Designing Benchmark Generator for Dynamic Optimization Algorithm

HONGJIAN LI¹ AND GENG ZHANG²

¹School of Information Science and Engineering, Jishou University, Jishou, Hunan 416000, China
²School of Physics and Electromechanical Engineering, Jishou University, Jishou, Hunan 416000, China

Corresponding author: Geng Zhang (zhanggeng63@jsu.edu.cn)

This work was supported by the Scientific Research Project of Education Department of Hunan Province, China, under Grants 19C1524 and 19C1510.

ABSTRACT Real-world problems always change over time. In the past two decades, many researchers have become increasingly interested in studying evolutionary optimization algorithms for dynamic problems owing to their great importance in real-world applications. Currently, there are no unified and mature test functions for dynamic problems, and the existing dynamic test functions are incapable of simulating a real dynamic environment. This study constructs a new benchmark test function generator for simulating dynamic environments that combine existing static benchmark functions and their related properties to form new dynamic test functions. It is simple to construct and conveniently control the dynamic properties by adjusting the corresponding parameters. Through a series of comparisons with the dynamic benchmark of the generalized dynamic benchmark generator (GDBG), the simulation results show that the dynamic test functions constructed by this generator can reflect different types of changing intensity.

INDEX TERMS Dynamic environment, dynamic benchmark, dynamic optimization problems, changing intensity.

I. INTRODUCTION

Many real-world problems are dynamic problems, and the fitness functions are not only related to the decision variables but also to the changing time, which is random and time-varying. For instance, owing to the complex structure and dynamic nature of the indoor environment, the strength of the wireless signal changes accordingly in indoor positioning, and the nodes of the wireless sensor network increase or decrease over time [1]. In logistics services, the real-time information sharing of logistics resources changes over time [2]. In data mining, the contents of the mined database are constantly updated [3]. In actual operation, solid oxide fuel cells are vulnerable to performance degradation over time [4]. New job arrivals and deletion of old tasks [5]. Because of their wide practical applications, dynamic optimization problems have aroused people’s concern over the past 20 years. There have been regular annual special sessions and workshops dedicated to DOPs, such as the Congress on Evolutionary Computation (CEC) and the Genetic and Evolutionary Computation Conference (GECCO) workshops. In the vast amount of research literature [6]–[9], any problems that change over time are considered dynamic or time-varying problems. Dynamic optimization problems (DOPs) are defined as a series of static optimization problems [10]–[12]. Each dynamic function is composed of one or several static functions, and a set of dynamic sequences is obtained by applying dynamic rules to the basis function. A DOP can be defined as follows [13]:

\[
DOP = \text{optimize } f(X, t) \text{ subject to } X \in S, \quad t \in T, \quad (1)
\]

where \(S\) is the search space, \(t\) is the time, \(f(X, t) : S \times T \rightarrow R\) is the objective function that assigns a value to each possible solution \(X \in S, X = [x_1, x_2, \ldots, x_D]\) is the vector of variables to be optimized and \(D\) is the number of variables.

To study the performance of evolutionary algorithms (EAs) in dynamic environments, developing proper dynamic benchmark generator is an important task [14], [15]. There are many DOP generators that can create dynamic test functions, which can be roughly divided into three types in this paper. For the first type, the environment is switched between several stationary problems or several states of a problem. For example, Cobb et al. proposed a switching function [16] to study a genetic algorithm in the changing tracking environment. The core idea is to construct two different landscapes, \(A\) and \(B\), and there are three main ways to change. (i) Linear
translation of all the peaks in landscape $A$: (ii) the global optimum randomly moves while the rest of landscape $A$ is fixed; (iii) environmental oscillations between landscapes $A$ and $B$, resulting in significant changes in the entire landscape. The second type of DOP generator constructs dynamic environments by reshaping a predefined fitness landscape. For example, Branke et al. suggested a dynamic benchmark problem, called the “moving peaks” benchmark (MPB) [17] problem. The fitness landscape consists of a number of randomly generated peaks, each of which has its own time-varying parameters, such as width, height, and position, a dynamic environment can be constructed by changing these three parameters. De Jong et al. suggested the DF1 benchmark problem [18], [19], which involves designing a static basic function from the problem at first, and then adding the dynamic characteristics required by the benchmark generator. Yang et al. used a rotating fitness landscape to reconstruct a continuous dynamic optimization problem generator (CDOPG) [20]. The principle of the problem is to use an orthogonal transformation matrix to periodically rotate a static landscape to create dynamic instances. The degree of change depends on the size of the rotation angle. Li et al. proposed a generalized dynamic benchmark generator (GDBG) in the dynamic optimization competition of IEEE CEC2009 [21]. This generator is a combination of existing ideas about landscape shifting, rotating, and using dynamic rules to control changes. It can realize the unification of a binary space, real space, and composite space. Dynamic control was achieved by setting different rotation angles. The third type of DOP generator is the benchmark generator in the combination domain. For example, dynamic match fitness [22] is a function based on static bit matching and finds a solution that matches a given string. Dynamic changes are introduced by changing the number of bits in the match-string. In addition, Yang et al. proposed the exclusive-or (XOR) benchmark generator [23], [24], which can not only take any static binary-coded functions to generate dynamic landscape problems by XOR operations, but the landscape structure was preserved after each change. Later, Yang proposed a decomposable trap function (DTF) [15] that is based on static unitation and trap functions. The static functions are made dynamic by time-varying the static parameters and changing the scales of the function values.

