An Efficient Benchmark Generator for Dynamic Optimization Problems

Changhe Li(✉)

Hubei Key Laboratory of Intelligent Geo-Information Processing,
China University of Geosciences, Wuhan 430074, China
changhe.lw@gmail.com

Abstract. A number of benchmark generators have been proposed for dynamic single objective optimization problems. The moving peaks benchmark and the GDBG benchmark are widely used to test the performance of an evolutionary algorithm. The two benchmarks construct a fitness landscape with a number of peaks that can change heights, widths, and locations. The two benchmarks are simple and easy to understand. However, they exist two major issues: (1) the time complexity is high for evaluating a solution and (2) peaks may become invisible when changes occur. To address the two issues, this paper proposes an efficient generator with enriched features. The generator applies the k-d tree to partition the search space and sets a simple unimodal function in each sub-space. The properties of the proposed benchmark are discussed and verified by a set of evolutionary algorithms.

Keywords: Dynamic optimization problem · Generator

1 Introduction

In recent years, there has been a growing interest in developing evolutionary algorithms in dynamic environments. To comprehensively evaluate the performance of an evolutionary algorithm (EA), an important task is to develop a good benchmark generator. Over the years, a number of benchmark generators for dynamic optimization problems (DOPs) have been proposed. Generally speaking, these benchmark generators can be classified to the following three classes in terms of the way to construct problems. Note that, this paper focuses on only dynamic continuous unconstrained single objective optimization problems.

The first class of generators switch the environment between several stationary problems or several states of a problem. Early generators normally belong to this class. A generator based on two static landscapes A and B was proposed in [7]. Changes can occur in three ways: (1) linear translation of peaks in landscape A; (2) only the global optimum randomly moves in landscape A; (3) switching landscapes between A and B. In [15], the environment oscillates among a set of fixed landscapes.

Like the first class of generators, the second class of generators also consist of a number of basic functions. However, the environment normally takes the form

M. Gong et al. (Eds.): BIC-TA 2016, Part II, CCIS 682, pp. 60–72, 2016.
DOI: 10.1007/978-981-10-3614-9_8

$f(\mathbf{x}) = max\{g_i(\mathbf{x})\}, i = 1, \ldots, N$ to construct the fitness landscape and it does not switch among the basic functions. The environmental changes are caused by the changes of every $g(\mathbf{x})$. Many generators fall into this class. The moving peaks benchmark (MPB) [15] is one of the widely used generators. The MPB consist of a number of peaks. Each peak is constructed by a simple unimodal function which can change in height, width, and location. The DF1 generator [15] and the rotation dynamic benchmark generator (RDBG) [10] use a similar way to construct the fitness landscape. The DF1 generator uses the logistic function to change the height, width and location of a peak, while the RDBG rotates the fitness landscape to generate changes. A new generator based on the framework of the DF1 was proposed in [20], where the basic function used to construct a peak in DF1 was replaced by two traditional functions in [20]. A fitness landscape consists of a number Gaussain peaks was introduced in [8] where a peak changes in its center, amplitude, and width. A challenging dynamic landscape was proposed in [10], called composition dynamic benchmark generator (CDBG), where a set of composition functions are shifted in the fitness landscape. The CDBG introduces several change types, e.g., small step changes, large step changes, random changes, chaotic changes, recurrent changes and noisy environments.

The third class of generators divide the search space into subspaces and set simple unimodal functions in each subspace. A disjoint generator was proposed in [21], where each dimension of the search space is evenly divided into w segments. The total number of subspaces is w^D (D denotes the number of dimensions). In each subspace, a peak function is defined where its global optimum is at the center of the subspace.

The first and the third classes of generators lack of the ability of manipulating a single peak. In the literature of EAs for DOPs, the second class of generators are mostly used for experimental studies. This class of generators are flexible and easy to manipulate the characteristics of a change for every peak. However, it has two disadvantages. Firstly, to evaluation a solution $\mathbf{x}$, we need to compute the objective value of $\mathbf{x}$ for each basic function ($g(\mathbf{x})$) in $O(D)$ and then find out the maximum value as the fitness value of $\mathbf{x}$. Therefore, the time complexity of evaluating a solution is at least $O(ND)$. Secondly, a peak may become invisible when a change occurs, and hence the total number of peaks will be less than the predefined value.

Besides the way of the construction of the fitness landscape, researchers have also been interested in developing characteristics of changes to simulate real-world applications, such as the predictability–whether changes are predictable in a regular pattern, time-linkage–whether future changes depend on the current/previous solutions found by optimizers [4,17], detectability–whether changes are detectable, severity– determines the magnitude of a change, and change factors (objective functions, the number of dimensions, constraints, domain of the search space, and function parameters).

A good benchmark generator should have the following characteristics [16]: (1) Flexibility, the generator should be configurable regarding different aspects,

e.g., the number of peaks, change severity, and change features, etc.; (2) Simplicity and efficiency, the generator should be simple to implement, analyze, and computationally efficient; (3) The generator should be able to resemble real-world problems to some extent. Most real-world problems are very hard and complex, with nonlinearities and discontinuities [14].

