub-section, the effect of the transformations is investigated with the five chosen algorithms on the eight one-peak problems. All algorithms stop running when the error is less than 1.E-6 or the number of fitness evaluations reaches 100,000. All results are averaged over 100 independent runs. Fig. 10 shows the results of problems in 100 dimensions.

From the results, we can have the following observations. Firstly, the transformations brings difficulties for all algorithms except for CMA-ES with transformations flat border (T5), irregularity (T7) and asymmetry (T8). Transformation T5 makes CMA-ES spend a slightly greater number of evaluations than it does on the original problems, while the other two transformations have no impact on the performance of CMA-ES. Note that, DE/Best/2, DE/Rand/1 and PSO/L achieve better results on problems with the corner shift (T2) than their results on the original problems. This is due to that these three algorithms always assign boundary values to their individuals if they go outside the search boundary.

Secondly, different transformations have different impact on the performance of all algorithms regarding a particular problem. For example, the transformations of shift, interdependence, imbalance, and noise make problems much harder for CMA-ES than the other transformations. The non-separable problems (T3-N) are harder than the partially separable problems (T3-P) for all algorithms. Thirdly, a particular transformation also has different impact on different problems. For example, problem s_6 seems to be the most sensitive to most transformations, which makes it much more difficult to solve than other problems in many cases.

C. Effect of the Number of Partitions

In the paper, we introduce a method for creating global structures using traditional functions with a number of subspaces. With this method, each subspace contains a peak (i.e., a local optimum). In this sub-section, the effect of the number of partitions is investigated. An experiment is conducted on two traditional functions with different numbers of dimensions. In the experiment, the search space is evenly divided. Note that, the location and objective value of a simulated function are the same as its original function.

Fig. 11 shows the results of the five algorithm on the two problems with different numbers of dimensions. For both problems in low dimensional spaces (i.e, D=2 and D=5), the performance of all the algorithms does not vary much with different numbers of partitions. However, for the two problems in high dimensional spaces (i.e., D=10 and D=30), the performance varies much with different numbers of partitions. For the simulated Sphere function, the more partitions means the more local optima, and hence the harder to solve in comparison with the original Sphere function. This is because, the original Sphere function always has one peak regardless of the number of dimensions. Therefore, the results of the simulated Sphere functions with a large number of partitions are much worse than that of the original function. For the multi-modal Schwefel function, the number of local optima will increase exponentially as the number dimensions increases. However, for a simulated Schwefel function, the number of local optima does not change with different dimensions as long as the number of partitions does not change. Although the difficulty of the simulated Schwefel function increases as the number of partitions increases, the increased difficulty of the simulated function is far less than the increased difficulty of the original Schwefel function as the number of dimensions increases. Therefore, the results with the simulated Schwefel functions are much better than that of the original function.

D. Effect of Transformations in the Global Optimal Subspace

In this sub-section, we evenly divide the search space into 100 subspaces, and only apply the transformations to the subspace of the global optimum. For all the other subspaces, the function $s1$ is used and there is no transformation. Fig. 12 shows the effect of the transformations on the two simulated problems used in Sec. V-C, with different dimensions.

From Fig. 12, it can be seen that different transformations have very different impacts on the search of the global optimum for all algorithms. Take the two graphs on the right (1000 dimensions) as an example. The global optimum becomes difficult to find when its shape switches from s_1 to s_2. The random shift and the corner shift also make the global optimum harder to find, especially for the corner shift.