Through the analysis of the above three types of DOP benchmark generators, we conclude that the benchmark problem constructed by the first type of DOP generator is based on a single problem and is limited to the basic static test function. The second type of dynamic generator is still developed for specific problems or instances of specific problems. The third type of dynamic composite system also has some defects, such as the lack of more scalable dynamic instances. Although all the above DOP benchmarks can generate dynamic environments, the changing intensity of the dynamic environment is difficult to be simulated by using the DOP generators mentioned above. For example, the rotation method can change the environment, but the changing intensity is unknown to the researchers. As the changing intensity determines the specific operators of the dynamic optimization algorithm, the DOP generators need to have the ability to simulate different changing intensity environments to test the effectiveness of the designed dynamic optimization algorithm. This paper proposes a new method for constructing a DOP generator that combines two static functions to form a new dynamic test function. When compared with the previous DOP generator methods, the proposed new method has two obvious contributions: the first contribution is that the proposed new method is simple and easy to be implemented; the second one is that it is capable of adjusting the changing intensity of the dynamic environment by controlling the parameters of two static test functions, and therefore it is convenient for the researchers to test and evaluate their designed dynamic optimization algorithms. The experimental simulation results demonstrate that the designed new DOP generator can adjust the changing intensity of the dynamic environment, allowing for environmental changes of different intensities, including two levels of gradual and abrupt changes, and different intensity types from small to large changes.

The remainder of this paper is organized as follows. Section II briefly describes the mathematical model of the benchmark generator commonly used in the field of dynamic composition. Section III introduces a new benchmark generator model based on a dynamic environment. The corresponding experimental simulation and analysis of the results are presented in Section IV. Section V summarizes this study.

II. DYNAMIC COMPOSITE BENCHMARK GENERATOR MODEL

The dynamic composition functions were extended from the static functions developed by Liang et al. [25]. Owing to its strong practicability, it is widely used [21], [26]–[29]. In short, the composition function is a new type of composition that is weighted and aggregated by $n$ basic functions. It can be described as [21]:

$$
F(x, \xi, t) = \sum_{i=1}^{n} (w_i \cdot f'_i(\frac{x - O_i(t) - O_{odi}}{\lambda_i} \cdot M_i) + H_i(t))
$$

where $F(x, \xi, t)$ is the new composition function, $\xi = (H, M, O)$ is the dynamic control parameter of the system, $t$ is the time, and $n$ is the number of basic functions used to form the composition function; the larger the value of $n$, the more complex of $F(x, \xi, t)$. $f'_i$ denotes a normalized function of the $i$-th basic function $f_i(x)$. $w_i$ denotes a normalized weight of each basic function. The parameter $\lambda_i$ is extendable factor for each $f_i(x)$. $M_i$ is the orthogonal rotation matrix, $O_i(t)$ is the shifted optimum position for each $f_i(x)$ at the time $t$, it has different values for different dimensions and is not in the center of the search range. $O_{odi}$ is the old optimum position for each $f_i(x)$. The parameter $H_i(t)$ is the height of the peak, which changes with the environment at time $t$. 

VOLUME 10, 2022
In Eq.(2), the normalized weight $w_i$ of each basic function can be calculated as follows:

$$w_i'' = \exp(-\frac{\sum_{l=1}^{D}(x_k - o_{ik} + o_{oldik})^2}{2 \cdot D \cdot \sigma_i^2})$$  \hspace{1cm} (3)

$$w_i' = \begin{cases} w_i''(l - max(w_i'')) & \text{if } w_i'' = \max(w_i'') \\ w_i''(1 - max(w_i'')) & \text{otherwise} \end{cases}$$  \hspace{1cm} (4)

$$w_i = \frac{w_i'}{\sum_{i=1}^{n}w_i'}$$  \hspace{1cm} (5)

where the parameter $\sigma_i$ is used to control coverage range, and a small $\sigma_i$ gives a narrow range for $f_i(x)$. $D$ is the number of variables.

In Eq.(2), the normalized function $f_i'(x)$ is obtained as follows:

$$f_i'(x) = C \cdot \frac{f_i(x)}{f_{i\max}(x)}$$  \hspace{1cm} (6)

where $C$ is a predefined constant, which is set to 2000 by default, and $f_{i\max}(x)$ is the estimated maximum value of $f_i(x)$, which is estimated as

$$|f_{i\max}(x)| = f_i(x_{\max} \cdot M_i)$$  \hspace{1cm} (7)

where $x_{\max}$ is the maximum of search space.

In order to avoid the distribution of the local optimal solution along the coordinate axis and retain the attributes of the benchmark function, so $M_i$ is an orthogonal rotation matrix of $D \times D$, calculated using the method provided by Salmon [30], the projection of a group of vectors $x$ on the $i - j$ plane is rotated $\theta$ degrees from the $i$-th axis to the $j$-th axis, the orthogonal rotation matrix algorithm as follows [27]:

Step 1. Randomly select $l$ dimensions ($l$ is an even number) from the $n$ dimensions to compose a vector $r = [r_1, r_2, \cdots, r_l]$. 

Step 2. For each pair of dimensions $r_{[i]}$ and dimension $r_{i+1}$, construct a rotation matrix $R_{[i\rightarrow n]}(\theta(t))$.

Step 3. An orthogonal matrix $M(t)$ is obtained as follows:

$$M(t) = R_{[1\rightarrow r_{[2]}]}(\theta(t)) \cdot R_{[3\rightarrow r_{[4]}]}(\theta(t)) \cdots R_{[r_{n-1}\rightarrow n]}(\theta(t)),$$

where the change severity of $\theta$ is set to 1, and the range of $\theta$ is $[0, 2\pi]$.

In the GDBG system [27], [31], there are six types of environmental change: small step, large step, random, chaotic, periodic, periodic change with noise. By changing the values of the above seven control parameters, seven different dynamic properties are presented. The six types of changes are described as follows.

