Swarm and Evolutionary Computation 50 (2019) 100559

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Evolutionary dynamic constrained optimization: Test suite construction and algorithm comparisons

Yong Wang [a], Jian Yu [a], Shengxiang Yang [b,*], Shouyong Jiang [c], Shuang Zhao [d,**]

[a] School of Automation, Central South University, Changsha, 410083, China
[b] School of Computer Science and Informatics, De Montfort University, Leicester, LE1 9BH, UK
[c] School of Computer Science, University of Lincoln, Lincoln, LN6 7TS, UK
[d] Department of Dermatology, Xiangya Hospital, Central South University, Changsha, 410008, China

ARTICLE INFO

Keywords:
Dynamic constrained optimization
Evolutionary algorithms
Benchmark test functions
Performance comparison
Constraint-handling technique

ABSTRACT

Many real-world applications can be modelled as dynamic constrained optimization problems (DCOPs). Due to the fact that objective function and/or constraints change over time, solving DCOPs is a challenging task. Although solving DCOPs by evolutionary algorithms has attracted increasing interest in the community of evolutionary computation, the design of benchmark test functions of DCOPs is still insufficient. Therefore, we propose a test suite for DCOPs. A dynamic unconstrained optimization benchmark with good time-varying characteristics, called moving peaks benchmark, is chosen to be the objective function of our test suite. In addition, we design adjustable dynamic constraints, by which the size, number, and change severity of the feasible regions can be flexibly controlled. Furthermore, the performance of three dynamic constrained optimization evolutionary algorithms is tested on the proposed test suite, one of which is presented in this paper, named dynamic constrained optimization differential evolution (DyCODE). DyCODE includes three main phases: 1) the first phase intends to enter the feasible region from different directions promptly via a multi-population search strategy; 2) in the second phase, some excellent individuals chosen from the first phase form a new population to search for the optimal solution of the current environment; and 3) the third phase combines the memory individuals of the first two phases with some randomly generated individuals to re-initialize the population for the next environment. From the experiments, one can understand the strengths and weaknesses of the three compared algorithms for solving DCOPs in depth. Moreover, we also give some suggestions for researchers to apply these three algorithms on different occasions.

1. Introduction

Many dynamic optimization problems have constraints, which are called dynamic constrained optimization problems (DCOPs) [1]. It is common to face a considerable number of DCOPs in the real-world applications [2–7]. For DCOPs, objective function and/or constraints change over time. In general, different DCOPs may have different mathematical expressions. In this paper, DCOPs are formulated as follows:

$$\begin{aligned} &\text{maximize } f(\vec{x},t),\ \vec{x}=(x_1,\ldots,x_D)\in S,\ L_i<x_i<U_i \\ &\text{subject to: } \begin{cases} g_j(\vec{x},t)\le 0,\ j=1,\ldots,l \\ h_j(\vec{x},t)=0,\ j=l+1,\ldots,m \end{cases} \end{aligned} \tag{1}$$

where t is the discrete time instance or the environmental variable, $\vec{x}$ is the decision vector, x_i is the ith decision variable, L_i and U_i are the lower and upper bounds of x_i, respectively, D is the number of decision variables, $S=\prod_{i=1}^{D}[L_i,U_i]$ is the decision space, $f(\vec{x},t)$ is the objective function, $g_j(\vec{x},t)$ is the jth inequality constraint, $h_j(\vec{x},t)$ is the jth equality constraint, and l and $(m-l)$ are the number of inequality and equality constraints, respectively.

In this paper, we are interested in the occasion that both objective function and constraints change over time simultaneously. Due to the dynamic change of both objective function and constraints, it is very challenging to solve DCOPs. In the early stage of dynamic constrained optimization, artificial test suite can help researchers judge the performance of an algorithm when solving DCOPs. Afterward, researchers

* Corresponding author.
** Corresponding author.
E-mail addresses: ywang@csu.edu.cn (Y. Wang), syang@dmu.ac.uk (S. Yang), shuangxy@csu.edu.cn (S. Zhao).

https://doi.org/10.1016/j.swevo.2019.100559
Received 9 March 2019; Received in revised form 4 June 2019; Accepted 24 July 2019
Available online 30 July 2019

can identify the advantages and disadvantages of an algorithm, with the aim of improving its performance. Moreover, as pointed out by Branke [8], artificial test suite should be easy to describe, easy to analyze, and also parameter-tunable. Nevertheless, when solving DCOPs by evolutionary algorithms (EAs), one of the main issues is the lack of well-established benchmark test functions. Although researchers have made some attempts [9–13], this issue does not attract as much attention as it deserves.

Based on the above consideration, we construct a new test suite for DCOPs. In the proposed test suite, a well-known dynamic unconstrained optimization benchmark, called moving peaks benchmark (MPB) [8], is considered as the objective function due to its outstanding time-varying characteristics. In addition, we construct adjustable dynamic constraints. Specifically, the center of each feasible region is the same with the center of one of the peaks in the decision space, and thus each feasible region follows the movement of one peak. We also take two different situations into account during the design of constraints. One is the number of the feasible regions and the other is the change severity of the feasible regions. By combining the above two situations, we design six test instances. The advantages of our test suite are the following: the global and local optima are known, the size and number of the feasible regions are adjustable, the change severity of the feasible regions is controllable, and the dimension of the decision space and the number of constraints are scalable. Furthermore, researchers can easily control the difficulty of our test suite by adjusting some parameters.

Subsequently, the performance of three dynamic constrained optimization EAs (DCOEAs) is assessed based on our test suite. One of these DCOEAs is proposed in this paper, called dynamic constrained optimization differential evolution (DyCODE). DyCODE can be divided into three phases. The purpose of the first phase is to find feasible solutions from different directions via a multi-population search strategy. Once the proportion of feasible solutions in all the subpopulations reaches a predefined value, some excellent individuals in each subpopulation will be preserved into the second phase. In the second phase, these individuals are aggregated into a new population to search for the optimal solution of the current environment. If a change has been detected, the third phase begins. In the third phase, DyCODE combines the memory individuals of the first two phases with some randomly generated individuals to re-initialize a new population, with the aim of tracking the next environment. The experimental results demonstrate that different algorithms have different strengths and drawbacks, which reveals that our test suite can distinguish these three algorithms and provide suggestions for researchers to apply them on different occasions.

The rest of this paper is organized as follows. Section 2 reviews existing test functions and the development of EAs for solving DCOPs. Section 3 introduces the proposed test suite. Section 4 illustrates the details of three DCOEAs. Section 5 presents a comprehensive comparison among these three DCOEAs on the proposed test suite. Finally, Section 6 concludes the paper and points out the future work.

2. Related work

2.1. Test functions of DCOPs

The first research on constructing test functions of DCOPs is conducted by Liu [11] in 2008. In Ref. [11], three test functions are designed, in which objective function and/or constraints are dependent on the time variable. In addition, the time period is divided into several equal subperiods. In each subperiod, a DCOP is approximated as a static constrained optimization problem. Richter [10] suggested a framework to construct test functions of DCOPs. In this framework, an $n-$dimensional "field of cones on a zero plane" is viewed as the objective function, which is subject to dynamic norm-based constraints. With respect to the dynamic norm-based constraints, only one of the four parameters (i.e., the center coordinates) varies with time. Nguyen et al. [14] designed a set of 18 test functions called G24. The main idea is to generalize a static objective function/constraint to its dynamic version by replacing each static parameter with a time-dependent expression. Zhang et al. [13] extended nine and 13 static benchmark test functions chosen from [15,16] to DCOPs, respectively. Similar to G24, these static benchmark test functions are associated with time-varying parameters, which change periodicity or monotonicity. Bu et al. [12] proposed two modified test sets of DCOPs. One test set is based on MPB [8], in which the global optimum and the size of each feasible region can be calculated accurately. The other test set is modified from G24, in which more constraints are added to G24, thus making the feasible region smaller.

In this paper, we consider that test functions designed for DCOPs should contain the following five characteristics: 1) Scalability. The number of decision variables and constraints should be scalable; 2) Adjustability. The shape and size of the feasible regions can be adjusted freely; 3) Multi-modality, which can test the search ability of a DCOEA to deal with a complex objective function with multiple local optimal solutions; 4) Change severity of the feasible regions, such as a slight change or a drastic change; and 5) Global and local optima known. Under this condition, the performance of a DCOEA can be evaluated conveniently. However, existing test functions do not satisfy all these five characteristics. For example, the number of decision variables of the test functions constructed in Ref. [14] cannot be scalable. The adjustability of the test functions designed in Refs. [11,13] is not good. The test functions developed in Refs. [11,14] only consider unimodal problems. Moreover, existing test functions do not consider the change severity of the feasible regions. Despite the test functions in Ref. [13] involve different types of DCOPs, they lack some important information, such as the global and local optimal solutions, and the dynamics of the feasible regions. The current situation motivates us to design a test suite satisfying all the above five characteristics.

2.2. EAs for solving DCOPs

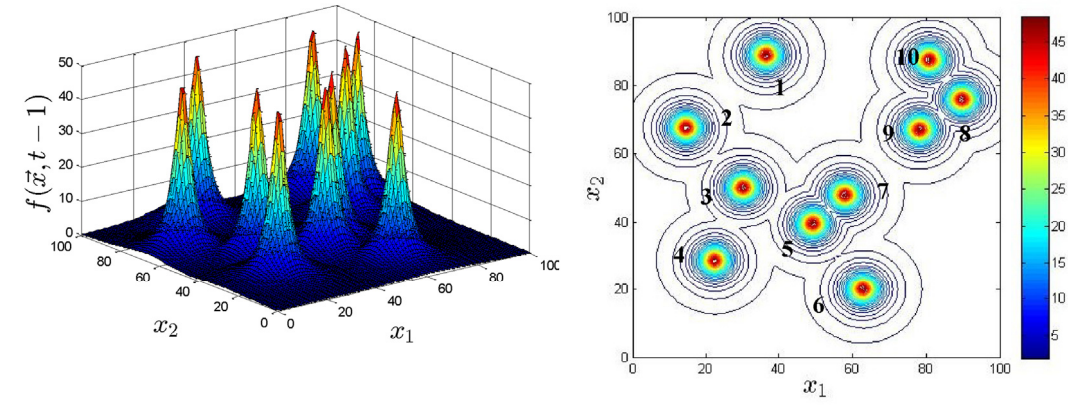
In terms of EAs for solving DCOPs, the related work can be roughly classified into four categories, i.e., revising original static constrained optimization EAs, repair methods, combining dynamic unconstrained optimization strategies with constraint-handling techniques, and other methods. Next, we briefly introduce them.

2.2.1. Revising original static constrained optimization EAs

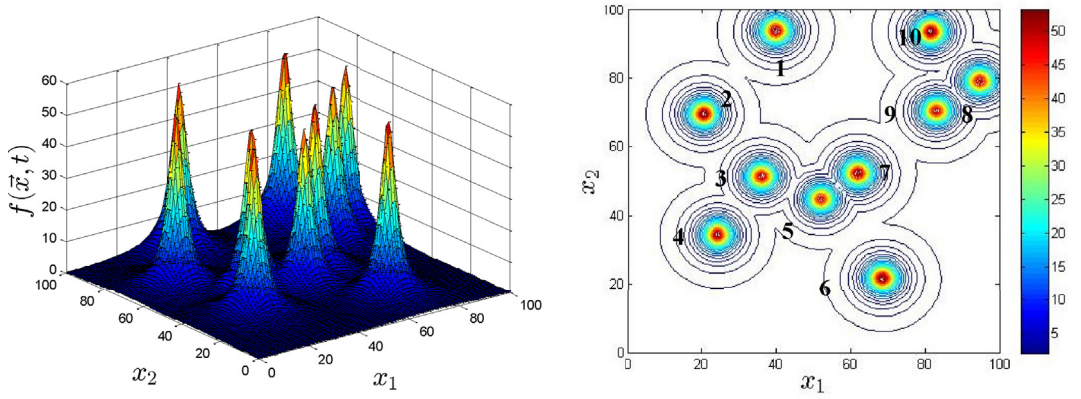
Liu [11] proposed a dynamic constrained optimization particle swam optimization to solve DCOPs. In this method, a new fitness function with a self-adaptive inertia weight is designed. Singh et al. [17] generalized the infeasibility driven EA [18] to solve DCOPs, in which infeasible individuals are utilized to approach the constrained optimum from both feasible and infeasible sides of the decision space. A dynamic constrained T-Cell with few parameters is proposed by Aragn et al. [19], which is an adaptation from Ref. [20].

Table 1
Default parameter settings of MPB.

Parameter	Value
Number of peaks (p)	10
Change frequency (U)	5000 FEs
Height_severity	7.0
Width_severity	1.0
Peak shape	Cone
Shift length (s)	$\{1.0, 2.0, 3.0, 4.0, 5.0, 6.0\}$
Correlation coefficient (λ)	0
Decision space (S)	$[0, 100]^D$
W	[1, 12]
H	[30, 70]
Initial height of each peak	50



(a) Environment $(t-1)$



(b) Environment t

Fig. 1. Fitness landscape and contours of MPB at environment $(t-1)$ and environment t in a two-dimensional decision space.

2.2.2. Repair methods

In 2009, Nguyen and Yao [14] combined genetic algorithm with the repair method and presented an algorithm called repairGA. In repairGA, there are two populations, namely, search population and reference population. It is necessary to emphasize that the reference population entirely contains feasible solutions. The core idea of repairGA is to use feasible solutions randomly chosen from the reference population to repair infeasible solutions in the search population. A modified version of the repair method with gravitational search algorithm is developed by Pal et al. [21]. In this algorithm, a feasible solution closest to an infeasible solution is employed to repair this infeasible solution. A dynamic differential evolution (DE) with combined variants (DDECV) is introduced in Ref. [22], which adopts two differential evolution variants to generate new individuals. Afterward, a novel repair method [23], which does not require feasible solutions as reference and is inspired by the differential mutation, is added to DDECV for tackling DCOPs.