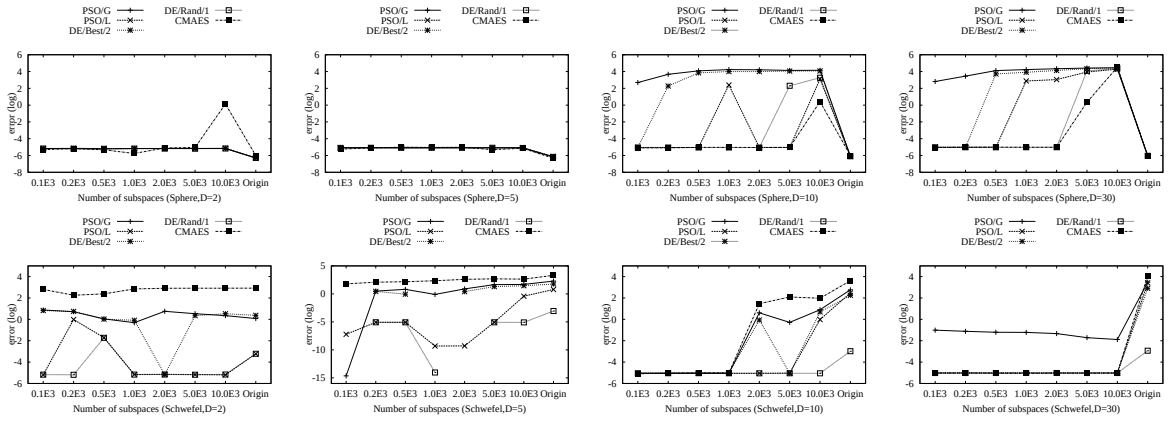


Fig. 11. Results of different algorithms on the two simulated functions with different numbers of partitions.

This is different from the effect of transformations on the one-peak problems in Fig. 10. The results of problems with transformation T3-N are better than that of T3-P. This is because that transformation T3-N makes individuals within the global optimal subspace hardly get improved and they all move to other peaks with relatively high heights.

VI. Conclusions

This paper proposes an open framework, named Free Peaks, for constructing continuous optimization problems. The FPs is able to construct different types of problems. With the exception of [61] in the domain of combinatorial optimization, this is the first framework that is able to create different types of numerical problems with a unified method. The framework is easy to understand, highly configurable, feature enriched, and computationally efficient. The properties and difficulties of problems constructed with the FPs are analyzable. This makes it easy for users to analyze weaknesses and strengths of an algorithm. The framework uses the building-blocks approach to construct a problem, therefore users can construct a problem with desired features. Moreover, the framework is extendable. Users can replace the component functions used in this paper with their own functions. To test an algorithm's performance on a problem with a certain feature, users just need to switch on or off that particular feature instead of switching to another problem. The framework is also compatible with existing problems. An important work in the future is to develop benchmark problems for each type of numerical problems. The design of typical global structures is also important.

Appendix

The appendix consists of the following contents:

1) Appendix A: Experimental studies.
2) Appendix B: Other one-peak problems.
3) Appendix C: Space partition with user-defined sizes.
4) Appendix D: Defining multi-funnel structures.
5) Appendix E: Proof of the POS.
6) Appendix F: Method of the generation of the POF.
7) Appendix G: The POS in multiple subspaces.
8) Appendix H: Default setups and experimental results.
9) Appendix I: Online source code at GitHub: https://github.com/Changhe160/FreePeaks.

References

[1] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[2] H.-G. Beyer and S. Finck, "Happycat — a simple function class where well-known direct search algorithms do fail," in *Proc. 12th Inter. Conf. on Parallel Problem Solving from Nature - Part I*, ser. PPSN'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 367–376.

[3] P. A. N. Bosman, "Learning, anticipation and time-deception in evolutionary online dynamic optimization," in *Proc. 2005 Genetic and Evol. Comput. Conf.* ACM, 2005, pp. 39–47.

[4] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. 1999 IEEE Congr. Evol. Comput.*, vol. 3, 1999, pp. 1875–1882.

[5] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. 2007 IEEE Swarm Intell. Symp.*, 2007, pp. 120–127.

[6] M. Clerc, *Particle Swarm Optimization*. ISTE (Int. Scientific and Technical Encyclopedia), 2006.

[7] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, Sep. 1999.

[8] K. Deb, L. Thiele *et al.*, "Scalable multi-objective optimization test problems," in *in Proc. 2002 IEEE Congr. Evol. Comput.*, 2002, pp. 825–830.

[9] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 425–442, Oct 2004.

[10] M. Gallagher and B. Yuan, "A general-purpose tunable landscape generator," *Trans. Evol. Comp*, vol. 10, no. 5, pp. 590–603, Oct. 2006.

[11] C. García-Martínez, P. D. Gutiérrez *et al.*, "Since cec 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness," *Soft Computing*, vol. 21, no. 19, pp. 5573–5583, Oct 2017. [Online]. Available: https://doi.org/10.1007/s00500-016-2471-9

[12] C.-K. Goh, "Evolutionary multi-objective optimization in uncertain environments," Ph.D. dissertation, Department of Electrical & Computer Engineering National University of Singapore, Singapore, 2007.

[13] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comp*, vol. 13, no. 1, pp. 103–127, Feb. 2009.

[14] S.-U. Guan, Q. Chen, and W. Mo, "Evolving dynamic multi-objective optimization problems with objective replacement," *Artificial Intell. Review*, vol. 23, no. 3, pp. 267–293, 2005.