Small step change ($T1$):

$$\Delta \xi = \alpha \cdot \|\xi\| \cdot r \cdot \xi_{\text{severity}}$$  \hspace{1cm} (8)

Large step change ($T2$):

$$\Delta \xi = \|\xi\| \cdot (\alpha \cdot \text{sign}(r) + (\alpha_{\text{max}} - \alpha) \cdot r) \cdot \xi_{\text{severity}}$$  \hspace{1cm} (9)

Random change ($T3$):

$$\Delta \xi = N(0, 1) \cdot \xi_{\text{severity}}$$  \hspace{1cm} (10)

Chaotic change ($T4$):

$$\xi(t + 1) = A \cdot (\xi(t) - \xi_{\text{min}}) \cdot (1 - (\xi(t) - \xi_{\text{min}})/\|\xi\|)$$  \hspace{1cm} (11)

Periodic change ($T5$):

$$\xi(t + 1) = \xi_{\text{min}} + \|\xi\| \cdot (\sin((2\pi/P) \cdot t + \varphi) + 1)/2$$  \hspace{1cm} (12)

Periodic change with noisy ($T6$):

$$\xi(t + 1) = (\xi_{\text{min}} + \|\xi\| \cdot (\sin((2\pi/P) \cdot t + \varphi) + 1)/2 + N(0, 1) \cdot \text{noisy\_severity}$$  \hspace{1cm} (13)

where $\|\xi\|$ is the change range of $\xi$, $\xi_{\text{severity}}$ is the change severity of $\xi$, $\xi_{\text{min}}$ is the minimum value of $\xi$, and $\text{noisy\_severity}$ is the noisy severity in recurrent with noisy change. $\alpha \in (0, 1)$ and $\alpha_{\text{max}} \in (0, 1)$ are constant values, which are set to 0.02 and 0.1 in the GDBG system. A logistics function is used in the chaotic change type, where $A$ is a positive constant between $(1.0, 4.0)$, and $P$ is the period of recurrent change and recurrent change with noise, $\varphi$ is the initial phase, $r$ is a random number in $(-1, 1)$, $\text{sign}(r)$ returns 1 when $r$ is greater than 0, returns $-1$ when $r$ is less than 0, and 0 otherwise.

$N(0, 1)$ denotes a normally distributed one-dimensional random number with a mean of zero and standard deviation.

III. A NEW BENCHMARK GENERATOR

To better test dynamic optimization algorithms, benchmark generators should be simple, easy to analyze, and easy to simulate dynamic environments [32], [33], such as a large change environment and small change environment. In addition, the degree of change should be adjusted using the parameters to facilitate testing. Despite the benchmark generators mentioned above, the requirements for designing a benchmark generator are still not fulfilled. This study proposes a new method that combines different existing static benchmark functions and forms all types of dynamic environments. This section elaborates on three subsections: construction of the dynamic generator, generation of dynamic test functions, and types of environmental change.

A. CONSTRUCTION OF DYNAMIC ENVIRONMENT

To solve the dynamic optimization problem, the first task is to model the environment. Dynamic optimization means that the fitness function value of the candidate solution changes with the environment, which is changeable and time-varying. To simulate a dynamic environment, several existing static benchmark functions (see Table 1) are applied to construct a dynamic benchmark function. The variables of the decision vectors of the benchmark functions in this study are randomly selected from the original decision vector $X$, but the dimensions of the benchmark functions are always consistent with the dimension of the original decision vector $X$ because each variable has its own special meaning in the real world. When the variable is not selected, it is given a value of 0, which means that the variable has no effect on the final fitness value. For example, a dynamic composition function $F(X)$ is
constructed by two basis functions, the Sphere and Rastrign functions, as shown in Eqs. (14), (15), and (16):

\[ F(X) = f_1(X_1) + f_2(X_2) \]  
\[ f_1(X_1) = \sum_{i=1}^{D} x_i^2 \]  
\[ f_2(X_2) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10) \]

where \( X \) is the original decision vector, and \( X_1 \) and \( X_2 \) are the decision vectors of the basis functions \( f_1(X_1) \) and \( f_2(X_2) \), respectively. \( D \) denotes the number of dimensions, and \( x \) is the variable of the basis function. The decision vector \( X_k (k = 1, 2) \) is obtained through the vector \( X_k' \) left multiplied by the orthogonal matrix \( M \), as described in equations (17) – (19). The purpose is to simulate the high correlation vector, which means that all variables in vector \( X_k \) will be affected once one variable in \( X_k' \) changes. The variables of vector \( X_k' \) are randomly selected from the original decision vector \( X \).

\[ X_k = MX_k' \]  
\[ X_k' = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1D} \\ m_{21} & m_{22} & \cdots & m_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ m_{D1} & m_{D2} & \cdots & m_{DD} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \]  
\[ X_k' = \begin{bmatrix} m_{11}x_1 + m_{12}x_2 + \cdots + m_{1D}x_D \\ m_{21}x_1 + m_{22}x_2 + \cdots + m_{2D}x_D \\ \vdots \\ m_{D1}x_1 + m_{D2}x_2 + \cdots + m_{DD}x_D \end{bmatrix} \]

In order to illustrate this, the dimension of the original decision vector \( X \) is set three for simplicity, expressed as \( X = [x_1, x_2, x_3]^T \), the ratio of the number of selected variables from the basis functions \( f_1(X_1) \) and \( f_2(X_2) \) is set 1:2 from the original decision vector. The variable chosen from the basis function \( f_1(X_1) \) is set to \( x_1 \), and the corresponding basis function \( f_2(X_2) \) selects the other two variables \( x_1 \) and \( x_3 \). Therefore, the vectors \( X_1' \) and \( X_2' \) are expressed as follows:

\[ X_1' = [0, x_2, 0]^T \]  
\[ X_2' = [x_1, 0, x_3]^T \]