Based on the above considerations, this paper aims to propose a novel generator, which is able to (1) address the two issues mentioned above of the current mainly used generators and (2) provide enriched characteristics of changes. To achieve the aims, this paper uses the idea of space partition and chooses a peak function for every sub-space from a predefined function set. A new benchmark generator, called Free Peaks (FPs), is proposed. The k-d tree [1] is used to partition the solution space. Each peak in a sub-space can be freely manipulated regarding its height, peak location, shape, and basin of attraction. A set of characteristics of changes are also introduced in this paper.

The rest of this paper is organized as follows. Section 2 introduce the construction of the FPs in detail, including the partition process of the k-d tree, a set of basic peak functions, and the setup of subspaces. Section 3 gives the construction of different types of changes. Section 4 presents the results of the experimental studies. Finally, conclusions are given in Sect. 5.

2 Free Peaks

The section introduces the basic elements of the free peaks (FPs) benchmark generator. Without loss of generality, maximization optimization problems are assumed in this paper. Before the introduction of the generator, we need to prepare a set of simple shape functions.

2.1 One Peak Function

In this paper, eight simple symmetrical unimodal functions are defined as follows:

$$s_1(\mathbf{x}) = h - d(\mathbf{x}), \tag{1a}$$

$$s_2(\mathbf{x}) = h \cdot \exp(-d(\mathbf{x})), \tag{1b}$$

$$s_3(\mathbf{x}) = h - \sqrt{h \cdot d(\mathbf{x})}, \tag{1c}$$

$$s_4(\mathbf{x}) = h/(1 + d(\mathbf{x})), \tag{1d}$$

$$s_5(\mathbf{x}) = h - d^2(\mathbf{x})/h, \tag{1e}$$

$$s_6(\mathbf{x}) = h - \exp\left(2\sqrt{d(\mathbf{x})/\sqrt{D}}\right) + 1, \tag{1f}$$

$$s_7(\mathbf{x}) = \begin{cases} h * \cos(\pi \cdot d(\mathbf{x})/r) & d(\mathbf{x}) \le r \\ -h - d(\mathbf{x}) + r & d(\mathbf{x}) > r \end{cases}, \tag{1g}$$

$$s_8(\mathbf{x}) = \begin{cases} \frac{h*(\cos(m\pi \cdot d(\mathbf{x})(1-1/r)) - \eta m d(\mathbf{x})/r)}{\sqrt{d(\mathbf{x})+1}} & d(\mathbf{x}) \le r \\ -h(\eta(m-1)+1)/\sqrt{r+1} & d(\mathbf{x}) > r \end{cases} \tag{1h}$$

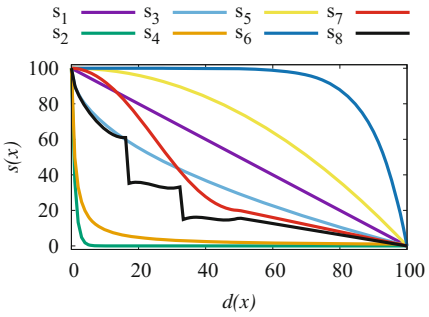

Fig. 1. Shapes of the eight functions with $D = 1$, where $h = 100, r = 50, \eta = 5.5$, and $m = 3$ (objective values of all functions are standardized within $[0, 100]$.

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

Figure 1 shows the shapes of the eight functions. Among these functions, s_1 is a linear function, s_2-s_4 are convex functions, s_5 and s_6 are concave functions, s_7 is partially convex, partially concave, and partially linear and s_8 is a disconnected function. Each function has a single peak located at X and is monotonic from the peak.

2.2 Partition the Search Space

The k-d tree [1] is a binary tree where each 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 sub-space

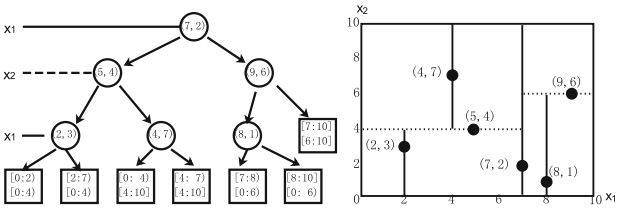

Fig. 2. An example of the k-d tree for the division of a 2-D space with ranges ([0:10], [0:10]) by a set of six points.

Algorithm 1. kdtree(*list*, *depth*)

```
1: axis ← depth%D;
2: Select the median by axis from list
3: if ||list||=1 then                                   ▷ A leaf node
4:     Create a sub-space;
5: else
6:     Create a node node with data of the median point;
7:     node.left ← kdtree(points in list before the median, depth+1);
8:     node.right← kdtree(points in list after the median, depth+1);
9:     return node;
10: end if
```

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

```
1: i ← depth%D;
2: if node is a leaf node then return the sub-space; end if
3: if x_i < node_i then
4:     inquire(x, node.left,depth+1);
5: else
6:     inquire(x, node.right,depth+1);
7: end if
```

of the solution space. Fig. 2 shows a k-d tree (Fig. 2-left) for the decomposition of a 2-D solution space (Fig. 2-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 sub-space, respectively. In this paper, the solution space is divided by default into N subspaces with random sizes