[15] N. Hansen, S. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)." *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, 2003.

[16] N. Hansen, R. Ros *et al.*, "Impacts of invariance in search: When cma-es and pso face ill-conditioned and non-separable problems," *Appl. Soft Comput.*, vol. 11, pp. 5755–5769, 2011.

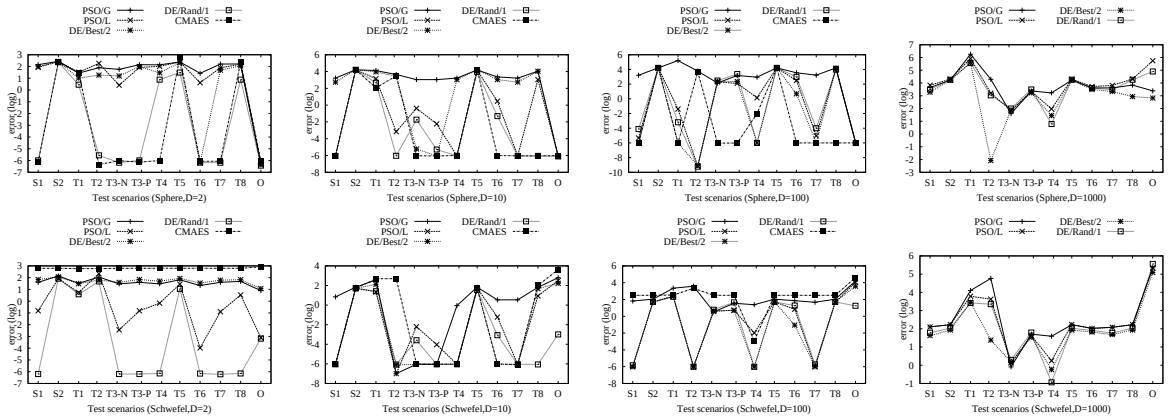


Fig. 12. Effect of transformations in the global optimal subspace of two simulated problems, where O denotes the original problems.

[17] N. Hansen and A. A. Steffen Finck, Raymond Ros, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions." RR-6829 INRIA, Tech. Rep., 2009.

[18] M. Helbig and A. P. Engelbrecht, "Archive management for dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation," in *Proc. 2011 IEEE Congr. Evol. Comput.*, June 2011, pp. 2047–2054.

[19] ——, "Dynamic multi-objective optimization using pso," in *Metaheuristics for Dynamic Optimization*, ser. Studies in Computational Intell., E. Alba, A. Nakib, and P. Siarry, Eds. Springer Berlin Heidelberg, 2013, vol. 433, pp. 147–188.

[20] ——, "Benchmarks for dynamic multi-objective optimisation algorithms," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 37:1–37:39, Jan. 2014.

[21] ——, "Benchmark functions for cec 2015 special session and competition on dynamic multi-objective optimization," University of Pretoria, Computer Science Department, Pretoria, South Africa, Tech. Rep., 2015.

[22] L. Huang, I. H. Suh, and A. Abraham, "Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants," *Inf. Sci.*, vol. 181, no. 11, pp. 2370 – 2391, 2011.

[23] V. L. Huang, A. K. Qin *et al.*, "Problem definitions for performance assessment of multi-objective optimization algorithms," Nanyang Technological University, Singapore., Special Session on Constrained Real-Parameter Optimization of the CEC07, Tech. Rep., 2007.

[24] S. Huband, P. Hingston *et al.*, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, 2006.

[25] H. Ishibuchi, N. Akedo, and Y. Nojima, "A many-objective test problem for visually examining diversity maintenance behavior in a decision space," in *Proc. of the 13th Annu. Conf. on Genetic and Evol. Comput.*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 649–656.

[26] S. Jiang and S. Yang, "Evolutionary dynamic multiobjective optimization: Benchmarks and algorithm comparisons," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 198–211, Jan 2017.

[27] Y. Jin and B. Sendhoff, "Constructing dynamic optimization test problems using the multi-objective optimization concept," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3005, pp. 525–536.

[28] P. Kerschke, M. Preuss *et al.*, "Detecting funnel structures by means of exploratory landscape analysis," in *Proc. of the 2015 Annu. Conf. on Genetic and Evol. Comput.*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 265–272.

[29] W. Koo, C. Goh, and K. Tan, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment," *Memetic Computing*, vol. 2, no. 2, pp. 87–110, 2010.

[30] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 561–578, Oct. 2007.