2.2.3. Combining dynamic unconstrained optimization strategies with constraint-handling techniques

As mentioned in Ref. [14], four kinds of common dynamic unconstrained optimization strategies are used to solve DCOPs, i.e., introducing diversity, maintaining diversity, tracking the previous optima, and the memory-based strategy. In terms of introducing diversity, the premise is that the change can be detected. Once the change has been detected, an algorithm can adopt some techniques to add the diversity of population. One of the representative is HyperM [24], which adaptively modifies levels of mutation to enhance the diversity of population. In contrast to the introducing diversity-based strategy, the maintaining diversity-based strategy can keep the population diversity without detecting the change explicitly. For instance, RIGA [25] randomly generates and immigrates some individuals to the population at each generation. With respect to tracking the previous optima, the aim is to track the environmental optimum as it changes over time [24]. In the memory-based strategy, some useful information is exploited for the next environment. There are two kinds of memory strategies, namely, explicit memory in which some good individuals are saved directly and abstract memory in which some information about the current environment is saved implicitly. In 2010, based on the idea of abstract memory from Ref. [26], Richter et al. [10] constructed two memory matrices. When an environmental change is detected, these two memory matrices are utilized to re-initialize the population. In most cases, single dynamic unconstrained optimization strategy could not solve DCOPs effectively. Therefore, many DCOEAs integrate several dynamic unconstrained optimization strategies together. For example, DDECV utilizes the maintaining diversity-based strategy to keep diversity at each generation and the memory-based strategy to save some excellent individuals into the next

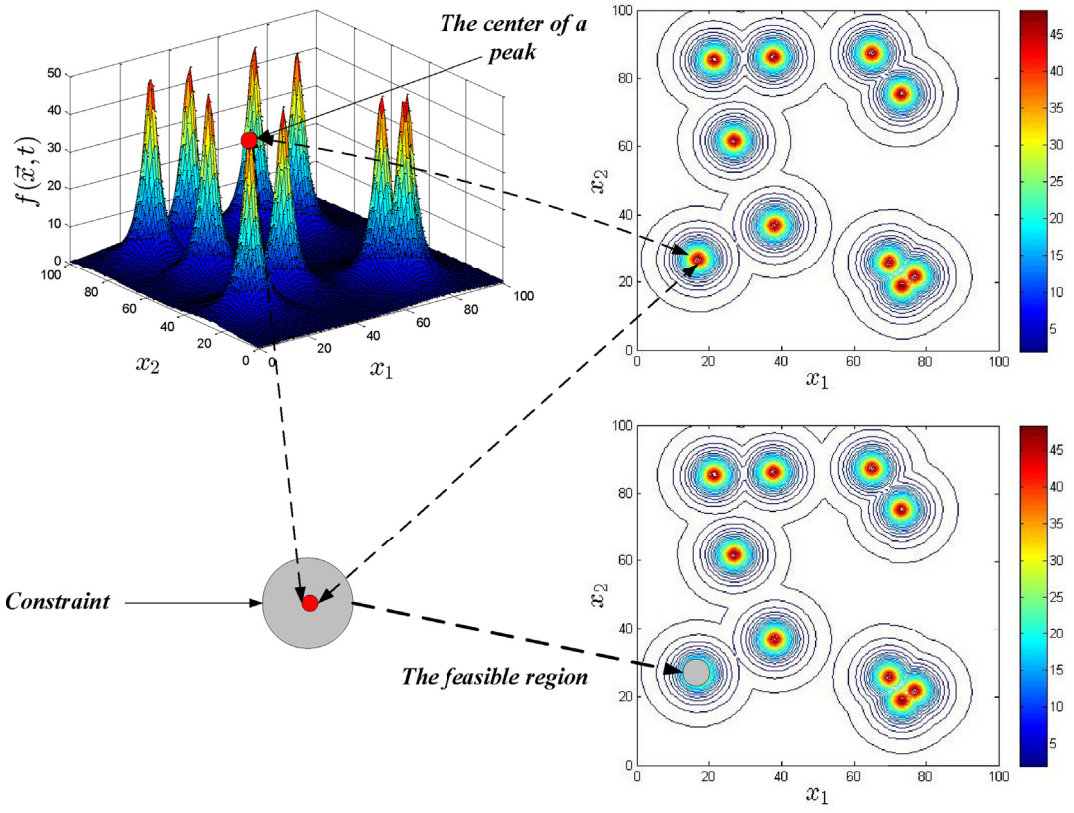


Fig. 2. Illustration of our test suite in a two-dimensional decision space.

environment. In 2017, inspired by the dynamic species-based particle swam optimization (DSPSO) [27], Bu et al. [12] designed a locating and tracking feasible regions DSPSO (LTFR-DSPSO), which uses the above four kinds of dynamic unconstrained optimization strategies to solve DCOPs.

2.2.4. Other methods

Apart from the above three categories, some other methods are designed to solve DCOPs. Liu [28] transformed an original DCOP into a biobjective dynamic optimization problem via a new dynamic entropy function. In 2011, Richter et al. [29] considered an asynchronous change pattern of DCOPs, in which the dynamic fitness landscape and dynamic constraints may change independently in terms of their respective time regimes. In addition, a speciation-based method with local search is proposed by Lu et al. [30].

3. Test suite construction and test instances

3.1. Test suite construction

First of all, we make use of MPB [8], which is a well-established dynamic unconstrained optimization benchmark generator and exhibits good time-varying properties, as the objective function. MPB is able to generate a given number of peaks with different heights. Therefore, MPB is a multi-modal benchmark. MPB can be expressed as follows:

$$f(\vec{x},t) = \max_{i=1,\ldots,p} \frac{H_i(t)}{1 + W_i(t)\sum_{j=1}^{D}(x_j - X_{i,j}(t))^2} \tag{2}$$

$$\vec{X}_i(t) = \vec{X}_i(t-1) + \vec{V}_i(t) \tag{3}$$

$$\vec{V}_i(t) = \frac{s}{|\vec{r} + \vec{V}_i(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{V}_i(t-1)) \tag{4}$$

where $f(\vec{x},t)$ is the objective function at environment t, $H_i(t)$ and $W_i(t)$ are the height and width of peak i at environment t, respectively, $X_{i,j}(t)$ is the jth dimension of the center of peak i at environment t, and p is the number of peaks. In (2), the position of the center of peak i (i.e., $\vec{X}_i(t) = (X_{i,1}(t), \ldots, X_{i,D}(t))$) is shifted in a random direction defined by a vector $\vec{V}_i(t)$. $\vec{V}_i(t)$ is controlled by there parameters, i.e., s, $\vec{r}$, and λ as shown in (4). s is the shift length, which defines the change severity of $f(\vec{x},t)$, $\vec{r}$ is a random vector, and λ is the correlated coefficient. Apart from the movement of the position, the movements of the height and width of peak i can be described as follows:

$$H_i(t) = H_i(t-1) + Height\_severity * \sigma \tag{5}$$

$$W_i(t) = W_i(t-1) + Width\_severity * \sigma \tag{6}$$

where $\sigma \in N(0,1)$, $N(\cdot,\cdot)$ is a normal distributed random number, *Height_severity* denotes the change severity of peak height, and *Width_severity* denotes the change severity of peak width. The default parameter settings of MPB are given in Table 1. As shown in Table 1, each peak is a cone. In addition, the change frequency U means that the environment will change every U fitness evaluations (FEs).

From the above introduction, it is clear that in MPB, the position, height, and width of each peak will change over time, and thus the optimal solution will also vary with time. Fig. 1 gives an example, in which there are 10 peaks, and their positions, heights, and widths change from environment $(t-1)$ to environment t.

Next, we add dynamic constraints to MPB and suggest a new test suite. The idea is very simple. Recognizing that in MPB the number of peaks can be adjusted flexibly and the position of each peak changes dynamically, we choose the center of one peak and add a constraint around this center, thus producing a feasible region. In this way, it is

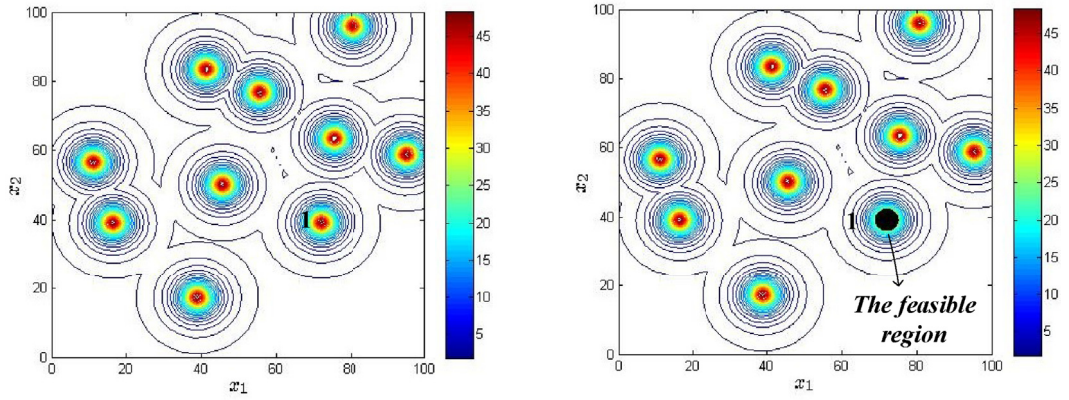
Table 2
Settings of l and a_k $(k = 1, \ldots, l)$ of the six test instances.

Test Instance	l	a_k $(k = 1, \ldots, l)$
1	1	$a_1 = 1$
2	1	a_1 is the index of the highest peak
3	2	$a_1 = 1$ and $a_2 = 6$
4	2	a_1 and a_2 are the indexes of the two highest peaks
5	3	$a_1 = 1$, $a_2 = 6$, and $a_3 = 10$
6	3	a_1, a_2, and a_3 are the indexes of the three highest peaks

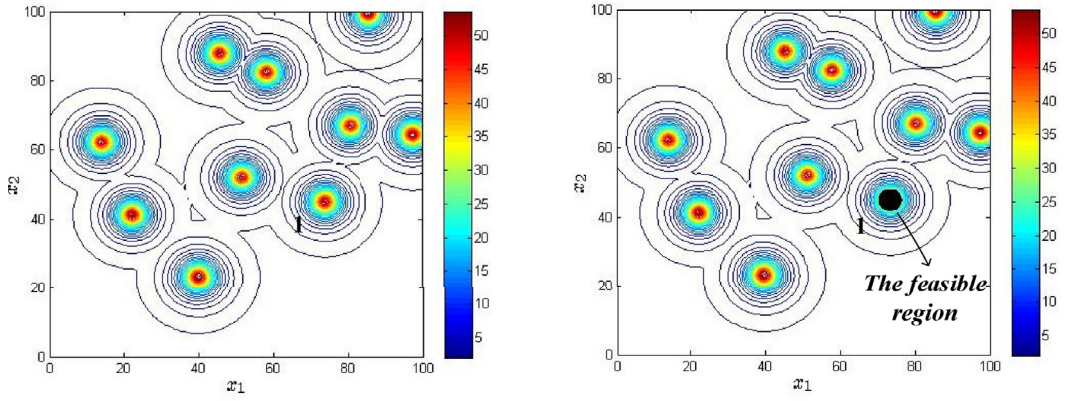
readily to control the number and positions of the feasible regions. Furthermore, each feasible region tracks the movement of its corresponding peak. As a result, by designing the tracking manner, we can adjust the change severity of the feasible regions. It is necessary to emphasize that in our test suite, the objective function is kept the same with the original MPB. Fig. 2 depicts how a constraint is added to MPB in a two-dimensional decision space. As shown in Fig. 2, first of all, we choose one peak from the 10 peaks and determine the center of this peak. Subsequently, we add a circle around this center and the area surrounded by the circle becomes a feasible region in the decision space.

$$\begin{cases} \text{maximize } f(\vec{x}, t) = \max\limits_{i=1,\ldots,p} \dfrac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{D} (x_j - X_{i,j}(t))^2} \\ \text{subject to: } g_1(\vec{x}, t) = \sum\limits_{j=1}^{D} (x_j - X_{a_1,j}(t))^2 - r_1^2(t) \le 0 \\ \quad \text{or } g_2(\vec{x}, t) = \sum\limits_{j=1}^{D} (x_j - X_{a_2,j}(t))^2 - r_2^2(t) \le 0 \\ \qquad \vdots \\ \quad \text{or } g_l(\vec{x}, t) = \sum\limits_{j=1}^{D} (x_j - X_{a_l,j}(t))^2 - r_l^2(t) \le 0 \end{cases} \tag{7}$$

The proposed test suite is given in (7), where $g_k(\vec{x}, t)$ denotes the kth constraint at environment t, $k \in \{1, \ldots, l\}$, l is the number of con-

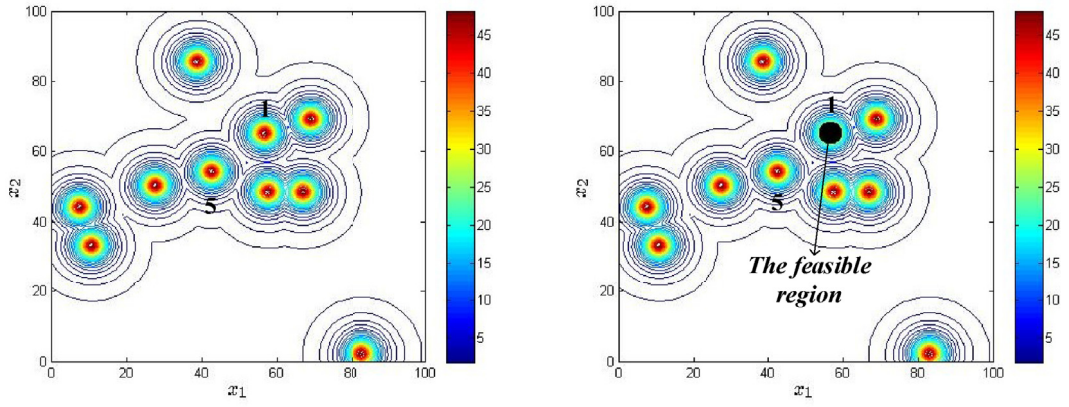


(a) Environment $(t-1)$

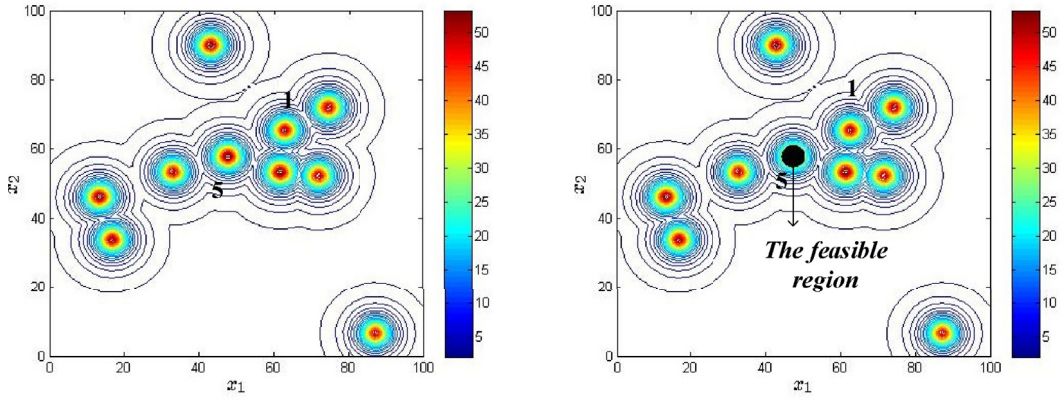


(b) Environment t

Fig. 3. Contours of test instance 1 at environment $(t-1)$ and environment t, the black area represents the feasible region, and the number represents the index of peak.



(a) Environment $(t-1)$



(b) Environment t

Fig. 4. Contours of test instance 2 at environment $(t-1)$ and environment t, the black area represents the feasible region, and the number represents the index of peak.

straints, $X_{a_k j}(t)$ is the jth dimension of the center of peak a_k at environment t, and $r_k(t)$ is the radius of the kth feasible region at environment t. The difficulty of solving a DCOP depends strongly on the number, locations, and sizes of the feasible regions. By tuning some parameters, the difficulty of this test suite can be adjusted freely. For example, the number of the feasible regions can be adjusted by l, and the location and size of each feasible region can be adjusted by $X_{a_k j}(t)$ and $r_k(t)$, respectively. It is noteworthy that, for simplicity, the shape of each feasible region in our test suite is a hype-sphere and we only consider inequality constraints in this paper.

For a decision vector $\vec{x}$, the degree of constraint violation on the kth constraint at environment t is computed as:

$$G_k(\vec{x},t) = \max(0, \sum_{j=1}^{D}(x_j - X_{a_k j}(t))^2 - r_k^2(t)) \tag{8}$$

Afterward, the degree of constraint violation of $\vec{x}$ on all the constraints at environment t is computed as:

$$G(\vec{x},t) = \min_{k=1,\ldots,l} G_k(\vec{x},t) \tag{9}$$

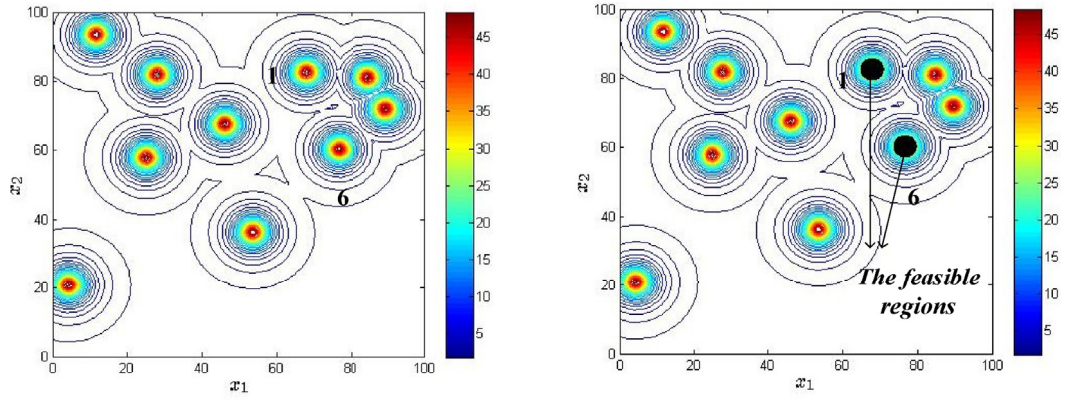
Through the above design, the proposed test suite not only maintains the original dynamics of MPB, but also considers the dynamics of constraints.