2.3 Setup of the Sub-space

A function $f(x)$ constructed based on the FPs can be defined by

$$f(\mathbf{x}) = \begin{cases} f^{b_1}(\mathbf{x}), \mathbf{x} \in [\mathbf{l}^{b_1}, \mathbf{u}^{b_1}] \\ f^{b_2}(\mathbf{x}), \mathbf{x} \in [\mathbf{l}^{b_2}, \mathbf{u}^{b_2}] \\ \cdots\cdots\cdots\cdots\cdots \\ f^{b_N}(\mathbf{x}), \mathbf{x} \in [\mathbf{l}^{b_N}, \mathbf{u}^{b_N}] \end{cases} \tag{2}$$

where each subspace b_k contains a basic function f^{b_k} associated with a shape function $s_v (k = 1, 2, \ldots N, v = 1, 2, \ldots, 8)$. The whole search space of $f([\mathbf{l}, \mathbf{u}])$ is divided into N subspaces: $[\mathbf{l}^{b_1}, \mathbf{u}^{b_1}], \ldots, [\mathbf{l}^{b_N}, \mathbf{u}^{b_N}]$, i.e., $[\mathbf{l}, \mathbf{u}]$=$\{[\mathbf{l}^{b_k}, \mathbf{u}^{b_k}], \ldots\}$, $k = 1, 2, \ldots N$. To compute the objective of $\mathbf{x}$ ($f(\mathbf{x})$), 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 $\mathbf{x}^{\mathbf{s_v}}$ in the search space of s_v associated with f^{b_k} in subspace b_k by

$$map(x_i) = x_i^{s_v} = l_i^{s_v} + (u_i^{s_v} - l_i^{s_v}) \frac{x_i - l_i^{b_k}}{u_i^{b_k} - l_i^{b_k}}, i = 1, 2, \ldots D, \tag{3}$$

where the mapping is linear from a solution in the subspace b_k of $\mathbf{f}([\mathbf{l}^{b_k}, \mathbf{u}^{b_k}])$ to a solution in the search space of s_v $([\mathbf{l}^{s_v}, \mathbf{u}^{s_v}])$. Eventually, we set the objective $f(\mathbf{x})$ by

$$f(\mathbf{x}) = f^{b_k}(\mathbf{x}) = s_v(\mathbf{x}^{s_v}). \tag{4}$$

2.4 Time Complexity

According to the above description of the FPs, to evaluate a solution $\mathbf{x}$ we need to perform the following three steps of procedures: (1) find out the sub-space (b_k) where $\mathbf{x}$ is; (2) map $\mathbf{x}$ to a location $(\mathbf{x}^{s_v})$ in the search space of f^{b_k}; (3) compute the objective of $(\mathbf{x}^{s_v})$ by one of the eight shape functions. Identifying a solution in which sub-space has a time 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 in the FPs is $O(log(N)) + 2O(D)$.

3 Constructing Dynamic Optimization Problems

This section introduces two types of changes: physical changes and non-physical changes. Physical changes are changes, which can be observed, including changes in peak location, peak shape, peak height, the size of the basin of attraction, and the number of peaks. Non-physical changes are characteristics of physical changes, including detectability, predictability, time-linkage, and noise. The physical changes are listed as follows.

3.1 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. (3)) in the search space of the associated component function s_v by

$$\mathbf{X}^{s_v}(t+1) = (\mathbf{X}^{s_v}(t) - \mathbf{X}^{s_v}(t-1))\lambda + \nu(1-\lambda)\mathrm{N}(0, \sigma^{s_v}), \tag{5}$$

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 as follows:

$$X_i^{s_v} = \begin{cases} l_i^{s_v} + (u_i^{s_v} - l_i^{s_v})\frac{(l_i^{s_v} - X_i^{s_v})}{(u_i{}^{s_v} - X_i^{s_v})} & X_i^{s_v} < l_i^{s_v}, \\ l_i^{s_v} + \frac{(u_i^{s_v} - l_i^{s_v})^2}{(X_i^{s_v} - l_i^{s_v})} & X_i^{s_v} > u_i^{s_v} \end{cases} \tag{6}$$

3.2 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 **dp** (point **dp** should be a parent node of a leaf node in the kd-tree, e.g., node (2,3) in Fig. 2) as follows.

$$dp_c = dp_c + \mathrm{R}(-\sigma_c, \sigma_c)(b^u_{k+1,c} - b^l_{k,c}), \tag{7}$$

where $\sigma_c = 0.01$ is the severity, $b^l_{k,c}$ and $b^u_{k+1,c}$ are the upper boundary and lower boundary of two neighbour subspaces b_k and b_{k+1}, respectively, which are generated by cutting the cth dimension of the hyper-rectangle for the generation of sub-spaces b_k and b_{k+1}; Note that, two neighbour subspaces will change if we change the value of a cutting dimension.

3.3 The Change in a Peak's Height

The height of a peak at time t is changed as follows:

$$H_i(t+1) = \begin{cases} H_i(t) - \delta_{h_i} & H_i(t+1) < H_{min} || H_i(t+1) > H_{max}, \\ H_i(t) + \delta_{h_i} & Otherwise, \end{cases} \tag{8}$$

where $\delta_{h_i} = \mathrm{N}(0, \sigma_{h_i}), \sigma_{h_i}$ 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.

3.4 The Change in the Number of Peaks

The number of peaks follows a recurrent change as follows:

$$N(t+1) = \begin{cases} \sigma_N(N(0)+t)\%T + N_{min} & (N(0)+t)\%T = 0, \\ \sigma_N(T - (N(0)+t)\%T) + N_{min} & Otherwise, \end{cases} \tag{9}$$

where $N(t)$ is the number of peaks at time t ($N(0)$ is the initial number of peaks); $\sigma_N = 2$ is a 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

$$x_i = x_i + \sigma_{noi} BR_{b_k} \mathrm{N}(0,1), \tag{10}$$

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

Table 2 summarizes the feature comparison between FPs and other four popular benchmarks. From the table, the FPs provides many more features than the other four benchmarks.