[31] C. Li and S. Yang, "A generalized approach to construct benchmark problems for dynamic optimization," in *7th Int. Conf. on Simulated Evolution and Learning*, 2008, pp. 391–400.

[32] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Trans. Systems, Man, and Cyberne., Part B*, vol. 42, no. 3, pp. 627–646, 2012. [Online]. Available: https://doi.org/10.1109/TSMCB.2011.2171946

[33] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, April 2009.

[34] M. Li, C. Grosan *et al.*, "Multi-line distance minimization: A visualized many-objective test problem suite," *IEEE Trans. on Evol. Comput.*, vol. PP, no. 99, pp. 1–1, 2017.

[35] M. Li, S. Yang, and X. Liu, "A test problem for visual investigation of high-dimensional multi-objective search," in *Proc. of the 2014 IEEE Congr. Evol. Comput.*, July 2014, pp. 2140–2147.

[36] J. Liang, P. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Swarm Intell. Symp., Proc. 2005 IEEE*, 2005, pp. 68–75.

[37] M. Lunacek, D. Whitley, and A. Sutton, *The Impact of Global Structure on Search*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 498–507.

[38] K. M. Malan and A. P. Engelbrecht, "Quantifying ruggedness of continuous landscapes using entropy," in *Proc. 2009 IEEE Congr. Evol. Comput.*, May 2009, pp. 1440–1447.

[39] ——, "A survey of techniques for characterising fitness landscapes and some possible ways forward," *Information Sciences*, vol. 241, pp. 148 – 163, 2013.

[40] J. Mehnen, T. Wagner, and G. Rudolph, "Evolutionary optimization of dynamic multiobjective functions," Department of Machining Technology,University of Dortmund, Germany, Tech. Rep., 2006.

[41] R. Morgan and M. Gallagher, "Using landscape topology to compare continuous metaheuristics: A framework and case study on edas and ridge structure," *Evol. Comput.*, vol. 20, no. 2, pp. 277–299, Jun. 2012.

[42] ——, *Length Scale for Characterising Continuous Optimization Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 407–416.

[43] R. W. Morrison and K. A. De Jon, "A test problem generator for non-stationary environments," in *Proc. 1999 IEEE Congr. Evol. Comput.*, 1999, pp. 2047–2053.

[44] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evol. Comput.*, vol. 6, no. 0, pp. 1 – 24, 2012.

[45] T. T. Nguyen and X. Yao, *Applications of Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ch. Dynamic Time-Linkage Problems Revisited, pp. 735–744.

[46] T. Okabe, Y. Jin *et al.*, "On test functions for evolutionary multi-objective optimization," in *Parallel Problem Solving from Nature - PPSN VIII*, X. Yao, E. K. Burke *et al.*, Eds. Springer Berlin Heidelberg, 2004, vol. 3242, pp. 792–802.

[47] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Inf. Sci.*, vol. 316, pp. 419 – 436, 2015, nature-Inspired Algorithms for Large Scale Global Optimization.

[48] J. Rönkkönen, X. Li *et al.*, "A framework for generating tunable test functions for multimodal optimization," *Soft Computing*, vol. 15, no. 9, pp. 1689–1706, 2011.

[49] G. Rudolph, B. Naujoks, and M. Preuss, "Capabilities of emoa to detect and preserve equivalent pareto subsets," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, S. Obayashi, K. Deb *et al.*, Eds. Springer Berlin Heidelberg, 2007, vol. 4403, pp. 36–50.

[50] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions - a survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, pp. 263–278, 1995.

[51] H. Seada and K. Deb, "A unified evolutionary optimization procedure for single, multiple, and many objectives," *IEEE Trans. Evol. Comput.*, vol. PP, no. 99, pp. 1–1, 2015.

[52] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. 1998 IEEE Congr. Evol. Comput.*, 1998, pp. 69–73.

[53] S. Shirakawa, N. Yata, and T. Nagao, "Evolving search spaces to emphasize the performance difference of real-coded genetic algorithms using genetic programming," *Trans. of the Japanese Society for Evol. Comput.*, vol. 1, no. 1, pp. 54–64, 2010.

[54] T. Smith, P. Husbands *et al.*, "Fitness landscapes and evolvability," *Evol. Comput.*, vol. 10, no. 1, pp. 1–34, Mar. 2002.

[55] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous space," *J. Global Optimization*, vol. 11, pp. 341–359, 1997.