3.2. Test instances

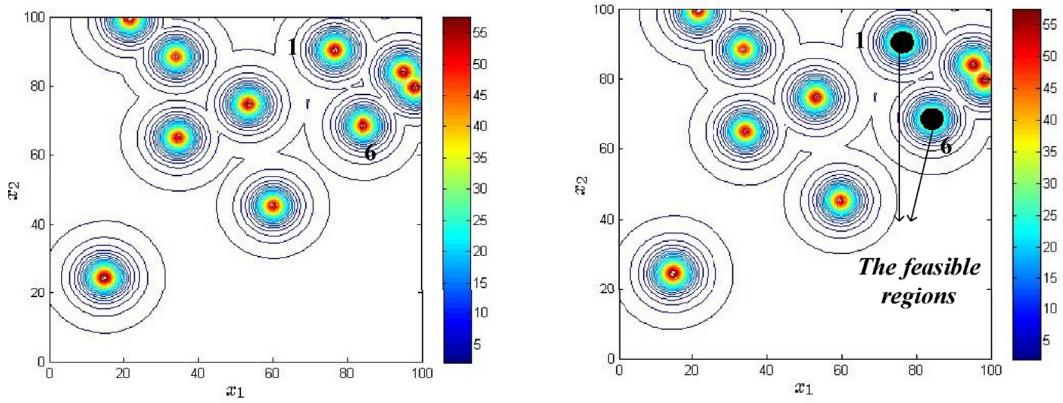
In this paper, we define two types of dynamics about constraints, namely, different numbers of the feasible regions and different change severities of the feasible regions.

- In the first type, the number of the feasible regions is controlled by the number of constraints. We take three different numbers of the feasible regions into account, e.g., single feasible region, two feasible regions, and three feasible regions, by adding one, two, and three constraints to MPB, respectively.
- The second type considers the change severity of the feasible regions. In order to simulate a slight change, the movements of the feasible regions always follow those of some fixed peaks, since a fixed peak of MPB may move slightly at successive environments. Additionally, our test suite also contains the feasible regions which track the highest peaks, aiming at simulating the circumstance that the feasible regions change drastically. It is because different environments may result in a drastic change of the highest peaks.

By coupling the above two types of dynamic constraints with MPB, we design six test instances. The major difference among them is the settings of l (i.e., the number of constraints/the feasible regions) and a_k



(a) Environment $(t-1)$



(b) Environment t

Fig. 5. Contours of test instance 3 at environment $(t-1)$ and environment t, the black areas represent the feasible regions, and the numbers represent the indexes of peaks.

$(k = 1, \ldots, l)$ (i.e., the index of a peak). Next, we introduce these six test instances one by one and their settings of l and a_k are summarized in Table 2.

3.2.1. Test instance 1

The first test instance has a single feasible region ($l = 1$), which changes slightly in the decision space due to the tracking of a fix peak. As displayed in Fig. 3, the feasible region tracks the first peak ($a_1 = 1$).

3.2.2. Test instance 2

The second test instance also has a single feasible region ($l = 1$), which changes drastically in the decision space because of the tracking of the highest peak. As shown in Fig. 4, at environment $(t-1)$, the index of the highest peak is 1 (i.e., $a_1 = 1$). While, at environment t, the index of the highest peak is 5 (i.e., $a_1 = 5$).

3.2.3. Test instance 3

The third test instance has two feasible regions in the decision space ($l = 2$), and they change slightly owing to the tracking of two fixed peaks. As shown in Fig. 5, the two feasible regions track the first peak ($a_1 = 1$) and the sixth peak ($a_2 = 6$), respectively.

3.2.4. Test instance 4

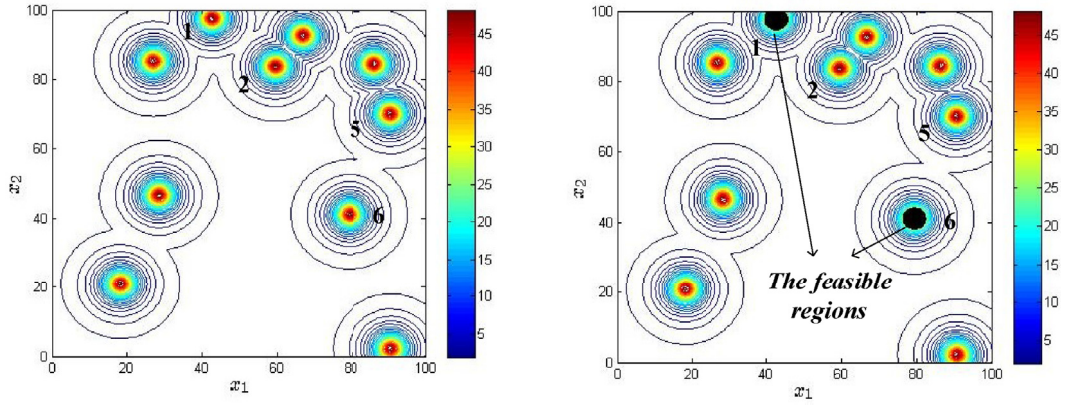
The fourth test instance also has two feasible regions in the decision space ($l = 2$), and they change drastically. In this case, we let the feasible regions track the two highest peaks. It can be seen from Fig. 6 that at environment $(t-1)$, the indexes of the two highest peaks are 1 and 6 (i.e., $a_1 = 1$ and $a_2 = 6$). However, at environment t, the indexes of the two highest peaks are 2 and 5 (i.e., $a_1 = 2$ and $a_2 = 5$).

3.2.5. Test instance 5

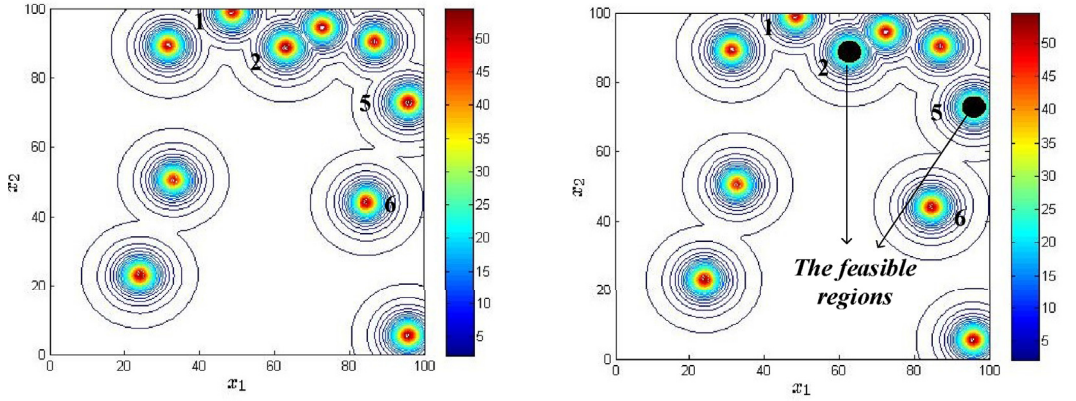
The fifth test instance has three feasible regions in the decision space ($l = 3$). They track three fixed peaks, and thus change slightly. As shown in Fig. 7, the three feasible regions track the first peak ($a_1 = 1$), the sixth peak ($a_2 = 6$), and the 10th peak ($a_3 = 10$), respectively.

3.2.6. Test instance 6

The sixth test instance also has three feasible regions in the decision space ($l = 3$), and they change dramatically in that they track the three highest peaks. As depicted in Fig. 8, at environment $(t-1)$, the indexes of the three highest peaks are 2, 4, and 10 (i.e., $a_1 = 2$, $a_2 = 4$, and $a_3 = 10$); at environment t, the indexes of the three highest peaks are 3, 7, and 9 (i.e., $a_1 = 3$, $a_2 = 7$, and $a_3 = 9$).



(a) Environment $(t-1)$



(b) Environment t

Fig. 6. Contours of test instance 4 at environment $(t-1)$ and environment t, the black areas represent the feasible regions, and the numbers represent the indexes of peaks.

In summary, we have designed six test instances based on our test suite. More test instances with various features can be constructed by changing the relevant parameter settings.

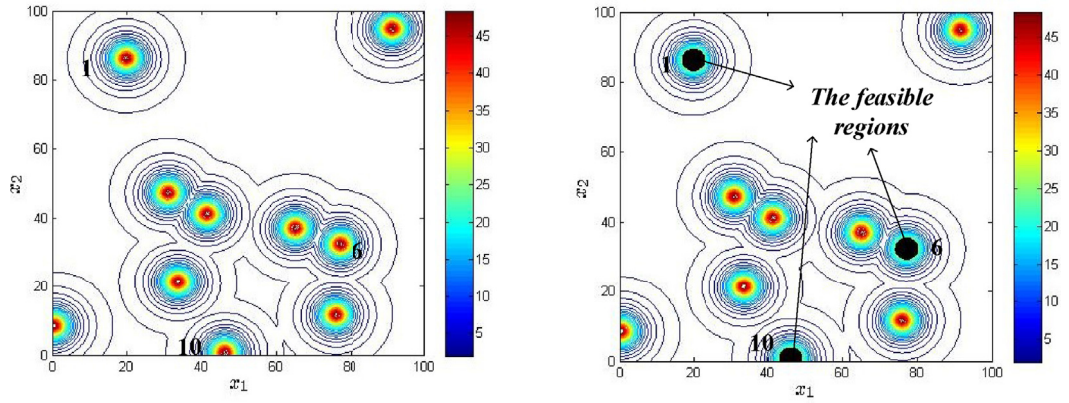
3.3. Characteristics of the proposed test suite

Compared with existing test suites, the proposed test suite has the following characteristics:

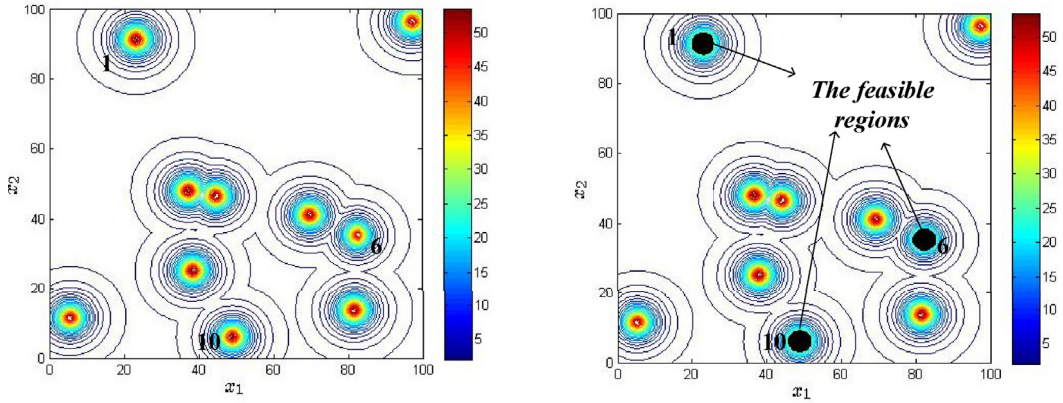
- *Scalability*: The number of peaks in MPB and the number of constraints are scalable. Moreover, the number of decision variables in objective function and constraints is also scalable.
- *Adjustability*: The size of each feasible region can be adjusted by the radius $r_k(t)$. The advantages of each feasible region tracking one of peaks are twofold: 1) the position of each feasible region is adjustable, and 2) the global optimum is also adjustable. Note that the global optimum may switch from one disconnected feasible region to another disconnected feasible region due to the random change of the peaks' heights. Moreover, the changing infeasible regions may lead to a new, better global optimum in a new environment.
- *Multi-modality*: It is obvious that our test suite inherits the multi-modality of MPB.
- *Change severity of the feasible regions*: In this paper, the change severity of the feasible regions is controlled by two different tracking manners. For example, the feasible regions track several fixed peaks or several highest peaks to achieve a slight change or a drastic change, respectively.
- *The global and local optimal solutions known*: Due to the fact that the local optimal solution of a feasible region is the center of the corresponding peak in the decision space and that all the centers of peaks in MPB are known *a priori*, all the local optimal solutions of our test suite are known. As a result, the global optimal solution is also known, which is the best local optimal solution.

Based on the above discussion, the proposed test suite satisfies the five important characteristics introduced in Section 2.1.

As pointed out by Branke [8], on the one hand test functions should be complex enough to simulate the real world, on the other hand they should be simple enough to gain insights into the working principle of an algorithm. Actually, in the proposed test suite, the scalability, adjustability, multi-modality, and change severity of the feasible regions can imitate the complex dynamics in the real world. In addition, the global and local optimal solutions known enable the test suite to evaluate the performance a DCOEA easily. Therefore, the proposed test suite can serve as a useful tool in evolutionary dynamic constrained optimization.



(a) Environment $(t-1)$



(b) Environment t

Fig. 7. Contours of test instance 5 at environment $(t-1)$ and environment t, the black areas represent the feasible regions, and the numbers represent the indexes of peaks.

Compared with the test suite in Ref. [12], our test suite has the following advantages:

- The test suite construction in Ref. [12] is complex. Thus, it is difficult to judge whether an algorithm performs well in a certain situation. In contrast, our test suite is easier to understand. It not only exhibits explicit characteristics as introduced previously, but also can analyze the performance of an algorithm in a certain situation. Thus, we can easily understand the advantages and disadvantages of an algorithm.
- The change of objective function and the change of constraints in Ref. [12] are independent with each other. Moreover, in some cases, both of them change randomly. When two independent/random changes are combined, the test suite seems uncontrollable, which cannot effectively test the performance of an algorithm. However, in our test suite, the dynamics of constraints has some relationship with that of objective function, which makes our test suite controllable.
- In real life, some problems change drastically while some problems change slightly. Our test suite takes the change severity into consideration, yet the test suite in Ref. [12] does not.
- In Ref. [12], the global and local optima are obtained by calculating the distances between all peaks and all the centers of the feasible regions. With the increase of the number of peaks and the feasible regions, the complexity of obtaining such information will increase drastically. In contrast, the global and local optima in our test suit can be obtained directly.

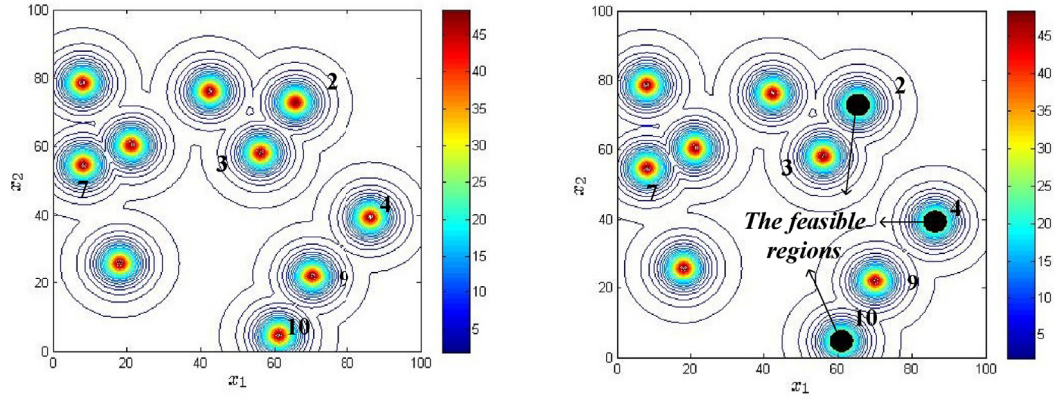
4. Compared algorithms

In this section, the performance of three DCOEAs is assessed based on the proposed six test instances. These three DCOEAs are a clustering particle swarm optimizer (CPSO) [31], LTFR-DSPSO [12] which has been briefly introduced in Section 2.2.3, and dynamic constrained optimization differential evolution (DyCODE) proposed in this paper.