Table 1. Default settings for the FPs, where u means that the problem changes every u objective evaluations, a peak is found if the distances in objective space and decision space are less than ϵ_o and ϵ_s, respectively.

Parameter	Value	Parameter	Value
Number of peaks (N)	10	Number of dimensions (D)	5
Change frequency (u)	5000	Correlation coefficient (λ)	0
Basin change	No	Ratio of changing peaks (r_c)	1.0
Time-linkage change	No	Noisy environments	No
Height severity(σ_h)	[0,7]	Basin severity (σ_c)	0.01
Height range	(0,100]	Domain range	$[-100, 100]$
Initial peak shape	Random	Initial peak height	100
Initial peak location	Sub-space center	Number of steps (σ_N)	2
Shift severity (σ^{s_v})	1.0	Noise severity (σ_{noi})	0.01
Objective threshold (ϵ_o)	0.01	Distance threshold (ϵ_s)	0.1

Table 2. Feature comparison with peer benchmarks

Physical change/ Non-physical change	MPB [5]	DF1 [15]	RDBG [10]	CDBG [10]	FPs
Peak location	✓	✓	✓	✓	✓
Peak height	✓	✓	✓	✓	✓
Peak width	✓	✓	✓	✓	✓
Movement within the basin	×	×	×	×	✓
Manageable basin size	×	×	×	×	✓
Number of peaks	×	×	✓	×	✓
Recurrent	×	×	✓	✓	✓
Partial	×	×	✓	×	✓
Time-linkage	×	×	×	×	✓
Noise	×	×	✓	×	✓
Predictable	✓	×	×	×	✓

4 Experimental Studies

In this section, two groups of experiments are carried out. The first group of experiments aim to investigate the performance of the FPs and the second group of experiments aim to compare the performance of a set of existing EAs on the FPs.

4.1 Comparison of Computing Efficiency

In this subsection, an experiment is carried out to compare the performance of the FPs with two peer benchmark generators (the MPB [5] and the rotation DBG [10]) in terms of two different aspects. The first comparison is the computational efficiency. Figure 3 shows the time cost on evaluating one million random points for the three benchmarks in different scenarios. The left graph of Fig. 3 shows the comparison on problems with different numbers of peaks with 100 dimensions and the right graph shows the comparison on problems with different numbers of dimensions with 1,000 peaks. From the results, it can be seen that the time spent with the FPs is significantly smaller than the other two benchmarks. In both cases, the time spent with the FPs is almost constant as the number of dimensions/peaks increases in comparison with the time spent with the other two benchmarks.

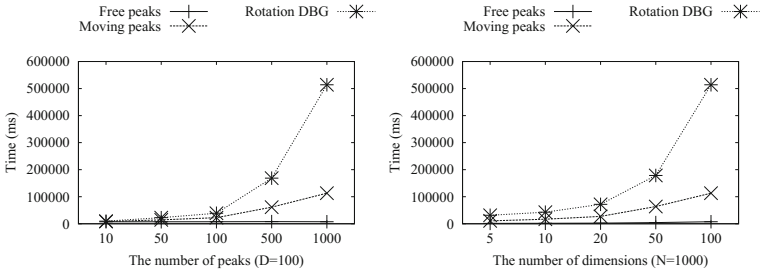

Fig. 3. Time cost of three benchmarks for evaluating one million random points in different scenarios.

The second comparison is the number of invisible peaks. Table 3 shows the average number of invisible peaks for the three benchmarks over 1,000 changes. From the results, the issue of the rotation DBG is more serious than the MPB in all test cases. Moreover, the number of invisible peaks will increase as the number of peaks increases for the MPB and the rotation DBG. This is because, in the two peer benchmarks a peak can be hidden by a higher peak with a broader basin of attraction. This issue does not exist in the FPs as each peak takes a different subspace.

4.2 The Performance of Existing Algorithms

In this subsection, 11 peer algorithms are selected. They are mQSO [3], SAMO [2], SPSO [18], AMSO [12], CPSO [22], CPSOR [11], FTMPSO [24], DynDE [13], DynPopDE [19], mNAFSA [23], and AMP/PSO [9]. For parameters of all the peer algorithms, default values suggested in their proposals are used. Note that, the parameter settings for these algorithms may be not the optimal values. The stopping criterion is 100 changes. All the results are averaged over

Table 3. The number of invisible peaks of three benchmarks with $D = 5$

Problem	The number of peaks (N)				
	10	50	100	500	1000
Free peaks	0	0	0	0	0
Moving peaks [15]	0	0.04	0.124	4.77	18.85
Rotation DBG [10]	0.12	4.12	10.23	90.22	217.24

30 independent runs of an algorithm on each problem. The offline error (E_O) [6] and the best-before-change error (E_{BBC}) are used. The offline error used in this paper is the average of the best error found every two objective evaluations and the best-before-change error is the average of the best error achieved at the fitness evaluation just before a change occurs.

A two-tailed t-test with 58° of freedom at a 0.05 level of significance was conducted for each pair of algorithms on E_O and E_{BBC}. The t-test results are given with the letters "w", "l", or "t", which denote that the performance of an algorithm is significantly better than, significantly worse than, or statistically equivalent to its peer algorithms, respectively.