[56] P. Suganthan, "Benchmarks for evaluation of evolutionary algorithms." [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/

[57] A. M. Sutton, M. Lunacek, and L. D. Whitley, "Differential evolution and non-separability: Using selective pressure to focus search," in *Proc. of the 9th Annu. Conf. on Genetic and Evol. Comput.*, ser. GECCO '07. New York, NY, USA: ACM, 2007, pp. 1428–1435.

[58] D. Thierens, D. Goldberg, and A. Pereira, "Domino convergence, drift, and the temporal-salience structure of problems," in *in Proc. 1998 IEEE Congr. Evol. Comput.*, 1998, pp. 535–540.

[59] F. Van Den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, Department of Computer Science, University of Pretoria, South Africa, 2002.

[60] Y. Wang and B. Li, "Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment," in *Proc. 2009 IEEE Congr. Evol. Comput.*, 2009, pp. 630–637.

[61] T. Weise, S. Niemczyk *et al.*, "A tunable model for multi-objective, epistatic, rugged, and neutral fitness landscapes," in *Proc. of the 10th Ann. Conf. on Genetic and Evol. Comput.*, ser. GECCO '08. New York, NY, USA: ACM, 2008, pp. 795–802.

[62] D. Whitley, S. Rana *et al.*, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245 – 276, 1996.

[63] M. Yang, C. Li *et al.*, "Differential evolution with auto-enhanced population diversity," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 302–315, Feb 2015.

[64] A. Zhou, Y. Jin *et al.*, "Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, S. Obayashi, K. Deb *et al.*, Eds. Springer Berlin Heidelberg, 2007, vol. 4403, pp. 832–846.

[65] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, pp. 173–195, 2000.

Changhe Li (M'12) received the B.Sc. and M.Sc. degrees in computer science from China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from the University of Leicester, U.K. in July 2011.

He has been an associate professor at China University of Geosciences (Wuhan) since 2011. He is the chair of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments. His research interests are evolutionary algorithms with machine leaning, swarm intelligence, multi-modal optimization and dynamic optimization.

Trung Thanh Nguyen received his BSc in 2000 from Vietnam National University, and his MPhil and PhD in Computing Science from University of Birmingham in 2007 and 2011, respectively.

He has been a Reader in Operational Research at Liverpool John Moores University (LJMU) since 2015. Prior to that, he was a Senior Lecturer in Optimisation and Simulation Modelling at LJMU since 2013, and a Research Fellow at LJMU and University of Birmingham in 2011. His current research interests include operational research/dynamic optimisation with a particular application to logistics/transport problems.

Sanyou Zeng received the B.Sc. degree in mathematics from Hunan University of Science and Technology, Xiangtan, China in 1983, the M.Sc. degree in mathematics from Hunan University, Changsha, China, in 1995, and the Ph.D. degree in computer science from Wuhan University, Wuhan, China in 2002.

He has been a professor at China University of Geosciences (Wuhan) since 2004. His research interest is evolutionary computation with machine learning for solving problems with constraints, multi-objective, dynamic environments and expensive costs, especially the antenna design problems.

Ming Yang received the B.Sc., M.Sc., and Ph.D. degrees in computer science from China University of Geosciences, Wuhan, China, in 2005, 2008, and 2012, respectively. He carried out a postdoctoral research at the School of Computer Science, University of Birmingham, U.K. from Dec., 2014 to Dec., 2015.

He is currently an Associate Professor with the School of Computer Science, China University of Geosciences, Wuhan, China. He also works in the Hubei Key Laboratory of Intelligent Geo-Information Processing, Wuhan, China. His research interests include swarm intelligence, large-scale optimization, multi-objective optimization and their applications.

Min Wu (SM'08) received the B.Sc. and M.Sc. degrees in engineering from Central South University (CSU), China, and the PhD degree in engineering from the Tokyo Institute of Technology, Japan, in 1983, 1986, and 1999, respectively.

He was a Lecturer and later a Professor with the School of Information Science and Engineering, CSU, Changsha, China, from 1986 to 2014. He was a Visiting Scholar with the Department of Electrical Engineering, Tohoku University, Sendai, Japan, from 1989 to 1990, a Visiting Research Scholar with the Department of Control and Systems Engineering, Tokyo Institute of Technology, from 1996 to 1999, and a Visiting Professor with the School of Mechanical, Materials, Manufacturing Engineering and Management, University of Nottingham, Nottingham, U.K., from 2001 to 2002. He joined China University of Geosciences, Wuhan, China, in 2014. He is currently a Professor with the School of Automation, China University of Geosciences. His current research interests include robust control and its application, process control, and intelligent control.