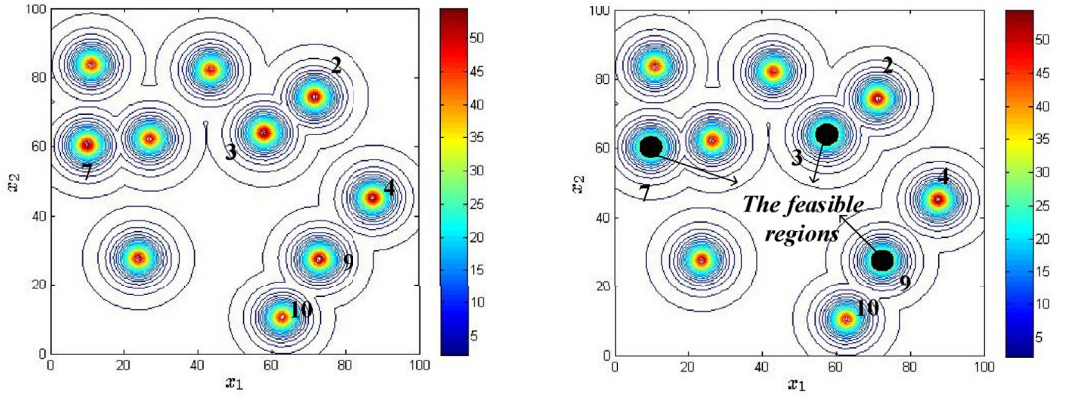
4.1. CPSO

CPSO is a state-of-the-art dynamic unconstrained optimization EA. Since CPSO achieves excellent performance on MPB, we choose it as one of the compared algorithms. The pseudocode of CPSO is shown in Algorithm 1.

Firstly, an initial cradle swarm C is generated from the decision space randomly. Then, a hierarchical clustering method is adopted to create several subswarms from C. Subsequently, a local search is implemented on each subswarm. If a particle in a subswarm is



(a) Environment $(t-1)$



(b) Environment t

Fig. 8. Contours of test instance 6 at environment $(t-1)$ and environment t, the black areas represent the feasible regions, and the numbers represent the indexes of peaks.

improved, which means a particle is better than its personal historical best position, then the information of this particle is used to update the historical best position (denoted as *gbest*) of the whole subswarm. Finally, the status (i.e., overlapping, convergence, and overcrowding) of each subswarm is checked and an operator is executed according to the corresponding status of each subswarm. If an environmental change is detected, a new cradle swarm will be regenerated by making use of the useful information of the last environment.

Due to its simplicity, generality, and ease of implementation, the feasibility-based rule proposed by Deb [32] is incorporated into CPSO, with the aim of dealing with constraints and extending CPSO to solve DCOPs. The feasibility-based rule compares pair-wise individuals as follows:

- When two feasible solutions are compared, the one with a better objective function value is chosen;
- When a feasible solution and an infeasible solution are compared, the feasible one is chosen;
- When two infeasible solutions are compared, the one with the less degree of constraint violation is chosen.

4.2. LTFR-DSPSO

LTFR-DSPSO is a very recent DCOEA with outstanding performance. The framework of LTFR-DSPSO is shown in Algorithm 2.

Firstly, LTFR-DSPSO starts from an initial population *pop*. Meanwhile, LTFR-DSPSO initializes three memory sets *LM*(*t*), *memory*, and *bestMem* as the empty sets. Then, *pop* is divided into several species via the clustering method in [27] and the best particle in each species is selected as the species seed. After multiple species have been created, *LM*(*t*) will be updated. Subsequently, the gradient-based repair method is used to repair infeasible species seeds and the sequential quadratic programming (SQP) is employed to conduct local search on feasible species seeds. If an environmental change is detected, LTFR-DSPSO enters the re-initialization phase; otherwise, each species seed is assigned as the neighborhood best of all particles in the same species. If some particles in a species are the same with its species seed, they are replaced with the same number of randomly generated particles. In LTFR-DSPSO, all the particles in each species are updated according to [33]. Afterward, LTFR-DSPSO again detects whether the environment changes or not. The main characteristic of LTFR-DSPSO is the re-initialization phase. In the re-initialization phase,

Algorithm 1 CPSO.
1: $clst = \emptyset$;
2: Generate an initial cradle swarm C;
3: Create several subswarms from C by a hierarchical clustering method: $slst[1]$, $slst[2]$, …;
4: **While** the stopping criterion is not satisfied **do**
5: **for** each subswarm $slst[i]$ **do**
6: Implement local search on $slst[i]$. In the local search, the historical best position (denoted as *gbest*) of $slst[i]$ isupdated by learning from an improved particle;
7: **end for**
8: Check the status of each subpopulation, i.e., overlapping, convergence, and overcrowding;
9: If two subswarms overlap with each other, they are combined into a new subswarm;
10: If the number of particles of a subswarm is greater than *max_subsize*, the redundant particles with the worst performance are removed from this subswarm one by one;
11: If a subswarm is converged, *gbest* of this subswarm is stored into *clst*. Afterward, this subswarm is removed from *slst*;
12: If all subswarms are converged, *max_subsize* particles are randomly generated and put into C;
13: **if** an environmental change is detected **then**
14: Re-initialize C;
15: The particles in *clst* are used to replace the same number of the worst particles in C;
16: Split C into several subswarms by a hierarchical clustering method: $slst[1]$, $slst[2]$, …;
17: $clst = \emptyset$;
18: **end if**
19: **end while**

an ensemble of locating and tracking feasible regions is proposed to handle different types of dynamics in constraints, including tracking current feasible regions (Step 12), tracking previous feasible regions (Step 13), and predicting future feasible regions (Step 14). Finally, LTFR-DSPSO updates *pop* for the next environment (Steps 17–25) and divides the updated *pop* into multiple species by the clustering method in [27]. The procedure iterates until the stopping criterion is satisfied.

4.3. DyCODE

Up to now, although researchers have realized the important significance of DCOPs in the real world, the algorithms specially designed for DCOPs remain scarce. Therefore, the third compared algorithm in this paper is our proposed DyCODE. In DyCODE, DE serves as the search engine.

4.3.1. DE

DE, proposed by Storn and Price in 1995 [34], is a population-based stochastic search algorithm and has been broadly applied to solve a variety of optimization problems in diverse fields [35–40]. Das et al. [41] and Del Ser et al. [42] summarized the recent research progress of DE. DE uses mutation, crossover, and selection operators at each generation to evolve the population towards the global optimum. Herein, we let $\vec{x}_i$ denote the ith individual (also called the ith target vector) of the population. Firstly, a mutant vector $\vec{v}_i$ is generated for $\vec{x}_i$ by the mutation operator:

$$\vec{v}_i = \vec{x}_{r_1} + F * (\vec{x}_{r_2} - \vec{x}_{r_3}) \tag{10}$$

where r_1, r_2 and r_3 are three mutually different integers randomly selected from $\{1, \ldots, NP\}$, NP is the population size, $(\vec{x}_{r_2} - \vec{x}_{r_3})$ is the differential vector, and F is the scaling factor to amplify the differential vector. After mutation, the crossover operator is implemented on $\vec{x}_i$ and $\vec{v}_i$ to produce a trial vector $\vec{u}_i$:

$$u_{i,j} = \begin{cases} v_{i,j}, \text{if } rand_j \le CR \text{ or } j = j_{rand} \\ x_{i,j}, \text{otherwise} \end{cases} \tag{11}$$

where $i = 1, \ldots, NP$, $j = 1, \ldots, D$, j_{rand} is a randomly chosen integer from $\{1, \ldots, D\}$ to ensure that $\vec{u}_i$ always differs from $\vec{x}_i$, $rand_j$ is a uniformly distributed random number between 0 and 1, and $CR \in [0, 1]$ is the crossover control parameter. Finally, the selection operator is performed to select the better one from $\vec{x}_i$ and $\vec{u}_i$ for the next generation:

$$\vec{x}_i = \begin{cases} \vec{u}_i, \text{if } f(\vec{u}_i) \ge f(\vec{x}_i) \\ \vec{x}_i, \text{otherwise} \end{cases} \tag{12}$$

where $f(\cdot)$ is the objective function.

4.3.2. Motivation

The target of solving a dynamic optimization problem is to track the movement of the optimal solution. When encountering a DCOP, it is necessary to deal with the dynamics of both objective function and constraints. When the environment changes, the new global optimum either moves along with a changing feasible region or appears in a new feasible region. Therefore, a DCOEA should have the ability to not only locate multiple feasible regions, but also probe the found feasible regions until the population converges. Moreover, a DCOEA should be capable of maintaining good diversity for the next environment. Motivated by these considerations, we propose a three-phase DCOEA, called DyCODE. In the first phase, DyCODE utilizes the multi-population search strategy to locate as many feasible regions as possible. In the second phase, all subpopulations are aggregated together and focus on the search of the optimal solution of the current environment. Once an environmental change is detected, DyCODE begins the third phase. The third phase makes use of the useful information of the first two phases to re-initialize a new population.

Algorithm 2 LTFR-DSPSO.

1: Initialize the population *pop*;
2: Initialize three memory sets $LM(t)$, *memory*, and *bestMem* to be the empty sets;
3: **while** the stopping criterion is not satisfied **do**
4: Evaluate unevaluated individuals in *pop*;
5: Divide *pop* into several species by the clustering method in [27]. The best particle of each species is selected as the species seed;
6: Update $LM(t)$;
7: Locate new feasible regions by repairing infeasible species seeds and conduct local search on feasible species seeds;
8: **if** an environmental change is detected **then**
9: Update *memory* and *bestMem*;
10: Re-evaluate the historical best position of each particle in each species;
11: $pool = \emptyset$;
12: Repair infeasible species seeds, assign each species seed as the neighborhood best of all particles in the same species, and update the particles in each species;*//tracking current feasible regions*
13: Retrieve some particles from $LM(t)$, *memory*, and *bestMem*, denoted as X_1;*//tracking previous feasible regions*
14: If $t > 1$, predict the future positions of some particles in $LM(t)$, denoted as X_2;*//predicting future feasible regions*
15: $pool = X_1 \cup X_2$ and $LM(t) = \emptyset$;
16: Put feasible species seeds into *pool* and execute local search on feasible particles in *pool*;
17: Remove the inactive particles from *pop*;
18: $ET = pool \cup pop$ and $pop = \emptyset$;
19: **if** $ET.size \leq NP$ **then**
20: $pop = ET$, and add some random particles into *pop* until *pop* is full;
21: **else**
22: Split *ET* into several species based on the clustering method in [27];
23: Choose some excellent particles of each species and merge them into *pop*;
24: if $pop.size \leq NP$, add some random particles into *pop* until *pop* is full;
25: **end if**
26: Divide *pop* into several species by the clustering method in [27]. The best particle of each species is selected as the species seed;
27: **end if**
28: Assign each species seed as the neighborhood best of all particles in the same species;
29: If some particles in a species are the same with their species seed, replace them with the same number of randomly generated particles;
30: Update the particles in each species;
31: **if** an environmental change is detected **then**
32: Executes Step 9–Step 25;
33: **end if**
34: **end while**

4.3.3. Algorithmic framework

Algorithm 3 shows the procedure of DyCODE, in which $detection = 0$ and $detection = 1$ denote that an environmental change is undetected and detected, respectively, *FeasiRate* denotes the rate of feasible individuals among the *NP* individuals, *SelectPro* is a coefficient, $|subpop\{i\}|$ and $|pop|$ denotes the number of individuals in $subpop\{i\}$ and *pop*, respectively, and *NS* is the subpopulation size. Next, we introduce the three phases of DyCODE.

- In the first phase (Steps 5–13 in Algorithm 3), DyCODE adopts a multi-population search strategy to locate multiple feasible regions. Firstly, DyCODE generates an initial population *pop* in the decision space, and then *pop* is divided into several subpopulations via the clustering method introduced in Ref. [43], which has been given in Algorithm 4. This clustering method is simple, and one only needs to specify the size of the subpopulation. After the clustering, DE operators are utilized to motivate subpopulations toward the feasible regions from different directions. It is worth noting that instead of the traditional one-to-one comparison of DE in (12), for each subpopulation, the offspring and parents are mixed together, and the better half of them will survive to the next generation. The advantage of this comparison is to enable each subpopulation to enter the feasible region promptly. As a result, the number of feasible solutions will increase quickly. When the rate of feasible solutions in *pop* exceeds a target value, i.e., *targetFeasiRate*, *pop* will enter the second phase.
- In the second phase (Steps 14–22 in Algorithm 3), the best $(SelectPro^* |subpop\{i\}|)$ individuals of each subpopulation are combined together, and then form an updated *pop*. It is clear that the size of the updated *pop* is less than that of the original *pop*. The main reason is the following. For dynamic constrained optimization, the computing resource of each environment may be very limited. Therefore, to save the computing resource, we only use some of the excellent individuals in each subpopulation to search the optimal solution. Under this condition, DE operators are used to probe the found feasible regions. When an environmental change is detected, the second phase will terminate.
- The third phase (Steps 23–26 in Algorithm 3) is the re-initialization phase for the next environment. The re-initialized population consists of three parts, i.e., the best individual of each subpopulation in the first phase, the best individual in the second phase, and some randomly generated individuals. The reasons for selecting the first two parts of individuals are twofold: 1) for some DCOPs, the feasible solutions of the previous environment may still be feasible at the next environment; and 2) the feasible regions of the previous environment may not be far away from the feasible regions of the next

Algorithm 3 DyCODE.

Initialize the population *pop* with *NP* individuals: $\vec{x}_1, \ldots, \vec{x}_{NP}$;
detection = 0;
MemoSet = ∅;
while the stopping criterion is not satisfied **do**
  while *detection* == 0and*FeasiRate* < *targetFeasiRate* **do**
    Clustering (*pop*,*subpop*);
    for each *subpop*{*i*} **do**
      Implement the mutation and crossover operators of DE in (10) and (11) on *subpop*{*i*} to generate an offspring subpopulation *offsubpop*{*i*};
      Select the best half of individuals in *subpop*{*i*} ∪ *offsubpop*{*i*} with the least degree of constraint violation, *subpop*{*i*} = ∅, and put them into *subpop*{*i*};
    end for
    Update *detection* and *FeasiRate*;
  end while
  Save the best individual in each subpopulation into *MemoSet*;
  if *detection* == 0and*FeasiRate* ≥ *targetFeasiRate* **then**
    pop = ∅;
    For each subpopulation *subpop*{*i*}), store the best (*SelectPro**|*subpop*{*i*}|) individuals into *pop*;
    while *detection* == 0 **do**
      Implement mutation, crossover, and selection operators of DE in (10), (11), and (12) on *pop*;
      Update *detection*;
    end while
  end if
  Save the best individual of *pop* into *MemoSet*;
  pop = *MemoSet*;
  if |*pop*| < *NP* **then**
    Randomly generate (*NP* − |*pop*|) individuals from the search space and put them into *pop*;
  end if
  detection = 0;
end while

environment. In addition, with respect to the third part of individuals, they can maintain the diversity.

Algorithm 4 Clustering(*pop*, *subpop*).

Randomly generate a reference point $\vec{r}$ from the decision space;
for $i = 1, \ldots, \lfloor NP/NS \rfloor$ **do**
  Determine the individual in *pop* with the least Euclidean distance to $\vec{r}$, denoted as $\vec{s}$;
  Select *NS* individuals with the least Euclidean distances to $\vec{s}$, store them into *subpop*{*i*}, and remove them from *pop*;
end for
if |*pop*| > 0 **then**
  subpop{$\lfloor NP/NS \rfloor + 1$} = *pop*;
end if

Generally speaking, to solve DCOPs effectively, it is important to detect the environmental change. In this paper, we use a simple method to achieve this goal for the three compared DCOEAs. A randomly generated individual is regarded as the detector in the decision space. The detector will be re-evaluated at each generation. If its objective function value or the degree of constraint violation changes, we consider that an environmental change is detected.