The decision vectors of the basis functions \( f_1(X_1) \) and \( f_2(X_2) \) can be calculated by formulas (22) – (27) as follows:

\[ X_1 = MX_1' = \begin{bmatrix} -0.2179 & 0.9760 & 0 \\ -0.9760 & -0.2179 & 0 \\ 0.9760x_2 \\ -0.2179x_2 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ x_2 \end{bmatrix} \]  
\[ X_2 = MX_2' = \begin{bmatrix} -0.2179 & 0.9760 & 0 \\ -0.9760 & -0.2179 & 0 \\ 0.9760x_2 \\ -0.2179x_2 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ x_3 \end{bmatrix} \]

Substituting the decision vectors \( X_1 \) and \( X_2 \) into equations (15) and (16) to calculate the values of the basis functions \( f_1(X_1) \) and \( f_2(X_2) \):

\[ f_1(X_1) = (0.9760x_2)^2 + (-0.2179x_2)^2 \]  
\[ f_2(X_2) = (-0.2179x_1)^2 - 10\cos(2\pi(-0.2179x_1)) + (-0.9760x_1)^2 - 10\cos(2\pi(-0.9760x_1)) + x_3^2 - 10\cos(2\pi x_3) + 30 \]

Finally, according to equation (14), the function value of the dynamic function \( F(X) \) is expressed as:

\[ F(X) = f_1(X_1) + f_2(X_2) = (0.9760x_2)^2 + (-0.2179x_2)^2 + (-0.2179x_1)^2 - 10\cos(2\pi(-0.2179x_1)) + (-0.9760x_1)^2 - 10\cos(2\pi(-0.9760x_1)) + x_3^2 - 10\cos(2\pi x_3) + 30 \]

\( X_1' \) and \( X_2' \) constitute the current environment, which is denoted as \( \{X_1', X_2'\} \) for the convenience of description. Over time, when the environment changes, the variables \( x_i \) selected by \( X_1' \) and \( X_2' \) from the original decision vector \( X \) may be different, \( i \in [1, 2, \ldots, D] \). If the variables selected at the next moment are different from the variables at the current moment, it is considered that environmental change has occurred. The steps of the environment creation mechanism are as follows:

Step 1. Initialize the vectors of \( X_1' \) and \( X_2' \) the basis functions \( f_1(X_1) \) and \( f_2(X_2) \) are initialized to 0. Note that their dimensions must be consistent with the original decision vector dimensions.

Step 2. Determine the number of dimensions’ ratio \( N_1:N_2 \), which are randomly selected by the basis functions \( f_1(X_1) \) and \( f_2(X_2) \) from the original decision vector \( X \).

Step 3. The indexes of all variables in the original decision vector \( X \) are arranged in a random order, and a new index vector \( \Phi \) can be obtained.

Step 4. Select the index of \( N_1 \) variables from the vector \( \Phi \), obtain the vector \( \Phi_1 \), and select the indexes of the variables from \( N_1 + 1 \) to \( D \) to obtain the vector \( \Phi_2 \).

Step 5. Extract the variables from the original decision vector \( X \) one by one through the index values in the vectors \( \Phi_1 \) and \( \Phi_2 \), and then store them in \( X_1' \) and \( X_2' \), respectively.

Step 6. Finally, \( X_1' \) and \( X_2' \) are the constructed environments, denoted as \( \{X_1', X_2'\} \).

B. GENERATION OF DYNAMIC TEST FUNCTIONS

Dynamic test functions can be generated by adding multiple basic functions that have different properties. First, the
TABLE 1. Details of the basis benchmark functions [27].

| name     | function                                                                 | $x^*$ | $f^*$ |
|----------|---------------------------------------------------------------------------|-------|-------|
| Sphere   | $f(x) = \sum_{i=1}^{n} x_i^2$                                            | 0     | 0     |
| Rastrigin| $f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$                  | 0     | 0     |
| Weierstrass| $f(x) = \sum_{i=1}^{D} (\sum_{k=0}^{m_{\text{max}}} (a^k \cos(2\pi b^k (0.5 + x_i))) + D \sum_{k=0}^{m_{\text{max}}} (a^k \cos(2\pi b^k / 2)))$ | 0     | 0     |
| Griewank  | $f(x) = \frac{x^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{\lambda}}) + 1$ | 0     | 0     |
| Ackley   | $f(x) = -0.2 \exp(-0.2\sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + \epsilon$ | 0     | 0     |

$f^*$: Global optimal value of current basic function  
$x^*$: The current function obtains the optimal solution of the global optimal value

basis functions search for the global optimal solution in their respective decision spaces, and finally add the optimal solution to form the optimal solution of the dynamic composition test function. The dynamic composition test function $CF(X)$ can be described by the following formula:

$$CF(X) = \sum_{k=1}^{N} f_k(X_k)$$  (31)

where $X_k = [x_1, x_2, \ldots, x_D]$, $X_k$ is the decision vector of each basis function, $X$ is the original decision vector, $N$ is the number of basis functions used to construct the composition function. Five basic benchmark functions are used in this study. Table 1 lists the details of these five functions.

C. TYPES OF ENVIRONMENTAL CHANGE

Dynamic optimization problems in real life are changing, and the types of environmental changes may also be different. Hence, the environment constructed by the benchmark test function is mainly considered based on two features: change frequency and change intensity.