Tables 4, 5 and 6 show the comparison between the chosen algorithms on the FPs with different numbers of peaks, different changing ratios, and the time-linkage feature, respectively. From the results, it can be seen that different features have very different impacts on the performance of a particular algorithm. For example, the partial change feature cause difficulties for algorithms

Table 4. Performance comparison on the FPs with different number of peaks, where the default settings in Table 1 are used.

N		AMP/PSO	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
10	E_O	**2.62**±0.1	3.11±0.081	7.75±1.7	13.2±3.5	2.7±0.075	4.14±0.25	7.53±0.3	3.97±0.13	3.23±0.049	3.36±0.7	10.2±0.97
	w,t,l	10,0,0	7,1,2	2,1,7	0,0,10	9,0,1	4,0,6	2,1,7	5,0,5	6,1,3	6,2,2	1,0,9
	E_{BBC}	**0.426**±0.12	1.64±0.12	6.25±1.4	12.2±3.7	1.22±0.11	2.39±0.25	6.26±0.28	2.96±0.088	2.22±0.079	1.95±0.91	7.48±0.87
	w,t,l	10,0,0	7,1,2	2,1,7	0,0,10	9,0,1	5,0,5	2,1,7	4,0,6	6,1,3	6,2,2	1,0,9
20	E_O	**3.04**±0.16	3.16±0.11	6.93±1.8	11.8±2.2	3.56±0.81	4.46±0.75	7.37±0.35	3.96±0.082	3.44±0.6	3.88±0.79	9.1±0.82
	w,t,l	10,0,0	9,0,1	2,1,7	0,0,10	6,2,2	4,0,6	2,1,7	5,1,4	7,1,2	5,2,3	1,0,9
	E_{BBC}	**1.2**±0.19	2.08±0.13	5.64±1.6	11.3±2.4	2.2±0.75	2.85±0.71	6.3±0.36	3.08±0.06	2.59±0.55	2.8±0.92	6.99±0.8
	w,t,l	10,0,0	8,1,1	3,0,7	0,0,10	8,1,1	4,3,3	2,0,8	4,2,4	5,2,3	4,3,3	1,0,9
30	E_O	**2.08**±0.065	2.24±0.08	5.36±1.3	9.9±2.3	3.23±0.77	3.65±0.68	5.23±0.2	2.69±0.084	3.73±0.88	2.57±0.5	8±0.65
	w,t,l	10,0,0	9,0,1	2,1,7	0,0,10	6,0,4	4,1,5	2,1,7	7,1,2	4,1,5	7,1,2	1,0,9
	E_{BBC}	**1.17**±0.069	1.65±0.077	4.24±1.2	9.24±2.7	2.28±0.69	2.51±0.66	4.36±0.21	2.03±0.053	3.08±0.84	1.92±0.51	5.83±0.51
	w,t,l	10,0,0	9,0,1	2,1,7	0,0,10	5,2,3	5,1,4	2,1,7	6,2,2	4,0,6	7,1,2	1,0,9
50	E_O	**2.73**±0.15	2.8±0.11	4.7±1.2	9.91±1.7	4.16±0.9	4.86±1.2	5.84±0.24	3.42±0.2	5.72±1.3	3.12±0.4	6.49±0.37
	w,t,l	9,1,0	9,1,0	4,2,4	0,0,10	5,1,4	4,1,5	2,1,7	7,0,3	2,1,7	8,0,2	1,0,9
	E_{BBC}	**1.7**±0.13	2.1±0.1	3.83±0.97	9.64±1.8	3±0.77	3.52±1.1	4.99±0.23	2.67±0.16	4.88±1.2	2.33±0.39	4.82±0.3
	w,t,l	10,0,0	9,0,1	4,1,5	0,0,10	6,0,4	4,1,5	1,1,8	7,0,3	1,2,7	8,0,2	2,1,7
100	E_O	**2.5**±0.16	3±0.25	4.87±1.3	12.5±1.8	9.59±2.2	5.99±1.8	6.14±0.17	4.17±0.5	9.89±2.7	2.96±0.36	7.33±0.67
	w,t,l	10,0,0	8,1,1	6,0,4	0,0,10	1,1,8	4,1,5	4,1,5	7,0,3	1,1,8	8,1,1	3,0,7
	E_{BBC}	**1.66**±0.16	2.26±0.24	3.97±1.1	12.3±1.8	8.41±2.2	4.73±1.7	5.26±0.17	2.79±0.34	9.2±2.6	2.18±0.31	5.51±0.59
	w,t,l	10,0,0	8,1,1	6,0,4	0,0,10	1,1,8	4,1,5	4,1,5	7,0,3	1,1,8	8,1,1	3,0,7
200	E_O	**1.94**±0.16	2.21±0.22	3.92±2.8	10.3±1.8	8.3±2.2	7.12±1.9	4.56±0.19	3.2±0.3	8.63±2	2.39±0.45	5.9±0.53
	w,t,l	10,0,0	8,1,1	5,2,3	0,0,10	1,1,8	3,0,7	5,1,4	6,1,3	1,1,8	8,1,1	4,0,6
	E_{BBC}	**1.32**±0.14	1.6±0.21	3.14±2.5	10.2±1.8	7±2.1	5.82±1.8	3.87±0.2	2.06±0.22	7.9±2	1.83±0.4	4.21±0.51
	w,t,l	10,0,0	9,0,1	5,1,4	0,0,10	1,1,8	3,0,7	5,1,4	7,0,3	1,1,8	8,0,2	4,0,6
w-l		119	87	-23	-120	6	-15	-43	31	-29	60	-73