5. Experimental study

5.1. Performance metrics

Two performance metrics are utilized in this paper. The first performance metric is the offline error [31]:

$$e = \frac{1}{K}\sum_{k=1}^{K}(g_k - b_k) \tag{13}$$

where b_k is the best solution provided by an algorithm at the end of the kth environment, g_k is the global optimal solution of the kth environment, and K is the total number of environments. Thus, the offline error e is the average difference between g_k and b_k in all the K environments.

The second performance metric is the normalized score [44], denoted as S_{norm}. The normalized score of the ith algorithm is calculated as follows:

$$S_{norm}(i) = \frac{1}{m}\sum_{j=1}^{m}\frac{|e_{\max}(j) - e(i,j)|}{|e_{\max}(j) - e_{\min}(j)|},\ i = 1, \ldots, n \tag{14}$$

where m is the number of test instances, n is the number of algorithms, $e_{\max}(j)$ and $e_{\min}(j)$ are the largest and smallest offline error values among all algorithms in solving the jth test instance, respectively, and $e(i,j)$ is the offline error of algorithm i on the jth test instance. From (14), the normalized score of each algorithm is between 0 and 1. The higher the normalized score, the better the performance of an algorithm. Moreover, it is easy to derive that the normalized scores of the best algorithm and the worst algorithm among n compared algorithms on one test instance are equal to 1 and 0, respectively.

5.2. Parameter settings

The parameter settings of LTFR-DSPSO and CPSO mostly inherited from their original papers. It is worth noting that the clustering method adopted by LTFR-DSPSO is relatively dependent on the problem. For example, it is very difficult to decide a proper clustering radius for different types of DCOPs. Therefore, we replaced the clustering method of LTFR-DSPSO with that of DyCODE. The main population size and the subpopulation size of LTFR-DSPSO were the same with DyCODE.

Table 3
The mean and standard deviation of the offline error for test instance 1 with 10D, 20D and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 1	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	3.39E-02±1.10E+00	9.67E-05±1.30E-04	4.45E-02±1.09E-01
10D (s=2)	5.50E-02±2.98E-01	6.39E-05±5.92E-05	5.71E-02±1.30E-01
10D (s=3)	4.24E-01±1.13E+00	8.98E-05±7.59E-05	2.13E-01±5.62E-01
10D (s=4)	1.12E-01±3.87E-01	1.15E-04±2.18E-04	4.09E-01±8.69E-01
10D (s=5)	2.23E-01±7.26E-01	1.64E-04±1.76E-04	5.64E-01±9.98E-01
10D (s=6)	9.86E-01±2.08E+00	1.20E-04±1.01E-04	3.84E-01±9.14E-01
20D (s=1)	1.60E+00±4.06E+00	4.86E-01±2.76E-01	2.07E-01±2.19E-01
20D (s=2)	2.59E+00±6.13E+00	4.14E-01±2.58E-01	2.97E-01±2.76E-01
20D (s=3)	5.38E+00±7.83E+00	3.83E-01±2.69E-01	6.42E-01±9.09E-01
20D (s=4)	3.92E+00±7.14E+00	5.64E-01±3.75E-01	8.33E-01±8.96E-01
20D (s=5)	2.22E+00±3.10E+00	5.66E-01±3.52E-01	1.35E+00±1.75E+00
20D (s=6)	3.61E+00±4.76E+00	6.00E-01±3.70E-01	1.36E+00±1.83E+00
30D (s=1)	2.85E+00±3.05E+00	2.63E+00±7.70E+00	2.30E+00±8.61E-01
30D (s=2)	3.50E+00±6.92E+00	2.52E+00±6.97E-01	2.59E+00±1.09E+00
30D (s=3)	6.12E+00±7.40E+00	2.18E+00±7.76E-01	3.23E+00±1.70E+00
30D (s=4)	6.95E+00±7.29E+00	2.67E+00±7.19E-01	3.41E+00±1.33E+00
30D (s=5)	6.40E+00±8.01E+00	4.17E+00±1.50E+00	4.47E+00±2.62E+00
30D (s=6)	5.12E+00±6.05E+00	4.80E+00±1.52E+00	5.23E+00±2.95E+00

Specifically, for LTFR-DSPSO and DyCODE, the main population size and the subpopulation size were set to 45 and 10, respectively. In terms of CPSO, the swarm size was set to 30, and the maximum number of particles in a subswarm was set to 5. For DyCODE, *targetFeasiRate* was set to 0.2, *SelectPro* was set to 0.3, and both F and CR in DE were set to 0.5.

CPSO, LTFR-DSPSO, and DyCODE were compared on the proposed six test instances. In these six test instances, we tested three different dimensions, i.e., 10D, 20D, and 30D. When the dimensions are equal to 10 and 20 (i.e., $D = 10$ and 20), the change frequency U was set to 5000. As the dimension increases, the difficulty of test instances will increase correspondingly. Therefore, U was set to 6000 when $D = 30$. In this paper, we implemented 10 dynamic environments for each test instance (i.e., $K = 10$). The radius $r_k(t)$ of each feasible region was set to 6.

For a fair comparison, the feasibility-based rule [32] was adopted as the constraint-handling technique for both CPSO and DyCODE.

Table 4
The mean and standard deviation of the offline error for test instance 2 with 10D, 20D and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 2	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	1.14E+00±2.05E+00	4.98E-03±5.07E-03	2.48E-01±6.22E-01
10D (s=2)	7.38E-01±1.93E+00	3.58E-03±5.02E-03	1.01E-01±3.40E-01
10D (s=3)	2.79E+00±4.39E+00	8.10E-03±1.59E-02	1.89E-01±8.48E-01
10D (s=4)	2.13E+00±3.70E+00	2.96E-03±3.09E-03	2.62E-01±7.03E-01
10D (s=5)	1.11E+00±3.19E+00	4.47E-03±5.12E-03	3.78E-01±1.59E+00
10D (s=6)	1.82E+00±2.46E+00	5.42E-03±5.44E-03	1.59E-01±4.48E-01
20D (s=1)	6.60E+00±7.77E+00	5.83E+00±2.88E+00	1.33E+00±9.93E-01
20D (s=2)	3.67E+00±3.71E+00	6.22E+00±3.33E+00	1.08E+00±8.01E-01
20D (s=3)	7.28E+00±6.73E+00	6.85E+00±3.55E+00	1.66E+00±1.66E+00
20D (s=4)	6.89E+00±6.96E+00	6.09E+00±3.44E+00	1.95E+00±2.14E+00
20D (s=5)	6.73E+00±6.56E+00	6.26E+00±2.43E+00	1.90E+00±1.86E+00
20D (s=6)	7.83E+00±7.92E+00	7.16E+00±3.87E+00	2.32E+00±2.66E+00
30D (s=1)	1.19E+01±6.91E+00	1.61E+01±5.61E+00	1.35E+01±3.72E+00
30D (s=2)	1.22E+01±9.32E+00	1.67E+01±5.83E+00	1.36E+01±5.03E+00
30D (s=3)	1.64E+01±8.73E+00	1.79E+01±5.21E+00	1.46E+01±3.90E+00
30D (s=4)	1.32E+01±5.95E+00	1.70E+01±6.10E+00	1.31E+01±4.13E+00
30D (s=5)	1.19E+01±8.66E+00	1.83E+01±7.09E+00	1.39E+01±4.55E+00
30D (s=6)	1.32E+01±1.00E+01	2.02E+01±6.64E+00	1.56E+01±5.05E+00

Table 5
The mean and standard deviation of the offline error for test instance 3 with 10D, 20D and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 3	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	1.20E+00±3.20E+00	3.61E+00±4.94E+00	6.05E+00±6.36E+00
10D (s=2)	1.73E-01±9.27E-01	2.84E+00±5.48E+00	8.95E+00±8.15E+00
10D (s=3)	7.76E-01±1.78E+00	2.22E+00±3.23E+00	7.12E+00±7.22E+00
10D (s=4)	7.10E-01±1.46E+00	2.05E+00±2.38E+00	6.71E+00±5.61E+00
10D (s=5)	8.22E-01±1.54E+00	3.46E+00±6.02E+00	9.27E+00±6.51E+00
10D (s=6)	2.09E+00±3.48E+00	5.17E+00±7.53E+00	1.12E+01±8.69E+00
20D (s=1)	1.68E+00±2.40E+00	5.18E+00±3.51E+00	8.86E+00±5.85E+00
20D (s=2)	1.95E+00±2.12E+00	6.74E+00±6.62E+00	9.97E+00±1.00E+01
20D (s=3)	2.60E+00±2.19E+00	6.16E+00±5.50E+00	1.08E+01±9.07E+00
20D (s=4)	1.48E+00±1.51E+00	6.27E+00±3.92E+00	1.31E+01±1.36E+01
20D (s=5)	1.47E+00±1.50E+00	6.92E+00±5.51E+00	1.01E+01±8.92E+00
20D (s=6)	2.73E+00±3.00E+00	6.86E+00±5.17E+00	1.29E+01±1.05E+01
30D (s=1)	8.92E+00±4.36E+00	1.18E+01±6.36E+00	9.82E+00±6.57E+00
30D (s=2)	9.04E+00±4.95E+00	1.17E+01±6.61E+00	1.28E+01±9.00E+00
30D (s=3)	1.24E+01±6.03E+00	1.01E+01±7.65E+00	1.28E+01±8.71E+00
30D (s=4)	9.04E+00±3.65E+00	1.11E+01±5.79E+00	1.52E+01±1.03E+01
30D (s=5)	1.09E+01±5.92E+00	1.70E+01±8.06E+00	1.22E+01±8.71E+00
30D (s=6)	9.78E+00±4.71E+00	1.36E+01±6.14E+00	1.44E+01±1.01E+01

5.3. Experimental analysis

We recorded the experimental results of CPSO, LTFR-DSPSO, and DyCODE according to the two performance metrics introduced in Section 5.1. In terms of the offline error, the experimental results are summarized in Tables 3–8, in which "Mean Offline Error" (abbreviated as Mean OE) and "Std Dev" indicate the mean and standard deviation of the offline error over 30 independent runs, respectively. In addition, as far as the normalized score is concerned, the experimental results are given in Tables 9–11. Note that the calculation of the normalized score is based on the offline error. As shown in Table 1, there are six different shift lengths of s in MPB. As a result, each test instance can generate six scenarios.

Next, we discuss the experimental results from three aspects: dimension, the number of the feasible regions, and the change severity of the feasible regions. The reason why these three aspects are important is the following. In the community of evolutionary computation, it is well-recognized that the performance of an EA is signif-

Table 6
The mean and standard deviation of the offline error for test instance 4 with 10D, 20D and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 4	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	2.66E-01±5.15E-01	7.93E-01±6.48E-01	2.74E+00±2.51E+00
10D (s=2)	1.55E-01±4.36E-01	9.58E-01±8.04E-01	2.95E+00±2.08E+00
10D (s=3)	9.35E-01±2.55E-01	7.14E-01±5.44E-01	3.80E+00±2.20E+00
10D (s=4)	8.96E-01±1.59E-01	9.89E-01±7.60E-01	4.00E+00±3.19E+00
10D (s=5)	4.52E-01±1.15E+00	1.10E+00±0.86E+00	3.83E+00±2.91E+00
10D (s=6)	7.23E-01±1.39E+00	1.40E+00±1.14E+00	5.72E+00±5.24E+00
20D (s=1)	2.98E+00±3.45E+00	7.29E+00±2.81E+00	7.72E+00±4.59E+00
20D (s=2)	2.37E+00±2.20E+00	7.37E+00±3.38E+00	7.14E+00±4.89E+00
20D (s=3)	3.46E+00±3.62E+00	8.22E+00±4.09E+00	8.94E+00±7.10E+00
20D (s=4)	2.68E+00±2.55E+00	8.21E+00±4.49E+00	8.17E+00±5.95E+00
20D (s=5)	2.43E+00±3.37E+00	9.70E+00±4.95E+00	7.09E+00±5.01E+00
20D (s=6)	3.76E+00±5.00E+00	9.59E+00±3.86E+00	1.03E+01±7.50E+00
30D (s=1)	1.53E+01±5.00E+00	1.64E+01±5.06E+00	1.51E+01±5.94E+00
30D (s=2)	1.42E+01±4.55E+00	1.66E+01±6.35E+00	1.32E+01±6.38E+00
30D (s=3)	1.56E+01±4.84E+00	1.65E+01±7.04E+00	1.53E+01±6.88E+00
30D (s=4)	1.45E+01±4.01E+00	1.66E+01±6.11E+00	1.42E+01±5.33E+00
30D (s=5)	1.49E+01±5.07E+00	2.16E+01±6.64E+00	1.76E+01±5.90E+00
30D (s=6)	1.57E+01±6.05E+00	2.09E+01±8.36E+00	1.63E+01±4.98E+00

Table 7
The mean and standard deviation of the offline error for test instance 5 with 10D, 20D and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 5	CPSO Mean OE±Std Dev	LTFR-DSPSO Mean OE±Std Dev	DyCODE Mean OE±Std Dev
10D (s=1)	4.11E-01±6.58E-01	5.55E+00±7.32E+00	1.07E+01±6.99E+00
10D (s=2)	7.12E-01±1.90E+00	7.62E+00±8.32E+00	1.29E+01±7.91E+00
10D (s=3)	4.18E-01±1.00E+00	2.91E+00±3.26E+00	1.13E+01±6.63E+00
10D (s=4)	5.23E-01±7.77E-01	7.03E+00±6.76E+00	1.61E+01±8.24E+00
10D (s=5)	5.61E-01±8.31E-01	4.96E+00±5.31E+00	1.37E+01±8.31E+00
10D (s=6)	1.54E+00±2.49E+00	8.00E+00±8.39E+00	1.36E+01±9.14E+00
20D (s=1)	4.97E+00±1.77E+00	9.82E+00±6.48E+00	1.27E+01±7.69E+00
20D (s=2)	5.32E+00±2.90E+00	7.08E+00±6.43E+00	1.31E+01±9.87E+00
20D (s=3)	6.19E+00±2.85E+00	8.04E+00±4.61E+00	1.57E+01±1.14E+01
20D (s=4)	6.32E+00±3.14E+00	1.00E+01±6.44E+00	1.20E+01±8.09E+00
20D (s=5)	5.90E+00±2.75E+00	1.11E+01±7.00E+00	1.91E+01±1.20E+01
20D (s=6)	7.77E+00±3.78E+00	1.21E+01±5.38E+00	1.60E+01±9.43E+00
30D (s=1)	2.00E+01±5.59E+00	1.36E+01±6.43E+00	1.16E+01±7.61E+00
30D (s=2)	2.28E+01±5.01E+00	1.51E+01±5.82E+00	1.45E+01±8.54E+00
30D (s=3)	2.65E+01±7.30E+00	1.35E+01±6.24E+00	1.64E+01±9.30E+00
30D (s=4)	2.53E+01±5.23E+00	1.59E+01±7.61E+00	1.47E+01±9.16E+00
30D (s=5)	2.79E+01±6.32E+00	1.91E+01±9.12E+00	1.69E+01±8.18E+00
30D (s=6)	2.52E+01±4.78E+00	1.93E+01±9.48E+00	1.43E+01±7.97E+00

Table 8
The mean and standard deviation of the offline error for test instance 6 with 10D, 20D and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 6	CPSO Mean OE±Std Dev	LTFR-DSPSO Mean OE±Std Dev	DyCODE Mean OE±Std Dev
10D (s=1)	3.62E-01±6.32E-01	1.87E+00±1.49E+00	4.78E+00±3.11E+00
10D (s=2)	2.89E-01±5.74E-01	2.55E+00±1.93E+00	5.89E+00±4.25E+00
10D (s=3)	4.18E-01±4.00E-01	2.26E+00±1.47E+00	6.24E+00±3.76E+00
10D (s=4)	3.98E-01±7.54E-01	2.21E+00±1.15E+00	7.35E+00±3.52E+00
10D (s=5)	5.35E-01±1.02E+00	2.51E+00±1.78E+00	8.22E+00±4.45E+00
10D (s=6)	5.35E-01±6.52E-01	2.42E+00±1.79E+00	8.38E+00±4.76E+00
20D (s=1)	6.86E+00±2.47E+00	1.03E+01±4.45E+00	8.89E+00±5.49E+00
20D (s=2)	6.10E+00±1.63E+00	1.00E+01±4.61E+00	9.44E+00±5.21E+00
20D (s=3)	7.82E+00±2.48E+00	1.05E+01±4.42E+00	8.94E+00±6.77E+00
20D (s=4)	6.81E+00±2.49E+00	8.91E+00±3.61E+00	1.08E+01±9.34E+00
20D (s=5)	8.07E+00±2.37E+00	1.00E+01±5.28E+00	1.06E+01±6.64E+01
20D (s=6)	7.37E+00±2.89E+00	1.04E+01±6.47E+00	1.13E+01±7.16E+00
30D (s=1)	2.72E+01±5.37E+00	1.86E+01±4.15E+00	1.53E+01±4.36E+00
30D (s=2)	2.81E+01±6.29E+00	1.61E+01±7.67E+00	1.46E+01±6.31E+00
30D (s=3)	3.08E+01±6.15E+00	1.80E+01±6.36E+00	1.66E+01±7.08E+00
30D (s=4)	3.02E+01±5.91E+00	1.58E+01±5.60E+00	1.44E+01±6.65E+00
30D (s=5)	3.06E+01±6.54E+00	1.89E+01±5.08E+00	1.66E+01±5.91E+00
30D (s=6)	3.21E+01±6.11E+00	2.25E+01±8.69E+00	1.82E+01±6.42E+00

icantly influenced by the dimension of an optimization problem. In addition, in this paper we introduce two types of dynamics about constraints in addition to objective function. We are interested in the effect of these two types of dynamics on the performance of a DCOEA.