1) CHANGING FREQUENCY

The environmental change is mainly controlled by the preset number of function evaluations ($FEs$); when the number of $FEs$ reaches the preset value, the dynamic generator is activated. In this study, the change frequency is set to two grades: rapid change and slow change. The changing frequency is mainly achieved by presetting the size of the $FEs$. Here, a popular optimization algorithm called the particle swarm optimizer (PSO) is applied to optimize a test function [34], and the parameters are listed in Table 2. Figure 1 shows the convergence characteristics of the PSO to the test function. When the $FEs$ reaches the preset 1600, the environment changes for the first time; therefore, all the particles are re-initialized to find the optimal point in the new environment. It can be seen from figure 1 that the best function value has an obvious increase. As time passes, the sum of the $FEs$ continuously accumulates. The following two changes are set to take the number of $FEs$ at 6600 and 16600. Then, when the number of $FEs$ reaches 36600, the environment changes for the fourth time. In summary, the first environment lasts 1600 $FEs$, which is denoted as $\Delta FEs_1 = 1600$, and then the number of $FEs$ is increased to $\Delta FEs_2 = 5000$, $\Delta FEs_3 = 10000$, and $\Delta FEs_4 = 20000$ for each environmental change.

It is obvious that the frequency of environmental change can be determined by the size of the $\Delta FEs$ between the two environments. The smaller the $\Delta FEs$, the faster the changing frequency; conversely, the larger the $\Delta FEs$, the slower the changing frequency. Figure 1 shows the changing frequency from fast to slow and then from slow to fast.

2) CHANGING INTENSITY

In the process of dynamic problems, the changing intensity has a significant impact on the optimization problems. When a drastic environmental change is detected, the essence of the problem changes. The previously preserved optimal point cannot be applied to this new environment. If there is only a small environmental change, the optimal point of the previous problem is still useful in this new environment. In this study,
the intensity of environmental change is determined by the ratio of the number of dimensions randomly selected by the basis function from the original decision vector, and the different ratios indicate that the intensity of environmental change is different. For example, as described in Subsection A of Section III, if the dynamic test function has six dimensions and two basis functions, the current environment is denoted as \( X^1 \), and the following changed environment is \( X'' \), where the vectors \( X^1 \), \( X'' \), and \( X'' \) are expressed as follows:

\[
X^1 = [x_1, 0, 0, 0, 0, 0]^	op 
\]

\[
X'' = [0, x_2, x_3, x_4, x_5, x_6]^	op 
\]

The environment \( X^1 \) is compared with \( X'' \) where the dimensional ratio of the two basis functions selected from the original decision vector is 1:5, and the dimensional ratio of the two basis functions selected from the original decision vector is 2:4. Owing to the different nature of the basis functions and their different effects, it is clear that the environment has changed from \( X^1 \) to \( X'' \). If there is another environment \( X''' \) expressed as follows:

\[
X''' = [0, x_2, x_3, x_4, x_5, x_6]^	op 
\]

The dimensional ratio of the two basis functions selected from the original decision vector is 5:1. Compared with the environment \( X'' \), the \( X''' \) has a larger change because of the larger difference in the number of selected variables from the original decision vector; therefore, the changing intensity can be controlled by adjusting the dimensional ratio of the basis functions. The algorithm steps of the changing intensity using the two basis functions are as follows:

Step 1: Determine whether the number of \( FEs \) reaches the preset value of environmental change; if it reaches the preset value of environmental change, execute Step 2 to Step 5; otherwise, continue to search for the optimal solution in the current environment.

Step 2: Determine the dimensional ratio \( N_1:N_2 \) randomly selected by the basis function from the original decision vector \( X \).

Step 3: According to the dimensional ratio in Step 2, randomly select variables of \( N_1 \) dimensions from the original vector \( X \), record the indexes of these variables, and store them in vector \( \Phi_1 \), and the indexes of the remaining \( N_2 \) variables are stored in vector \( \Phi_2 \).

Step 4: Use index vectors \( \Phi_1 \) and \( \Phi_2 \) to extract variables from the original decision vector \( X \) individually, and then store them in \( X^1 \) and \( X^2 \), respectively.

Step 5: Get a new environment \( \{X^1', X^2'\} \).

### IV. EXPERIMENTS AND RESULTS ANALYSIS

#### A. EXPERIMENTAL DESIGN

1) SETTING TEST FUNCTIONS AND ALGORITHM

From the descriptions in Section III, the dynamic test functions can be obtained by combining basis functions with different properties. In the dynamic experiments, six different dynamic composition functions (CFs) are created using the basis functions listed in Table 1, and the PSO algorithm is applied to simulate these dynamic composition functions. Since the PSO algorithm converges slower in the later stages of the search, we introduced a dynamic adjustment strategy based on the betarnd \((a, b)\) function [35] for the inertia weight of the PSO algorithm, the specific expression for its dynamic inertia weights is as follows:

\[
w = w_{\min} + (w_{\max} - w_{\min})\cos\left(\frac{\pi t}{2T_{\max}}\right) + \sigma \times \text{betarnd}(a, b)\]

where \( T_{\max} \) denotes the maximum number of particle iterations, \( t \) denotes the \( t \)-th iteration count, and \( w_{\max} \) and \( w_{\min} \) are the maximum and minimum values of the inertia weights, respectively. \( \sigma \) is the inertia adjustment factor, and \( \text{betarnd}(a, b) \) generates a random number that obeys a beta distribution. \( a \) and \( b \) are the two parameters. The parameters of the PSO algorithm are listed in Table 2. The six new dynamic composition functions are as follows:

\[
CF_1(X) = f_{\text{sphere}}(X_1) + f_{\text{rastrigin}}(X_2) 
\]

\[
CF_2(X) = f_{\text{weiweistrass}}(X_1) + f_{\text{griewank}}(X_2) 
\]

\[
CF_3(X) = f_{\text{rastrigin}}(X_1) + f_{\text{weiweistrass}}(X_2) 
\]

\[
CF_4(X) = f_{\text{sphere}}(X_1) + f_{\text{griewank}}(X_2) 
\]

\[
CF_5(X) = f_{\text{weiweistrass}}(X_1) + f_{\text{ackley}}(X_2) 
\]

\[
CF_6(X) = f_{\text{rastrigin}}(X_1) + f_{\text{ackley}}(X_2) 
\]

The number of environmental changes is set to four, that is, when the number of \( FEs \) reaches 1600, 6600, 16600, and 36600, the environmental change will occur; therefore, the \( \Delta FEs \) for each change are 5000, 10000, and 20000, respectively. In summary, the changing frequency is gradually reduced. Corresponding to each change, the dimensional ratios randomly selected by the basis functions \( f_1(X_1) \) and \( f_2(X_2) \) from the original decision space are set as 5:25, 10:20,
15:15, 20:10, and 25:5, respectively. Therefore, the change in the intensity gradually increases. Fig.2 shows a flowchart of the designed dynamic environment.