Table 5. Performance comparison on the FPs with different ratios of the number of changing peaks, where the default settings in Table 1 are used.

r_c		AMP/PSO	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
0.1	E_O	1.52±0.15	0.496±0.087	11.7±2.5	16.5±5.8	**0.398**±0.3	2.12±0.25	28.9±5	1.22±0.12	0.872±0.047	1.97±3.8	5.9±1.7
	w,t,l	5,1,4	9,1,0	2,0,8	1,0,9	9,1,0	4,1,5	0,0,10	6,1,3	7,1,2	4,4,2	3,0,7
	E_{BBC}	**0.00983**±0.022	0.161±0.094	9.84±2.1	16.4±5.8	0.163±0.3	1.11±0.25	28.7±5.1	0.872±0.12	0.479±0.051	1.5±3.9	4.99±1.6
	w,t,l	10,0,0	7,2,1	2,0,8	1,0,9	7,2,1	4,1,5	0,0,10	5,1,4	6,1,3	4,5,1	3,0,7
0.3	E_O	1.68±0.13	1.25±0.097	10.9±1.5	11.8±5.1	**0.99**±0.048	2.86±0.18	21.7±1.9	2.58±0.16	1.83±0.044	2.19±0.77	6.65±1.1
	w,t,l	8,0,2	9,0,1	1,1,8	1,1,8	10,0,0	4,0,6	0,0,10	5,0,5	7,0,3	6,0,4	3,0,7
	E_{BBC}	**0.126**±0.077	0.622±0.12	9.05±1.3	11.5±5.1	0.448±0.054	1.79±0.19	21.2±1.9	1.96±0.12	1.24±0.054	1.46±0.85	5.39±1.1
	w,t,l	10,0,0	8,0,2	2,0,8	1,0,9	9,0,1	5,0,5	0,0,10	4,0,6	6,1,3	6,1,3	3,0,7
0.5	E_O	2.23±0.24	1.86±0.16	7.9±0.85	9.24±1.6	**1.57**±0.094	3.48±0.27	15.5±1.3	3.59±0.077	2.61±0.054	3.13±0.93	8.71±0.86
	w,t,l	8,0,2	9,0,1	3,0,7	1,1,8	10,0,0	5,1,4	0,0,10	4,0,6	7,0,3	5,1,4	1,1,8
	E_{BBC}	**0.482**±0.2	1.09±0.19	6.32±0.81	8.5±1.7	0.786±0.1	2.39±0.29	14.8±1.3	2.83±0.076	1.97±0.068	2.3±0.99	6.93±0.79
	w,t,l	10,0,0	8,0,2	3,0,7	1,0,9	9,0,1	5,1,4	0,0,10	4,0,6	6,1,3	5,2,3	2,0,8
0.7	E_O	2.12±0.083	2.41±0.13	8.89±1.3	12.1±1.9	**1.96**±0.15	3.59±0.17	11.5±0.7	3.79±0.08	2.73±0.059	2.91±0.74	8.68±0.96
	w,t,l	9,0,1	8,0,2	2,1,7	0,1,9	10,0,0	5,0,5	0,1,9	4,0,6	6,1,3	6,1,3	2,1,7
	E_{BBC}	**0.287**±0.1	1.3±0.15	7.34±1.4	11±2	0.918±0.15	2.26±0.19	10.5±0.75	2.75±0.097	1.94±0.065	1.82±0.78	6.74±0.87
	w,t,l	10,0,0	8,0,2	2,0,8	0,1,9	9,0,1	5,0,5	0,1,9	4,0,6	6,1,3	6,1,3	3,0,7
0.9	E_O	2.57±0.11	2.73±0.11	7.82±2.1	13.8±2.4	**2.31**±0.079	3.64±0.19	9.24±0.65	3.84±0.15	2.98±0.053	3.54±1.4	11.1±1.1
	w,t,l	9,0,1	8,0,2	3,0,7	0,0,10	10,0,0	5,1,4	2,0,8	4,1,5	7,0,3	4,2,4	1,0,9
	E_{BBC}	**0.465**±0.11	1.55±0.12	6.46±1.9	13.4±2.5	1.09±0.11	2.21±0.22	8.13±0.67	2.79±0.063	2.17±0.078	2.42±1.4	8.63±0.99
	w,t,l	10,0,0	8,0,2	3,0,7	0,0,10	9,0,1	5,2,3	2,0,8	4,1,5	5,2,3	4,3,3	1,0,9
1	E_O	**2.62**±0.1	3.11±0.081	7.75±1.7	13.2±3.5	2.7±0.075	4.14±0.25	7.53±0.3	3.97±0.13	3.23±0.049	3.36±0.7	10.2±0.97
	w,t,l	10,0,0	7,1,2	2,1,7	0,0,10	9,0,1	4,0,6	2,1,7	5,0,5	6,1,3	6,2,2	1,0,9
	E_{BBC}	**0.426**±0.12	1.64±0.12	6.25±1.4	12.2±3.7	1.22±0.11	2.39±0.25	6.26±0.28	2.96±0.088	2.22±0.079	1.95±0.91	7.48±0.87
	w,t,l	10,0,0	7,1,2	2,1,7	0,0,10	9,0,1	5,0,5	2,1,7	4,0,6	6,1,3	6,2,2	1,0,9
w-l		99	77	-62	-104	103	-1	-100	-10	40	28	-70

Table 6. Performance comparison on the FPs with the time-linkage on different numbers of peaks, where the default settings in Table 1 are used except the time-linkage feature.