Table 9
The normalized scores of the three compared algorithms on different dimensions.

Dimension	CPSO	LTFR-DSPSO	DyCODE
10D	0.7027	0.8010	0.1951
20D	0.6804	0.3904	0.3523
30D	0.4461	0.4836	0.7670

Table 10
The normalized scores of the three compared algorithms on different numbers of the feasible regions.

The number of the feasible regions	CPSO	LTFR-DSPSO	DyCODE
Single feasible region	0.2100	0.6599	0.7549
Two feasible regions	0.9526	0.4108	0.2011
Three feasible regions	0.6667	0.6044	0.3585

Table 11
The normalized scores of the three compared algorithms on different change severities of the feasible regions.

The change severity of the feasible regions	CPSO	LTFR-DSPSO	DyCODE
Slight change	0.5697	0.7070	0.3443
Drastic change	0.7468	0.3985	0.5255

5.3.1. Dimension

Generally speaking, the increase of dimension poses a great challenge on the performance of a DCOEA. It is because the search space will rapidly enlarge with the increase of dimension. It is clear from Tables 3–8 that the performance of all the three compared algorithms degrades as the dimension increases.

In terms of the offline error, when $D = 10$, the mean offline error values of LTFR-DSPSO are consistently less than those of CPSO and DyCODE on test instances 1 and 2. For test instances 3–6, CPSO achieves better performance than LTFR-DSPSO and DyCODE in all cases except for test instance 4 with $s = 3$. In the case of $D = 20$, CPSO outperforms LTFR-DSPSO and DyCODE on test instances 3–6, and LTFR-DSPSO and DyCODE show the best performance on test instance 1 and test instance 2, respectively. In addition, in the case of $D = 30$, LTFR-DSPSO, CPSO, and DyCODE perform the best on test instance 1, test instances 2–3, and test instances 4–6, respectively.

As shown in Table 9, the normalized score of CPSO decreases with the increase of dimension. This phenomenon can be attributed to the fact that the local search adopted by CPSO is related to the dimension. When a particle in a subswarm finds a better position, the best particle in the subswarm will learn from it on each dimension. Therefore, with the increase of dimension, this local search will consume more FEs. When $D = 10$, LTFR-DSPSO achieves the highest normalized score. However, with the increase of dimension, the normalized score of LTFR-DSPSO drops significantly. It is not difficult to understand, since the gradient-based repair and SQP in LTFR-DSPSO need more number of evaluations for constraints and objective function, respectively, with the increase of dimension. Furthermore, when SQP is used to conduct local search on feasible solutions, its performance would degenerate as the dimension increases. It is interesting to see that the normalized score of DyCODE increases with the increase of dimension. This can be explained as follows. In the second phase, some excellent individuals of each subpopulation are combined together to search the optimal solution, thus achieving fast convergence of DyCODE.

5.3.2. The number of the feasible regions

For a DCOEA, the increase of the number of the feasible regions will cause the following three difficulties: 1) a DCOEA should have the capability to approach the feasible regions from various directions, 2) a DCOEA runs the risk of getting stuck at a local feasible optimal solution if it ignores any of the feasible regions, and 3) the optimal solution will switch from one feasible region to another feasible region with the change of environment.

Table 10 reports the normalized scores of the three compared algorithms on different numbers of the feasible regions. Specifically, for DCOPs with a single feasible region, the normalized score of DyCODE is the highest. However, the performance of DyCODE is worse than

Table 12
The mean and standard deviation of the offline error for test instance 1 with 10 peaks and 15 peaks. We also tested three different dimensions: 10D, 20D, and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 1	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	3.39E-01±1.10E+00	9.67E-05±1.30E-04	4.45E-02±1.09E-01	1.10E+00±2.23E+00	7.71E-05±7.42E-05	3.21E-02±7.96E-02
10D (s=2)	5.49E-02±2.98E-01	6.39E-05±5.92E-05	5.71E-02±1.38E-01	8.32E-01±2.95E+00	6.70E-05±5.52E-05	2.00E-01±5.34E-01
10D (s=3)	4.24E-01±1.13E+00	8.98E-05±7.59E-05	2.13E-01±5.62E-01	5.76E-01±1.58E+00	7.66E-05±7.05E-05	2.16E-01±6.80E-01
10D (s=4)	1.12E-01±3.87E-01	1.15E-04±2.18E-04	4.09E-01±8.69E-01	9.16E-01±2.14E+00	1.17E-04±1.64E-04	5.38E-01±1.03E+00
10D (s=5)	2.23E-01±7.26E-01	1.64E-04±1.76E-04	5.64E-01±9.98E-01	4.67E-01±1.12E+00	3.68E-04±1.30E-03	3.98E-01±1.08E+00
10D (s=6)	9.86E-01±2.08E+00	1.21E-04±1.01E-04	3.84E-01±9.14E-01	5.77E-03±2.35E-02	9.57E-05±6.46E-05	4.14E-01±8.50E-01
20D (s=1)	1.60E+00±4.06E+00	4.86E-01±2.76E-01	2.07E-01±2.19E-01	2.36E+00±4.04E+00	4.07E-01±2.95E-01	4.90E-01±8.83E-01
20D (s=2)	2.59E+00±6.13E+00	4.14E-01±2.58E-01	2.97E-01±2.76E-01	2.08E+00±3.65E+00	4.99E-01±4.25E-01	7.82E-01±1.21E+00
20D (s=3)	5.38E+00±7.83E+00	3.83E-01±2.69E-01	6.42E-01±9.09E-01	2.69E+00±6.03E+00	5.50E-01±5.06E-01	7.07E-01±8.94E-01
20D (s=4)	3.92E+00±7.14E+00	5.64E-01±3.75E-01	8.33E-01±8.96E-01	5.31E+00±6.98E+00	5.22E-01±3.59E-01	1.38E+00±1.52E+00
20D (s=5)	2.22E+00±3.10E+00	5.66E-01±3.52E-01	1.35E+00±1.75E+00	5.59E+00±8.27E+00	5.55E-01±3.44E-01	1.41E+00±1.73E+00
20D (s=6)	3.61E+00±4.76E+00	6.00E-01±3.70E-01	1.36E+00±1.83E+00	4.14E+00±5.55E+00	7.18E-01±5.61E-01	1.45E+00±1.71E+00
30D (s=1)	2.85E+00±3.05E+00	2.63E+00±7.70E-01	2.30E+00±8.61E-01	4.05E+00±6.95E+00	2.72E+00±7.07E-01	2.14E+00±8.05E-01
30D (s=2)	3.50E+00±6.92E+00	2.52E+00±6.97E-01	2.59E+00±1.09E+00	5.05E+00±6.32E+00	2.96E+00±8.45E-01	2.95E+00±1.23E+00
30D (s=3)	6.12E+00±7.40E+00	2.18E+00±7.76E-01	3.23E+00±1.70E+00	4.50E+00±7.05E+00	2.31E+00±6.12E-01	2.19E+00±1.65E+00
30D (s=4)	6.95E+00±7.29E+00	2.67E+00±7.19E-01	3.41E+00±1.33E+00	4.84E+00±5.50E+00	2.90E+00±1.21E+00	3.92E+00±2.05E+00
30D (s=5)	6.40E+00±8.01E+00	4.17E+00±1.50E+00	4.47E+00±2.62E+00	9.85E+00±9.53E+00	4.37E+00±1.73E+00	4.26E+00±2.11E+00
30D (s=6)	5.12E+00±6.05E+00	4.80E+00±1.52E+00	5.23E+00±2.95E+00	6.26E+00±8.28E+00	4.72E+00±1.87E+00	4.42E+00±2.40E+00

Table 13
The mean and standard deviation of the offline error for test instance 2 with 10 peaks and 15 peaks. We also tested three different dimensions: 10D, 20D, and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 2	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	1.14E+00±2.05E+00	4.98E-03±5.07E-03	2.48E-01±6.22E-01	1.99E+00±2.60E+00	5.53E-03±6.31E-03	3.41E-01±1.11E+00
10D (s=2)	7.38E-01±1.93E+00	3.58E-03±5.02E-03	1.01E-01±3.40E-01	1.89E+00±3.14E+00	5.50E-03±5.70E-03	2.44E-01±8.73E-01
10D (s=3)	2.79E+00±4.39E+00	8.10E-03±1.59E-02	1.89E-01±8.48E-01	2.83E+00±5.23E+00	4.74E-03±5.28E-03	1.59E-01±4.55E-01
10D (s=4)	2.13E+00±3.70E+00	2.96E-03±3.09E-03	2.62E-01±7.03E-01	1.09E+00±1.95E+00	1.06E-02±9.89E-03	4.09E-01±1.18E+00
10D (s=5)	1.11E+00±3.19E+00	4.47E-03±5.12E-03	3.78E-01±1.59E+00	1.41E+00±2.74E+00	8.02E-03±1.10E-02	4.04E-01±1.41E+00
10D (s=6)	1.82E+00±2.46E+00	5.42E-03±5.44E-03	1.59E-01±4.48E-01	1.40E+00±2.71E+00	3.92E-03±4.30E-03	3.99E-01±1.09E+00
20D (s=1)	6.60E+00±7.77E+00	5.83E+00±2.88E+00	1.33E+00±9.93E-01	8.29E+00±7.47E+00	7.72E+00±3.23E+00	1.03E+00±9.02E-01
20D (s=2)	3.67E+00±3.71E+00	6.22E+00±3.33E+00	1.08E+00±8.01E-01	1.08E+01±8.72E+00	8.51E+00±3.78E+00	1.79E+00±2.34E+00
20D (s=3)	7.28E+00±6.73E+00	6.85E+00±3.55E+00	1.66E+00±1.66E+00	8.90E+00±7.25E+00	7.85E+00±3.80E+00	2.20E+00±1.63E+00
20D (s=4)	6.89E+00±6.96E+00	6.09E+00±3.44E+00	1.95E+00±2.14E+00	7.29E+00±7.76E+00	8.24E+00±4.56E+00	1.74E+00±1.14E+00
20D (s=5)	6.73E+00±6.56E+00	6.26E+00±2.43E+00	1.90E+00±1.86E+00	1.25E+01±1.03E+01	9.27E+00±4.45E+00	1.81E+00±1.36E+00
20D (s=6)	7.83E+00±7.92E+00	7.16E+00±3.87E+00	2.32E+00±2.66E+00	9.62E+00±7.14E+00	8.51E+00±3.39E+00	1.78E+00±1.43E+00
30D (s=1)	1.18E+01±6.91E+00	1.61E+01±5.61E+00	1.35E+01±3.72E+00	1.44E+01±9.64E+00	1.72E+01±5.71E+00	1.37E+01±4.08E+00
30D (s=2)	1.22E+01±9.32E+00	1.67E+01±5.83E+00	1.36E+01±5.03E+00	1.28E+01±9.30E+00	1.74E+01±5.94E+00	1.47E+01±5.30E+00
30D (s=3)	1.64E+01±8.73E+00	1.79E+01±5.21E+00	1.46E+01±3.90E+00	1.39E+01±8.43E+00	2.01E+01±6.05E+00	1.62E+01±5.53E+00
30D (s=4)	1.32E+01±5.95E+00	1.70E+01±6.10E+00	1.31E+01±4.13E+00	1.23E+01±8.70E+00	2.02E+01±5.33E+00	1.54E+01±3.88E+00
30D (s=5)	1.19E+01±8.66E+00	1.83E+01±7.09E+00	1.39E+01±4.55E+00	1.46E+01±9.90E+00	2.20E+01±7.43E+00	1.81E+01±5.19E+00
30D (s=6)	1.32E+01±1.00E+01	2.02E+01±6.64E+00	1.56E+01±5.05E+00	1.20E+01±8.68E+00	2.25E+01±6.74E+00	1.59E+01±4.29E+00

that of the two competitors for DCOPs with two and three feasible regions. The above phenomenon can be explained as follows. DyCODE combines some high-quality individuals of each subpopulation together in the second phase, which is beneficial to find the optimal solution in a single feasible region. Nevertheless, as the number of the feasible regions increases, it is very likely for DyCODE to converge into a local optimal solution in one of the feasible regions. In the case of two and three feasible regions, CPSO achieves the best performance. Based on our observation, compared with a single feasible region, multiple feasible regions enable multiple subswarms of CPSO to evolve independently for a longer period before the overlapping, converges, and overcrowding are identified. Thus, the probability that CPSO locates the feasible region containing the global optimal solution increases. It seems that LTFR-DSPSO is insensitive to the number of the feasible regions.

5.3.3. Change severity of the feasible regions

According to the offline error in Tables 3–8, in terms of the slight change, LTFR-DSPSO and CPSO perform the best on test instance 1 and test instance 3, respectively. In contrast, with respect to the drastic change, DyCODE and CPSO provide the best performance on test instance 2 and test instance 4, respectively. For test instance 5 and test instance 6, CPSO beats the two competitors when $D = 10$ and 20, and DyCODE surpasses the two competitors when $D = 30$.

Table 11 records the normalized scores of the three compared algorithms on different change severities of the feasible regions. When the feasible regions change slightly, the normalized score of LTFR-DSPSO is the highest. The reason may be that LTFR-DSPSO re-initializes the population by two strategies, i.e., tracking previous feasible regions and predicting future feasible regions. When the feasible regions change slightly, the feasible regions of the previous environment may not be distant from those of the next environment. Thus, these two strategies are effective. However, these two strategies may fail when a drastic change occurs. When the feasible regions change drastically, CPSO has the highest normalized score. It is because CPSO continuously detects whether a subswarm is overlapping, converges and overcrowding. Moreover, if all subswarms are converged, some individuals are randomly generated from the decision space, which enhances the diversity of the population. In addition, DyCODE is insensitive to the change severity of the feasible regions.