2) DETECTING ENVIRONMENTAL CHANGES
The algorithm is required not only to search for the optimal solution, but also to detect whether the environment has changed precisely. There are two frequently used methods for detecting environmental changes [36]. The first is to use a re-evaluation method to detect changes in the environment. The detector can be a sub-population [37], it can also be a global optimal solution and a global sub-optimal solution [38], or a group of solutions is randomly selected in the solution space [39]. The algorithm re-evaluates the fitness value of the detector before each iteration; when the algorithm detects that the fitness of the detector has changed, it is considered that the solution environment is changed. The second is to detect environmental changes based on algorithm behavior. For example, a similarity detection operator [40] determines whether the environment has changed. In [41], where the swarm was divided into a tree-based hierarchy of sub-swarms, environmental change was detected based on the observation of changes in the hierarchy itself. In [42], a memory-based linear regression method was used to detect when and how the environment changed. In this study, a fixed set of detectors is used to detect the environmental changes. Before updating the global optimal value of each particle, the fitness values need to be re-evaluated. If their fitness values have changed, it indicates that an environmental change has occurred; otherwise, the environment has not changed.

3) CHANGING RESPONSE
In general, there are some connections among environments, and the previous information can provide an effective experience for the algorithm to search in a new environment. Therefore, it can be used to improve the search efficiency of the algorithm; however, if the environment has changed drastically, it means that the previous information cannot provide any help to the algorithm’s decision-making in the new environment. For this type of situation, re-initialization is a better choice. Therefore, when a change is detected in the environment, reasonable dynamic measures should be adopted to respond to changes. Many methods have been proposed. In [43], a strategy for decomposing decision variables was proposed, through which all the decision variables were divided into two groups according to the interval similarity between each decision variable and interval parameters. Then, two sub-populations were utilized to cooperatively optimize the decision variables in the two groups. Meanwhile, a strategy based on the change intensity and a random mutation strategy were employed to rapidly track the changing Pareto front of the optimization problem. In [44], a cooperative co-evolutionary strategy based on environmental sensitivity was proposed. In this strategy, all decision variables were first divided into two sub-components according to their interrelationships with the environment. Furthermore, differential prediction and Cauchy mutations were used to collaboratively optimize the two sub-components to accelerate their response to environmental changes. Other methods, such as memory-based methods, enhancing diversity, and introducing multiple groups [45]. As the main purpose of this study is to design dynamic test functions, a simple re-initialization method is applied for all change situations.

B. RESULTS AND ANALYSIS
The simulation results of six CF functions in different environments are shown in Figs.3 – 8. The mean and standard deviation of the initial values of each environmental change within the 30 runs are listed in Table 3. Meanwhile, a two-tailed t-test with 58 degrees of freedom at a 0.05 level of significance is applied to analyze the initial values’ difference between the former environment and the subsequent latter environment. The results are listed in Table 4. By observing the experimental results from Fig.3 to Fig.8, it is obvious that the consumption of FEIs is increased for each environmental change, which is in accordance with our previous experimental setting. As for the changing intensity, environment 1 takes five variables from the basis function $f_1(X_1)$ and 25 variables from $f_2(X_2)$. As more of the variables are involved from $f_1(X_1)$ and less of those from $f_2(X_2)$ for each environmental change, this means that the dynamic

![FIGURE 2. Flow chart of dynamic environment mechanism.](image-url)
environment is more influenced by $f_1(X_1)$ and less influenced by $f_2(X_2)$. The initial values shown in Table 3 are consistent with this statement. The gradual increase in the initial value indicates that not only does the environment change, but the changing intensity increases with the dimensional ratio. In other words, the parameter settings of the dimensional ratio can be used to adjust the changing intensity of the
designing benchmark generator for dynamic optimization algorithm. For example, a large change in the environment can be implemented by drastically changing the dimensional ratio from environment 1 to environment 5, while a small change in the environment is achieved by adjusting the dimensional ratio within a small change.

In the dynamic environment of the GDBG, the re-initialization method is also adopted to respond to environmental changes. The environment changes when the FEs reaches 4000, 8000, 12000, and 16000. Fig. 9 shows the composition of the sphere’s function in GDBG’s small step change type, and Fig. 10 shows the ones with GDBG’s large step change type. These test functions include the composition of Sphere’s function (\(F_2\)), composition of Rastrign’s function (\(F_3\)), composition of Griewank’s function (\(F_4\)), composition of Ackly’s function (\(F_5\)), and hybrid composition function (\(F_6\)), the results of the \(t\)-test analysis are presented in Table 7. Based on these two figures and Table 6, it is clear that the GDBG method can make a small change, but it is very difficult to make a larger step change, and the degree of change in intensity cannot be controlled through parameter setting.