N		AMP/PSO	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
10	E_O	**2.17**±0.41	8.38±5.8	18.2±9.9	4.16±3.4	2.9±0.73	4.65±1.4	26.6±14	4.81±0.86	3.22±0.57	4.21±2.2	2.77±3.6
	w,t,l	9,1,0	2,0,8	1,0,9	3,5,2	7,2,1	3,3,4	0,0,10	3,3,4	6,3,1	3,4,3	5,5,0
	E_{BBC}	**0.246**±0.23	6.38±5.7	17±9.9	3.73±3.3	1.35±0.62	2.93±1.3	26.5±14	3.7±0.71	2.27±0.39	2.73±2.3	1.93±3.7
	w,t,l	10,0,0	2,0,8	1,0,9	3,4,3	8,1,1	4,3,3	0,0,10	3,1,6	6,2,2	4,4,2	4,5,1
20	E_O	**3.01**±1.3	6.89±4.3	28.4±9	3.1±4.1	11±3.6	5.52±3.1	42.7±17	10.1±2.9	9.23±3.5	5.25±2.7	4.48±1.3
	w,t,l	9,1,0	5,2,3	1,0,9	8,2,0	2,2,6	5,3,2	0,0,10	2,2,6	2,2,6	5,3,2	6,3,1
	E_{BBC}	**1.24**±1.3	5.69±4.1	26.5±9.7	2.89±3.9	9.64±3.6	4.01±3	42.7±17	9.16±2.8	8.55±3.4	4.06±2.6	3.25±1.3
	w,t,l	10,0,0	5,2,3	1,0,9	6,3,1	2,2,6	5,4,1	0,0,10	2,2,6	2,2,6	5,4,1	6,3,1
30	E_O	**1.74**±0.45	4.68±2.3	13.3±11	5.45±3	3.32±2.2	3.95±2	6.6±7	4.1±2	2.14±1.4	3.68±1.8	6.07±3.8
	w,t,l	9,1,0	1,6,3	0,0,10	1,3,6	5,3,2	3,5,2	1,5,4	3,5,2	9,1,0	4,4,2	1,3,6
	E_{BBC}	**0.514**±0.32	3.84±2.2	11.7±10	5.12±2.7	2.38±2.1	2.91±1.9	6.59±7	3.64±1.9	1.75±1.4	2.86±1.6	5.56±3.6
	w,t,l	10,0,0	4,3,3	0,0,10	1,2,7	6,3,1	4,4,2	1,2,7	4,3,3	8,1,1	4,4,2	1,2,7
50	E_O	**2.39**±0.74	3.59±1.4	13.8±7	6.28±4	4.34±1.6	3.94±1.3	25.4±22	5.02±1.2	3.16±1	3.55±1.3	6.8±2.4
	w,t,l	10,0,0	5,4,1	1,0,9	2,2,6	4,3,3	5,3,2	0,0,10	3,2,5	7,2,1	6,3,1	2,1,7
	E_{BBC}	**1.06**±0.71	3.06±1.3	11.6±6.1	6.12±3.8	3.47±1.6	2.88±1.2	25.4±22	4.67±1.2	2.72±0.97	2.87±1.3	6.22±2.5
	w,t,l	10,0,0	5,4,1	1,0,9	2,2,6	5,3,2	5,4,1	0,0,10	3,1,6	6,3,1	5,4,1	2,1,7
100	E_O	**2.27**±0.77	2.38±0.99	13.8±3.2	7.02±2.8	5.61±2.3	3.53±1.3	17.3±14	5.55±1.7	6.21±3	3.5±1.6	6.28±2.3
	w,t,l	9,1,0	9,1,0	0,1,9	2,2,6	3,3,4	7,1,2	0,1,9	3,3,4	2,4,4	7,1,2	2,4,4
	E_{BBC}	**0.948**±0.81	1.87±0.95	11.2±2.8	6.9±2.7	4.85±2.2	2.64±1.3	17.3±14	5.35±1.7	5.84±3	2.84±1.6	5.56±2.3
	w,t,l	10,0,0	9,0,1	1,0,9	2,1,7	3,3,4	7,1,2	0,0,10	3,3,4	2,4,4	7,1,2	3,3,4
200	E_O	3.29±0.91	**2.81**±0.82	9.29±1.9	4.39±2.3	7.55±1.7	6.78±1.8	24.4±19	6.14±1.2	6.78±1.6	3.38±1.1	2.98±1.1
	w,t,l	7,2,1	9,1,0	1,0,9	6,0,4	2,2,6	2,3,5	0,0,10	3,2,5	2,3,5	7,2,1	7,3,0
	E_{BBC}	2.44±1	**2.37**±0.79	6.18±1.5	4.27±2.3	6.86±1.7	6.03±1.7	24.4±19	6.01±1.2	6.42±1.6	2.84±1.1	2.4±1.1
	w,t,l	7,3,0	7,3,0	1,4,5	6,0,4	1,3,6	1,4,5	0,0,10	2,3,5	1,4,5	7,3,0	7,3,0
w-l		109	32	-97	-10	6	20	-108	-22	17	45	8