Table 14
The mean and standard deviation of the offline error for test instance 3 with 10 peaks and 15 peaks. We also tested three different dimensions: 10D, 20D, and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 3	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	1.20E+00±3.20E+00	3.61E+00±4.94E+00	6.05E+00±6.36E+00	5.11E-01±1.36E+00	4.01E+00±4.63E+00	7.41E+00±6.46E+00
10D (s=2)	1.73E-01±9.27E-01	2.84E+00±5.48E+00	8.95E+00±8.15E+00	1.24E+00±2.21E+00	3.95E+00±5.47E+00	7.31E+00±5.25E+00
10D (s=3)	7.76E-01±1.78E+00	2.22E+00±3.23E+00	7.12E+00±7.22E+00	1.97E-01±5.91E-01	2.50E+00±3.58E+00	1.03E+01±8.47E+00
10D (s=4)	7.10E-01±1.46E+00	2.05E+00±2.38E+00	6.71E+00±5.61E+00	7.58E-01±1.15E+00	3.73E+00±4.88E+00	8.06E+00±6.40E+00
10D (s=5)	8.22E-01±1.54E+00	3.46E+00±6.02E+00	9.27E+00±6.51E+00	5.63E-01±1.22E+00	4.36E+00±6.13E+00	8.46E+00±6.89E+00
10D (s=6)	2.09E+00±3.48E+00	5.17E+00±7.53E+00	1.12E+01±8.69E+00	6.01E-01±1.05E+00	2.62E+00±3.98E+00	9.71E+00±7.28E+00
20D (s=1)	1.68E+00±2.40E+00	5.18E+00±3.51E+00	8.86E+00±5.85E+00	2.14E+00±2.70E+00	6.69E+00±5.29E+00	1.29E+01±9.28E+00
20D (s=2)	1.95E+00±2.12E+00	6.74E+00±6.62E+00	9.97E+00±1.00E+01	1.96E+00±2.91E+00	3.53E+00±3.04E+00	8.77E+00±8.08E+00
20D (s=3)	2.60E+00±2.19E+00	6.16E+00±5.50E+00	1.08E+01±9.07E+00	2.29E+00±2.50E+00	5.51E+00±5.39E+00	1.24E+01±1.08E+01
20D (s=4)	1.48E+00±1.51E+00	6.27E+00±3.92E+00	1.31E+01±1.36E+01	2.85E+00±4.38E+00	6.65E+00±6.75E+00	8.99E+00±7.88E+00
20D (s=5)	1.47E+00±1.50E+00	6.92E+00±5.51E+00	1.06E+01±8.92E+00	3.25E+00±4.73E+00	7.07E+00±5.35E+00	9.02E+00±9.26E+00
20D (s=6)	2.73E+00±3.00E+00	6.86E+00±5.17E+00	1.29E+01±1.05E+01	1.99E+00±2.60E+00	5.48E+00±4.78E+00	1.39E+01±1.30E+01
30D (s=1)	8.92E+00±4.36E+00	1.18E+01±6.36E+00	9.82E+00±6.57E+00	7.44E+00±3.68E+00	9.48E+00±6.66E+00	7.03E+00±4.75E+00
30D (s=2)	9.04E+00±4.95E+00	1.17E+01±6.61E+00	1.28E+01±9.00E+00	1.06E+01±4.23E+00	8.73E+00±4.54E+00	9.90E+00±8.06E+00
30D (s=3)	1.24E+01±6.03E+00	1.01E+01±7.65E+00	1.28E+01±8.71E+00	9.75E+00±5.79E+00	1.02E+01±5.94E+00	1.20E+01±7.93E+00
30D (s=4)	9.04E+00±3.65E+00	1.11E+01±5.79E+00	1.52E+01±1.03E+01	1.05E+01±5.76E+00	1.11E+01±5.65E+00	1.28E+01±9.57E+00
30D (s=5)	1.09E+01±5.92E+00	1.47E+01±8.06E+00	1.22E+01±8.71E+00	1.17E+01±5.39E+00	1.61E+01±8.77E+00	1.22E+01±7.48E+00
30D (s=6)	9.78E+00±4.71E+00	1.36E+01±6.14E+00	1.44E+01±1.01E+01	9.81E+00±4.02E+00	1.64E+01±8.24E+00	1.37E+01±9.93E+00

Table 15
The mean and standard deviation of the offline error for test instance 4 with 10 peaks and 15 peaks. We also tested three different dimensions: 10D, 20D, and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 4	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	2.66E-01±5.15E-01	7.93E-01±6.48E-01	2.74E+00±2.51E+00	7.54E-01±1.48E+00	7.79E-01±7.50E-01	3.56E+00±3.09E+00
10D (s=2)	1.55E-01±4.36E-01	9.58E-01±8.04E-01	2.95E+00±2.08E+00	4.52E-01±1.25E+00	9.06E-01±7.72E-01	3.37E+00±2.42E+00
10D (s=3)	9.35E-01±2.55E+00	7.14E-01±5.44E-01	3.80E+00±2.21E+00	4.32E-01±1.09E+00	8.56E-01±5.91E-01	4.19E+00±3.91E+00
10D (s=4)	8.96E-01±1.59E+00	9.89E-01±7.60E-01	4.00E+00±3.19E+00	6.87E-01±1.71E+00	9.32E-01±6.21E-01	4.35E+00±2.86E+00
10D (s=5)	4.52E-01±1.15E+00	1.10E+00±8.61E-01	3.84E+00±2.91E+00	4.10E-01±9.28E-01	7.77E-01±7.09E-01	3.04E+00±2.54E+00
10D (s=6)	7.23E-01±1.39E+00	1.40E+00±1.14E+00	5.72E+00±5.24E+00	6.98E-01±1.30E+00	8.43E-01±8.08E-01	4.96E+00±3.94E+00
20D (s=1)	2.98E+00±3.45E+00	7.29E+00±2.81E+00	7.72E+00±4.59E+00	2.93E+00±3.03E+00	8.53E+00±3.49E+00	6.43E+00±4.32E+00
20D (s=2)	2.37E+00±2.20E+00	7.37E+00±3.38E+00	7.14E+00±4.89E+00	2.35E+00±2.28E+00	9.86E+00±4.22E+00	6.77E+00±4.43E+00
20D (s=3)	3.46E+00±3.62E+00	8.22E+00±4.09E+00	8.94E+00±7.10E+00	2.74E+00±2.87E+00	8.78E+00±4.36E+00	6.26E+00±3.91E+00
20D (s=4)	2.68E+00±2.55E+00	8.21E+00±4.49E+00	8.17E+00±5.95E+00	2.17E+00±2.24E+00	9.53E+00±4.99E+00	8.25E+00±6.21E+00
20D (s=5)	2.43E+00±3.37E+00	9.70E+00±4.95E+00	7.09E+00±5.01E+00	3.25E+00±3.79E+00	9.62E+00±3.86E+00	6.78E+00±5.21E+00
20D (s=6)	3.76E+00±5.00E+00	9.59E+00±3.86E+00	1.03E+01±7.50E+00	2.94E+00±2.11E+00	9.92E+00±4.83E+00	8.47E+00±4.94E+00
30D (s=1)	1.53E+01±5.00E+00	1.64E+01±5.06E+00	1.51E+01±5.94E+00	1.46E+01±5.42E+00	1.98E+01±5.73E+00	1.44E+01±6.89E+00
30D (s=2)	1.42E+01±4.55E+00	1.66E+01±6.35E+00	1.32E+01±6.38E+00	1.52E+01±4.79E+00	1.78E+01±6.25E+00	1.45E+01±6.19E+00
30D (s=3)	1.56E+01±4.84E+00	1.65E+01±7.04E+00	1.53E+01±6.88E+00	1.57E+01±5.38E+00	2.10E+01±6.92E+00	1.91E+01±7.06E+00
30D (s=4)	1.45E+01±4.01E+00	1.66E+01±6.11E+00	1.42E+01±5.33E+00	1.69E+01±5.60E+00	2.00E+01±6.29E+00	1.82E+01±5.82E+00
30D (s=5)	1.49E+01±5.07E+00	2.16E+01±6.64E+00	1.76E+01±5.90E+00	1.68E+01±5.06E+00	2.18E+01±7.39E+00	1.54E+01±6.93E+00
30D (s=6)	1.57E+01±6.05E+00	2.09E+01±8.36E+00	1.63E+01±4.98E+00	1.58E+01±5.24E+00	2.39E+01±8.61E+00	1.57E+01±5.64E+00

In general, when the feasible regions change dramatically, the test instances are more difficult to be solved. However, it is interesting to note that compared with the slight change, the three compared algorithms perform better under the drastic change for test instances 3–6 with $D = 10$. Based on our observation, even the feasible regions dramatically change, the index of the highest peak may not change in the successive environments. Thus, an algorithm only needs to track the feasible region associated with the same highest peak in different environments. However, in a slight change situation with multiple feasible regions, the optimal solution may jump among different feasible regions; thus, an algorithm needs to track multiple feasible regions. Therefore, sometimes a DCOP with a drastic change may be simpler than a DCOP with a slight change. As a result, the three compared algorithms under the drastic change obtain better performance than under the slight change in some cases.

5.3.4. *Effect of the number of peaks*

In the previous experiments, the number of peaks was fixed to 10 as shown in Table 1. One may be interested in the effect of the number of peaks on the performance of the three compared algorithms, i.e., CPSO, LTFR-DSPSO, and DyCODE. To this end, we run CPSO, LTFR-DSPSO, and DyCODE on our test suite with 15 peaks. To have a fair comparison, other parameter settings were consistent with those suggested in Section 5.2.

Tables 12–17 summarize the mean and standard deviation (abbreviated as "Mean OE" and "Std Dev") of the offline error obtained by the three compared algorithms on the six test instances with 10 and 15 peaks over 30 independent runs. In addition, Tables 18–20 provide the normalized score derived from the three compared algorithms in terms of the dimension, the number of the feasible regions, and the change severity of the feasible regions. Note that, the experimental results of the three compared algorithms with 10 peaks were directly taken from Tables 3–11.

As shown in Tables 12–17, CPSO, LTFR-DSPSO, and DyCODE provide similar offline error values on all the six test instances with 10 and 15 peaks, no matter what dimension and the value of s are. Moreover, Tables 18–20 also show similar normalized scores of CPSO, LTFR-DSPSO, and DyCODE for 10 and 15 peaks, regardless of the dimension, the number of the feasible regions, and the change severity of the feasible regions. The above experimental results signify that the number of peaks does not have a significant effect on the performance of the three compared algorithms. It is because we did not change the construction of the feasible regions.

Table 16
The mean and standard deviation of the offline error for test instance 5 with 10 peaks and 15 peaks. We also tested three different dimensions: 10D, 20D, and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 5	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	4.11E-01±6.58E-01	5.55E+00±7.32E+00	1.07E+01±6.99E+00	1.29E+00±2.25E+00	7.78E+00±8.47E+00	1.11E+01±7.66E+00
10D (s=2)	7.12E-01±1.90E+00	7.62E+00±8.32E+00	1.29E+01±7.91E+00	7.21E-01±1.30E+00	5.27E+00±5.42E+00	1.34E+01±9.44E+00
10D (s=3)	4.18E-01±9.96E-01	2.91E+00±3.26E+00	1.13E+01±6.63E+00	6.54E-01±1.21E+00	4.68E+00±5.96E+00	1.23E+01±8.71E+00
10D (s=4)	5.23E-01±7.77E-01	7.03E+00±6.76E+00	1.61E+01±8.24E+00	7.32E-01±1.16E+00	7.25E+00±6.54E+00	1.36E+01±8.33E+00
10D (s=5)	5.61E-01±8.31E-01	4.96E+00±5.31E+00	1.37E+01±8.31E+00	5.14E-01±1.33E+00	5.34E+00±6.63E+00	1.07E+01±6.54E+00
10D (s=6)	1.54E+00±2.49E+00	8.00E+00±8.39E+00	1.36E+01±9.14E+00	1.10E+00±2.25E+00	6.73E+00±6.54E+00	1.41E+01±8.47E+00
20D (s=1)	4.97E+00±1.77E+00	9.82E+00±6.48E+00	1.27E+01±7.69E+00	6.86E+00±3.80E+00	1.27E+01±7.10E+00	1.50E+01±1.10E+01
20D (s=2)	5.32E+00±2.90E+00	7.08E+00±6.43E+00	1.31E+01±9.87E+00	6.46E+00±3.87E+00	9.16E+00±6.72E+00	1.63E+01±1.04E+01
20D (s=3)	6.19E+00±2.85E+00	8.04E+00±4.61E+00	1.57E+01±1.14E+01	6.16E+00±2.75E+00	8.83E+00±6.01E+00	1.71E+01±1.01E+01
20D (s=4)	6.32E+00±3.14E+00	1.00E+01±6.44E+00	1.20E+01±8.09E+00	7.17E+00±4.06E+00	9.73E+00±6.20E+00	1.56E+01±1.01E+01
20D (s=5)	5.90E+00±2.75E+00	1.11E+01±7.00E+00	1.91E+01±1.20E+01	7.48E+00±3.68E+00	1.18E+01±9.43E+00	1.89E+01±1.24E+01
20D (s=6)	7.77E+00±3.78E+00	1.21E+01±5.38E+00	1.60E+01±9.43E+00	6.46E+00±2.91E+00	1.08E+01±6.37E+00	1.60E+01±9.83E+00
30D (s=1)	2.00E+01±5.59E+00	1.36E+01±6.43E+00	1.16E+01±7.61E+00	2.02E+01±6.00E+00	1.08E+01±6.16E+00	1.02E+01±9.62E+00
30D (s=2)	2.28E+01±5.01E+00	1.51E+01±5.82E+00	1.45E+01±8.54E+00	2.45E+01±7.08E+00	1.21E+01±7.49E+00	1.13E+01±7.41E+00
30D (s=3)	2.65E+01±7.31E+00	1.35E+01±6.24E+00	1.64E+01±9.30E+00	2.54E+01±7.70E+00	1.43E+01±8.36E+00	1.54E+01±9.51E+00
30D (s=4)	2.53E+01±5.23E+00	1.59E+01±7.61E+00	1.47E+01±9.16E+00	2.63E+01±5.08E+00	1.40E+01±6.48E+00	1.69E+01±7.53E+00
30D (s=5)	2.79E+01±6.32E+00	1.91E+01±9.12E+00	1.69E+01±8.18E+00	2.64E+01±5.61E+00	2.02E+01±9.67E+00	1.87E+01±8.95E+00
30D (s=6)	2.52E+01±4.78E+00	1.93E+01±9.48E+00	1.43E+01±7.97E+00	2.72E+01±4.78E+00	2.19E+01±7.92E+00	2.05E+01±1.08E+01