Comparing Tables 4 and 7, it can be seen that the change in the initial values created by the proposed method is obvious, but the change created by GDBG in the large dynamic environment. For example, a large change in the environment can be implemented by drastically changing the dimensional ratio from environment 1 to environment 5,
The GDBG’s large step change type is helpful for researchers to simulate dynamic situations and strategies. Therefore, this new dynamic benchmark generator is necessary, and different dynamic situations need to adopt different simulating the dynamic environment. In real-world dynamic changing intensity of each environment, which is critical for functions, the designed benchmark generator can adjust the dimensional ratio that is randomly selected from the basis terms of controlling the changing intensity. Relying on the benchmark generator has obvious advantages in simulating the dynamic environment and the following latter one.

**REFERENCES**

[1] M. M. Atia, A. Noureldin, and M. J. Korenberg, “Dynamic online-calibrated radio maps for indoor positioning in wireless local area networks,” *IEEE Trans. Mobile Comput.*, vol. 12, no. 9, pp. 1774–1787, Sep. 2013.

[2] S. Liu, Y. Zhang, Y. Liu, L. Wang, and X. V. Wang, “An ‘Internet of Things’ enabled dynamic optimization method for smart vehicles and logistics tasks,” *J. Cleaner Prod.*, vol. 215, pp. 806–820, Apr. 2019.

[3] S. Yang, Y.-S. Ong, and Y. Jin, Eds., “Evolutionary computation in dynamic and uncertain environments,” in *Proc. IEEE Comput. Intell. Soc., Evol. Comput. Tech. Committee, Stud. Comput. Intell.*, vol. 51. Berlin, Germany: Springer-Verlag, 2007.

[4] X. Wu, L. Xu, J. Wang, D. Yang, F. Li, and X. Li, “A diagnostic-based dynamic optimization strategy for a degraded solid oxide fuel cell,” *Sustain. Energy Technol. Assessments*, vol. 39, Jun. 2020, Art. no. 100682.

[5] F. B. Ozsoydan and A. Baykasoglu, “Quantum firefly swarms for multimodal dynamic optimization problems,” *Expert Syst. Appl.*, vol. 115, pp. 189–199, Jan. 2019.

[6] P. A. N. Bosman, “Learning, anticipation and time-deception in evolutionary online dynamic optimization,” in *Proc. Workshops Genet. Evol. Comput. (GECCO)*, 2005, pp. 1165–1172.

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

[8] T. Bäck, “On the behavior of evolutionary algorithms in dynamic environments,” in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1998, pp. 446–451.

[9] Y. G. Woldeisenbet and G. G. Yen, “Dynamic evolutionary algorithm with variable relocation,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 500–513, Jun. 2009.

[10] Z. Yuan, Y. Yuan, T. Luo, and J. Qon, “A hybrid immune algorithm for solving dynamic optimization problems,” in *Proc. 30th Chin. Control Decis. Conf. (CCDC)*, Jun. 2018, pp. 1174–1180.

[11] V. S. Aragon and S. C. Esquivel, “An evolutionary algorithm to track changes of optimum value locations in dynamic combinatorial optimizations,” *J. Comput. Sci. Technol.*, vol. 4, no. 3, pp. 127–133, Oct. 2004.

[12] P. Rohlfshagen and X. Yao, “Dynamic combinatorial optimisation problems: An analysis of the subset sum problem,” *Soft Comput.*, vol. 15, no. 9, pp. 1723–1734, Sep. 2011.

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

[14] D. Yazdani, J. Branke, M. N. Omidvar, X. Li, C. Li, M. Mavrovouniotis, T. T. Nguyen, S. Yang, and X. Yao, “IEEE CEC 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark,” 2021, arXiv:2106.06174.

[15] S. Yang, “Constructing dynamic test environments for genetic algorithms based on problem difficulty,” in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2, Jun. 2004, pp. 1262–1269.

[16] H. G. Cobb and J. J. Grefenstette, “Genetic algorithms for tracking changing environments,” in *Proc. 5th Int. Conf. Genet. Algorithms, 1993*, pp. 523–530.

[17] J. Branke, “Memory enhanced evolutionary algorithms for changing optimization problems: An analysis of the subset sum problem,” *Soft Comput.*, vol. 15, no. 9, pp. 1723–1734, Sep. 2011.

[18] R. W. Morrison and K. A. De Jong, “A test problem generator for non-stationary environments,” in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3, Jul. 1999, pp. 1875–1882.

[19] R. W. Morrison and K. A. De Jong, “A test problem generator for non-stationary environments,” in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3, Jul. 1999, pp. 2047–2053.

[20] R. Morrison, “Performance measurement in dynamic environments,” in *Proc. GECCO Workshop Evol. Algorithms Dyn. Optim. Problems*, 2003, pp. 5–8.

[21] R. Tinos and S. Yang, “Continuous dynamic problem generators for evolutionary algorithms,” *IEEE Congr. Evol. Comput. (CEC)*, Sep. 2007, pp. 236–243.

[22] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, “Automatic benchmark generation for CEC2009 competition on dynamic optimization,” *Univ. Leicester, Univ. Birmingham, Birmingham, U.K.*, Tech. Rep. 2008, 2008, pp. 900–909, vol. 11, no. 5.

[23] S. A. Stanhope and J. M. Daida, “Optimal mutation and crossover rates for genetic algorithm operating in a dynamic environment,” in *Proc. 7th Int. Conf. Evol. Program.*, in Lecture Notes in Computer Science, vol. 1447, Conf. Evol. Program., 2003, pp. 900–909, vol. 11, no. 5.

[24] T. T. Nguyen, S. Yang, and X. Yao, “The dynamic knapsack problem revisited: A new benchmark problem for dynamic combinatorial optimisation,” *Appl. Evol. Comput.*, vol. 5, pp. 4548, pp. 745–754, Apr. 2009.
S. Zhang, Z. Li, S. Chen, and R. Li, “Novel composition test functions for numerical global optimization,” in Proc. IEEE Swarm Intell. Symp. (SIS), Jun. 2005, pp. 68–75.