(e.g., CPSO) that need the detection of changes. In this case, changes are hard to be detected as only a part of the fitness landscape is allowed to change. The detection will fail by monitoring the changes of the fitness of a set of solutions if these solutions are in unchange areas. As a result, mechanisms for handling

changes will be not triggered. Therefore, the performance of this type of algorithm is poor in this case. The time-linkage feature is hard for most algorithms, where their performance gets worse when this feature is enabled. For all the algorithms, AMP/PSO, which was recently proposed, performs best in terms of both performance metrics in most test cases Table 5.

5 Conclusions

This paper proposes a efficient benchmark generator, named free peaks, for constructing dynamic optimization problems. The framework is simple, feature enriched, and computing efficient. The properties and difficulties are analytical without the assistance of the visualization of the fitness landscape. The approach uses the building-blocks approach to construct a problem, therefore users can construct a problem with desired features. 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 with a quite different structure.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grant 61673355.

References

1. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
2. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Yang, S., Ong, Y.-S., Jin, Y. (eds.) Evol. Comput. Dynamic Uncertain Environ. Studies in Computational Intelligence, vol. 51, pp. 29–49. Spinger, Heidelberg (2007). doi:10.1007/978-3-540-49774-5_2
3. Blackwell, T., Branke, J.: Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Trans. Evol. Comput. **10**(4), 459–472 (2006)
4. Bosman, P.A.N.: Learning, anticipation and time-deception in evolutionary online dynamic optimization. In: Proceedings of 2005 Genetic and Evolutionary Computation Conference, pp. 39–47. ACM (2005)
5. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of 1999 IEEE Congress on Evolutionary Computation, vol. 3, pp. 1875–1882 (1999)
6. Song, T., Liu, X., Zhao, Y., Zhang, X.: Spiking neural P systems with white hole neurons. IEEE Trans. Nanobiosci. (2016). doi:10.1109/TNB.2016.2598879
7. Cobb, H.G., Grefenstette, J.J.: Genetic algorithms for tracking changing environments. In: 5th International Conference on Genetic Algorithms, pp. 523–530 (1993)
8. Grefenstette, J.J.: Evolvability in dynamic fitness landscapes: a genetic algorithm approach. In: Proceedings of 1999 IEEE Congress on Evolutionary Computation, vol. 3, p. 2038 (1999)
9. Li, C., Nguyen, T.T., Yang, M., Mavrovouniotis, M., Yang, S.: An adaptive multipopulation framework for locating and tracking multiple optima. IEEE Trans. Evol. Comput. **99**, 1 (2015)

10. Li, C., Yang, S.: A generalized approach to construct benchmark problems for dynamic optimization. In: 7th International Conference on Simulated Evolution and Learning, pp. 391–400 (2008)
11. Li, C., Yang, S.: A general framework of multipopulation methods with clustering in undetectable dynamic environments. IEEE Trans. Evol. Comput. **16**(4), 556–577 (2012)
12. Li, C., Yang, S., Yang, M.: An adaptive multi-swarm optimizer for dynamic optimization problems. Evol. Comput. **22**(4), 559–594 (2014)
13. Mendes, R., Mohais, A.S.: DynDE: a differential evolution for dynamic optimization problems. In: Proceedings of 2005 IEEE Congress on Evolutionary Computation, pp. 2808–2815 (2005)
14. Michalewicz, Z.: The emperor is naked: evolutionary algorithms for real-world applications. ACM Ubiquity **2012**, 1–13 (2012)
15. Morrison, R.W., De Jon, K.A.: A test problem generator for non-stationary environments. In: Proceedings of 1999 IEEE Congress on Evolutionary Computation, pp. 2047–2053 (1999)
16. Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evol. Comput. **6**, 1–24 (2012)
17. Nguyen, T.T., Yao, X.: Dynamic time-linkage problems revisited. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 735–744. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01129-0_83
18. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evol. Comput. **10**(4), 440–458 (2006)
19. du Plessis, M.C., Engelbrecht, A.P.: Differential evolution for dynamic environments with unknown numbers of optima. J. Glob. Optim. **55**, 1–27 (2012)
20. Tfaili, W., Siarry, P.: Fitting of an ant colony approach to dynamic optimization through a new set of test functions. Int. J. Comput. Intell. Res. **3**, 203–216 (2007)
21. Trojanowski, K., Michalewicz, Z.: Searching for optima in non-stationary environments. In: Proceedings of 1999 IEEE Congress on Evolutionary Computation, vol. 3, p. 1850 (1999)
22. Song, T., Pan, Z., Dennis, M.W., Wang, X.: Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control. Inf. Sci. **372**, 380–391 (2016)
23. Yazdani, D., Nasiri, B., Sepas-Moghaddam, A., Meybodi, M., Akbarzadeh-Totonchi, M.: mNAFSA: a novel approach for optimization in dynamic environments with global changes. Swarm Evol. Comput. **18**, 38–53 (2014)
24. Yazdani, D., Nasiri, B., Sepas-Moghaddam, A., Meybodi, M.R.: A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. Appl. Soft Comput. **13**(4), 2144–2158 (2013)