P. N. Suganthan, N. Hansen, J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, “Problem definitions and evaluation criteria for the CEC’2005 special session on real-parameter optimization,” Nanyang Technol. Univ., Singapore, Tech. Rep. 2005005, 2005.

C. Li and S. Yang, “A generalized approach to construct benchmark problems for dynamic optimization,” in Proc. 7th Int. Conf. Simulated Evol., in Lecture Notes in Computer Science, vol. 5361, 2008, pp. 391–400.

X. Li, A. Engelbrecht, and M. G. Epitropakis, “Benchmark functions for CEC’2013 special session and competition on niching methods for multimodal function optimization,” Evol. Comput. Mach. Learn. Group, RMIT Univ., Melbourne, VIC, Australia, Tech. Rep., 2013. [Online]. Available: http://titan.csit.rmit.edu.au/~e46507/cec13-niching/competition/

B.-Y. Qu and P. N. Suganthan, “Novel multimodal problems and differential evolution with ensemble of restricted tournament selection,” in Proc. IEEE Conge. Evol. Comput., Jul. 2010, pp. 1–7.

R. Salomon, “Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions,” Bio Syst., vol. 39, pp. 263–278, Jun. 1996.

S. Yang, T. T. Nguyen, and C. Li, “Evolutionary dynamic optimization: Test and evaluation environments,” in Proc. Evol. Comput. Dyn. Optim. Problems, in Studies in Computational Intelligence, vol. 490, 2013, pp. 3–37.

D. Yazdani, M. N. Omidvar, R. Cheng, J. Branke, T. T. Nguyen, and X. Yao, “Benchmarking continuous dynamic optimization: Survey and generalized test suite,” IEEE Trans. Cybern., early access, Aug. 14, 2020, doi: 10.1109/TCYB.2020.3011828.

C. Li, “An efficient benchmark generator for dynamic optimization problems,” in Proc. Int. Conf. Bio-Inspired Comput., Theories Appl., Bio-Inspired Comput.-Theories Appl. in Communications Computer and Information Science, vol. 682, Oct. 2016, pp. 60–72.

R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in Proc. 6th Int. Symp. Micro MHS, Oct. 1995, pp. 39–43.

Y. Huang, H. Lu, K. Xu, and W. Shen, “Simplified mean particle swarm optimization algorithm with dynamic adjustment of inertia weight,” J. Chin. Comput. Syst., vol. 39, no. 12, pp. 2590–2595, 2018.

Z. Liang, S. Zheng, Z. Zhu, and S. Yang, “Hybrid of memory and prediction strategies for dynamic multiobjective optimization,” Inf. Sci., vol. 485, pp. 200–218, Jun. 2019.

K. Deb and S. Karthik, “Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling,” in Proc. EMO, in Lecture Notes in Computer Science, vol. 4403, 2007, pp. 803–817.

S. Yang and C. Li, “A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments,” IEEE Trans. Evol. Comput., vol. 14, no. 6, pp. 959–974, Dec. 2010.

A. Meier, O. Kramer, K. Sim, and P. Kaufmann, “Prediction with recurrent neural networks in evolutionary dynamic optimization,” in Proc. Int. Conf. Appl. Evol. Comput., vol. 10784, 2018, pp. 848–863.

S. Zhang, Z. Li, S. Chen, and R. Li, “Dynamic multi-objective optimization algorithm based on ecological strategy,” J. Comput. Res. Develop., vol. 51, no. 6, pp. 1313–1330, 2014.

S. Janson and M. Middendorf, “A hierarchical particle swarm optimizer for noisy and dynamic environments,” Genet. Program. Evolvable Mach., vol. 7, no. 4, pp. 329–354, Nov. 2006.

A. Simões and E. Costa, “Improving prediction in evolutionary algorithms for dynamic environments,” in Proc. 11th Annu. Conf. Genet. Evol. Comput. (GECCO), Jul. 2009, pp. 875–882.

D. Gong, B. Xu, Y. Zhang, Y. Guo, and S. Yang, “A similarity-based cooperative co-evolutionary algorithm for dynamic interval multiobjective optimization problems,” IEEE Trans. Evol. Comput., vol. 24, no. 1, pp. 142–156, Feb. 2020.

B. Xu, Y. Zhang, D. Gong, Y. Guo, and M. Rong, “Environment sensitivity-based cooperative co-evolutionary algorithms for dynamic multi-objective optimization,” IEEE/ACM Trans. Comput. Biol. Bioinf., vol. 15, no. 6, pp. 1877–1890, Nov. 2018.

M. Mavrovouniotis, C. Li, and S. Yang, “A survey of swarm intelligence for dynamic optimization: Algorithms and applications,” Swarm Evol. Comput., vol. 33, pp. 1–17, Apr. 2017.

HONGJIAN LI is currently a postgraduate student with the School of Information Science and Engineering, Jishou University, Jishou, China. His current research interest includes dynamic optimization algorithms.

GENG ZHANG received the B.S. degree in electrical engineering and the M.E. degree in control theory and engineering from the Lanzhou University of Technology, in 2004 and 2007, respectively. She received the Ph.D. degree in electrical engineering from the University of Macau in 2017. From 2017 to 2019, she was an Assistant Researcher with the Southern University of Science and Technology, Shenzhen, China. She is currently a Lecturer with the School of Physics and Electromechanical Engineering, Jishou University, Jishou, China. Her current research interests include optimization and neural networks. She was the winner of the competition of single objective bound constrained optimization held in IEEE World Congress on Computational Intelligence (WCCI), in 2018.

* * *