Table 17
The mean and standard deviation of the offline error for test instance 6 with 10 peaks and 15 peaks. We also tested three different dimensions: 10D, 20D, and 30D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance 6	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev	Mean OE±Std Dev
10D (s=1)	3.62E-01±6.32E-01	1.87E+00±1.49E+00	4.78E+00±3.11E+00	3.15E-01±5.61E-01	1.63E+00±1.08E+00	5.77E+00±4.67E+00
10D (s=2)	2.89E-01±5.74E-01	2.55E+00±1.93E+00	5.89E+00±4.25E+00	3.67E-01±6.14E-01	1.59E+00±1.07E+00	4.30E+00±2.80E+00
10D (s=3)	4.18E-01±4.00E-01	2.26E+00±1.47E+00	6.24E+00±3.76E+00	4.33E-01±7.48E-01	1.78E+00±9.09E-01	5.93E+00±2.95E+00
10D (s=4)	3.98E-01±7.54E-01	2.21E+00±1.15E+00	7.35E+00±3.52E+00	2.92E-01±5.99E-01	2.20E+00±1.41E+00	7.38E+00±5.25E+00
10D (s=5)	5.35E-01±1.02E+00	2.51E+00±1.78E+00	8.22E+00±4.45E+00	2.87E-01±3.46E-01	1.67E+00±1.17E+00	7.77E+00±5.13E+00
10D (s=6)	5.35E-01±6.52E-01	2.42E+00±1.79E+00	8.38E+00±4.76E+00	5.83E-01±1.11E+00	1.82E+00±1.17E+00	7.13E+00±3.44E+00
20D (s=1)	6.86E+00±2.47E+00	1.03E+01±4.45E+00	8.89E+00±5.49E+00	6.50E+00±2.39E+00	1.08E+01±5.05E+00	7.36E+00±5.84E+00
20D (s=2)	6.10E+00±1.63E+00	1.00E+01±4.61E+00	9.44E+00±5.21E+00	7.42E+00±2.96E+00	1.06E+01±3.88E+00	8.65E+00±8.01E+00
20D (s=3)	7.82E+00±2.48E+00	1.05E+01±4.42E+00	8.94E+00±6.77E+00	7.31E+00±3.23E+00	1.02E+01±3.96E+00	9.09E+00±5.94E+00
20D (s=4)	6.81E+00±2.49E+00	8.91E+00±3.61E+00	1.08E+01±9.34E+00	7.19E+00±2.47E+00	1.02E+01±5.42E+00	9.27E+00±6.64E+00
20D (s=5)	8.07E+00±2.37E+00	1.00E+01±5.28E+00	1.06E+01±6.64E+00	7.24E+00±2.76E+00	1.31E+01±5.02E+00	9.20E+00±6.10E+00
20D (s=6)	7.37E+00±2.89E+00	1.04E+01±6.47E+00	1.13E+01±7.16E+00	8.21E+00±2.30E+00	1.21E+01±5.69E+00	1.13E+01±8.13E+00
30D (s=1)	2.72E+01±5.37E+00	1.86E+01±4.15E+00	1.53E+01±4.36E+00	2.52E+01±5.14E+00	1.86E+01±5.19E+00	1.74E+01±6.81E+00
30D (s=2)	2.81E+01±6.29E+00	1.61E+01±7.67E+00	1.46E+01±6.31E+00	3.04E+01±5.39E+00	1.81E+01±6.91E+00	1.74E+01±6.47E+00
30D (s=3)	3.08E+01±6.15E+00	1.80E+01±6.36E+00	1.65E+01±7.08E+00	3.17E+01±5.88E+00	2.04E+01±7.99E+00	1.86E+01±6.42E+00
30D (s=4)	3.02E+01±5.91E+00	1.58E+01±5.59E+00	1.44E+01±6.65E+00	2.91E+01±6.23E+00	2.10E+01±6.92E+00	1.88E+01±6.75E+00
30D (s=5)	3.06E+01±6.54E+00	1.89E+01±5.08E+00	1.66E+01±5.91E+00	3.17E+01±5.93E+00	2.16E+01±7.70E+00	1.66E+01±4.73E+00
30D (s=6)	3.21E+01±6.11E+00	2.25E+01±8.69E+00	1.82E+01±6.42E+00	3.29E+01±7.18E+00	2.53E+01±7.50E+00	1.87E+01±6.08E+00

Table 18
The normalized scores of the three compared algorithms on different dimensions.

Dimension	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
10D	0.7027	0.8010	0.1951	0.6940	0.8048	0.2120
20D	0.6804	0.3904	0.3523	0.6707	0.3895	0.4486
30D	0.4461	0.4836	0.7670	0.4496	0.5023	0.7986

Table 19
The normalized scores of the three compared algorithms on different numbers of the feasible regions.

The number of the feasible regions	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
Single feasible region	0.2100	0.6599	0.7549	0.1926	0.6776	0.7843
Two feasible regions	0.9526	0.4108	0.2011	0.9550	0.4208	0.2682
Three feasible regions	0.6667	0.6044	0.3585	0.6666	0.5982	0.4068

Table 20
The normalized scores of the three compared algorithms on different change severities of the feasible regions.

The change severity of the feasible regions	10 peaks			15 peaks		
	CPSO	LTFR-DSPSO	DyCODE	CPSO	LTFR-DSPSO	DyCODE
Slight change	0.5697	0.7070	0.3443	0.5521	0.7306	0.4053
Drastic change	0.7468	0.3985	0.5255	0.6574	0.4005	0.5676

5.3.5. Effect of the optimal solution on the boundary of the feasible region

In many practical situations, the optimum solution is on the boundary of the feasible region. In fact, it is easy to generalize our test suite to have this characteristic by revising constraints as follows:

$$\begin{cases} \text{maximize } f(\vec{x},t) = \max\limits_{i=1,\ldots,p} \dfrac{H_i(t)}{1+W_i(t)\sum_{j=1}^{D}(x_j - X_{i,j}(t))^2} \\ \text{subject to: } g_1(\vec{x},t) = \sum\limits_{j=1}^{D}(x_j - \sqrt{D}/D * r_1(t) - X_{a_1,j}(t))^2 - r_1^2(t) \le 0 \\ \quad \text{or } g_2(\vec{x},t) = \sum\limits_{j=1}^{D}(x_j - \sqrt{D}/D * r_2(t) - X_{a_2,j}(t))^2 - r_2^2(t) \le 0 \\ \quad \vdots \\ \quad \text{or } g_l(\vec{x},t) = \sum\limits_{j=1}^{D}(x_j - \sqrt{D}/D * r_l(t) - X_{a_l,j}(t))^2 - r_l^2(t) \le 0 \end{cases} \tag{15}$$

where $g_k(\vec{x},t)$ denotes the kth constraint at environment t, $k \in \{1,\ldots,l\}$, l is the number of constraints, $X_{a_k,j}(t)$ is the jth dimension of the center of peak a_k at environment t, $r_k(t)$ is the radius of the kth feasible region at environment t, and D is the dimension. Fig. 9 illustrates the principle of (15). Compared with the original test suite, the center of each feasible region in (15) is moved by $\sqrt{D}/D * r_k(t)$ in each dimension. As a result, the local optimum of each feasible region is located on its boundary, and the best local optimum is the global optimum which is also located on the boundary of a certain feasible region.

To evaluate the influence of the optimal solution on the boundary of the feasible region, the performance of CPSO, LTFR-DSPSO and DyCODE on (15) with 10D is compared with that on the original test suite with 10D in which the optimal solution is in the center of the feasible region. For a fair comparison, all the three compared algorithms were implemented with the same parameter settings as suggested in Section 5.2.

Table 21 records the mean and standard deviation (abbreviated as "Mean OE" and "Std Dev") of the offline error provided by the three compared algorithms. Note that, the experimental results of the three compared algorithms on the original test suite were directly taken from Tables 3–8. It is clear from Table 21 that the performance of all the three compared algorithms degrades significantly under the condition that the optimal solution moves from the center of the feasible region to the boundary of the feasible region. The reason is explained as follows. For searching for the optimal solution on the boundary of the feasible region, an effective way is to recombine feasible solutions with infeasible solutions close to the boundary of the feasible region. However, in this paper, the feasibility rule was employed as the constraint-handling technique. It prefers feasible solutions to infeasible solutions.

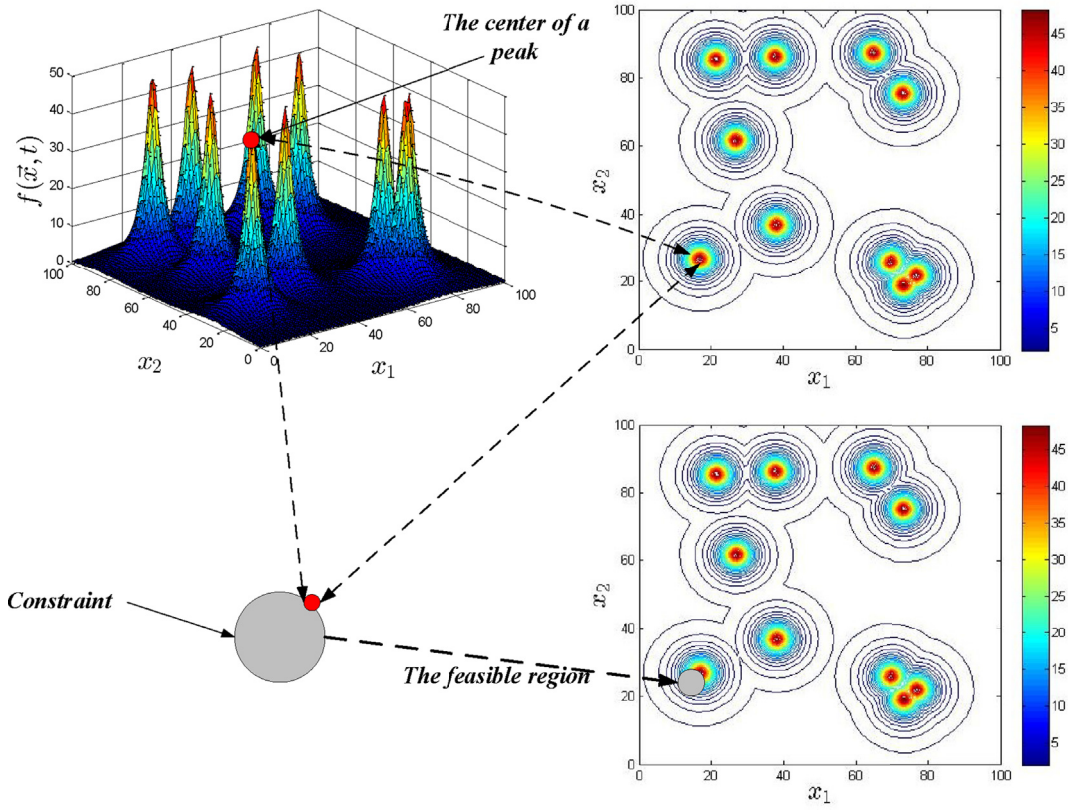


Fig. 9. Illustration of our test suite with the optimal solution on the boundary of the feasible region in a two-dimensional decision space.

Table 21
The mean and standard deviation of the offline error for six test instances with the optimal solution in the center of the feasible region and with the optimal solution on the boundary of the feasible region. The experiments were implemented on 10D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance	Shift Length	The Optimal Solution in the Center of the Feasible Region			The Optimal Solution on the Boundary of the Feasible Region		
		CPSO (10D) Mean OE±Std Dev	LTFR-DSPSO (10D) Mean OE±Std Dev	DyCODE (10D) Mean OE±Std Dev	CPSO (10D) Mean OE±Std Dev	LTFR-DSPSO (10D) Mean OE±Std Dev	DyCODE (10D) Mear OE±Std Dev
Instance 1	(*s*=1)	3.39E-01±1.10E+00	9.67E-05±1.30E-04	4.45E-02±1.09E-01	1.46E+00±3.59E+00	1.63E+00±1.71E+00	2.64E-01±8.49E-01
	(*s*=2)	5.49E-02±2.98E-01	6.39E-05±5.92E-05	5.71E-02±1.38E-01	9.56E-01±2.13E+00	8.90E-01±6.54E-01	5.07E-01±1.14E+00
	(*s*=3)	4.24E-01±1.13E+00	8.98E-05±7.59E-05	2.13E-01±5.62E-01	1.21E+00±3.46E+00	1.51E+00±1.51E+00	7.22E-01±1.27E+00
	(*s*=4)	1.12E-01±3.87E-01	1.15E-04±2.18E-04	4.09E-01±8.69E-01	6.87E-01±2.72E+00	1.64E+00±1.20E+00	1.03E+00±1.65E+00
	(*s*=5)	2.23E-01±7.26E-01	1.64E-04±1.76E-04	5.64E-01±9.98E-01	1.69E+00±4.21E+00	1.76E+00±1.31E+00	1.03E+00±1.65E+00
	(*s*=6)	9.86E-01±2.08E+00	1.21E-04±1.01E-04	3.84E-01±9.14E-01	4.87E+00±9.73E+00	2.22E+00±1.69E+00	1.45E+00±2.56E+00
Instance 2	(*s*=1)	1.14E+00±2.05E+00	4.98E-03±5.07E-03	2.48E-01±6.22E-01	3.68E+00±4.52E+00	2.81E-01±1.78E-01	1.02E+00±1.80E+00
	(*s*=2)	7.38E-01±1.93E+00	3.58E-03±5.02E-03	1.01E-01±3.40E-01	2.73E+00±3.78E+00	2.76E-01±1.63E-01	2.53E+00±2.79E+00
	(*s*=3)	2.79E+00±4.39E+00	8.10E-03±1.59E-02	1.89E-01±8.48E-01	5.60E+00±6.90E+00	4.16E-01±5.29E-01	1.76E+00±1.78E+00
	(*s*=4)	2.13E+00±3.70E+00	2.96E-03±3.09E-03	2.62E-01±7.03E-01	5.55E+00±5.35E+00	3.40E-01±2.00E-01	2.65E+00±2.93E+00
	(*s*=5)	1.11E+00±3.19E+00	4.47E-03±5.12E-03	3.78E-01±1.59E+00	4.08E+00±4.76E+00	3.14E-01±2.66E-01	2.21E+00±2.60E+00
	(*s*=6)	1.82E+00±2.46E+00	5.42E-03±5.44E-03	1.59E-01±4.48E-01	7.02E+00±7.22E+00	4.23E-01±3.78E-01	2.32E+00±3.52E+00
Instance 3	(*s*=1)	1.20E+00±3.20E+00	3.61E+00±4.94E+00	6.05E+00±6.36E+00	1.21E+00±2.41E+00	4.88E+00±3.61E+00	7.38E+00±6.78E+00
	(*s*=2)	1.73E-01±9.27E-01	2.84E+00±5.48E+00	8.95E+00±8.15E+00	1.10E+00±2.45E+00	6.06E+00±4.47E+00	1.09E+01±7.52E+00
	(*s*=3)	7.76E-01±1.78E+00	2.22E+00±3.23E+00	7.12E+00±7.22E+00	1.36E+00±2.56E+00	5.33E+00±4.31E+00	8.19E+00±6.02E+00
	(*s*=4)	7.10E-01±1.46E+00	2.05E+00±2.38E+00	6.71E+00±5.61E+00	6.01E-01±1.10E+00	5.54E+00±3.00E+00	1.03E+01±6.22E+00
	(*s*=5)	8.22E-01±1.54E+00	3.46E+00±6.02E+00	9.27E+00±6.51E+00	1.47E+00±2.66E+00	5.57E+00±5.14E+00	9.00E+00±6.38E+00
	(*s*=6)	2.09E+00±3.48E+00	5.17E+00±7.53E+00	1.12E+01±8.69E+00	3.41E+00±5.53E+00	8.69E+00±6.57E+00	1.24E+01±8.42E+00
Instance 4	(*s*=1)	2.66E-01±5.15E-01	7.93E-01±6.48E-01	2.74E+00±2.51E+00	5.18E-01±8.46E-01	3.01E+00±2.07E+00	6.57E+00±4.74E+00
	(*s*=2)	1.55E-01±4.36E-01	9.58E-01±8.04E-01	2.95E+00±2.08E+00	7.81E-01±1.84E+00	2.63E+00±1.82E+00	8.04E+00±4.39E+00
	(*s*=3)	9.35E-01±2.55E+00	7.14E-01±5.44E-01	3.80E+00±2.21E+00	1.82E+00±2.84E+00	3.29E+00±1.68E+00	8.00E+00±4.41E+00
	(*s*=4)	8.96E-01±1.59E+00	9.89E-01±7.60E-01	4.00E+00±3.19E+00	1.36E+00±2.86E+00	3.94E+00±2.72E+00	6.06E+00±3.97E+00
	(*s*=5)	4.52E-01±1.15E+00	1.10E+00±8.61E-01	3.84E+00±2.91E+00	1.33E+00±2.19E+00	3.95E+00±2.05E+00	7.38E+00±4.67E+00
	(*s*=6)	7.23E-01±1.39E+00	1.40E+00±1.14E+00	5.72E+00±5.24E+00	1.46E+00±2.67E+00	3.84E+00±2.23E+00	9.12E+00±5.98E+00
Instance 5	(*s*=1)	4.11E-01±6.58E-01	5.55E+00±7.32E+00	1.07E+01±6.99E+00	6.64E-01±1.22E+00	1.04E+01±6.94E+00	1.39E+01±7.53E+00
	(*s*=2)	7.12E-01±1.90E+00	7.62E+00±8.32E+00	1.29E+01±7.91E+00	1.21E+00±3.36E+00	1.00E+01±6.18E+00	1.66E+01±8.21E+00
	(*s*=3)	4.18E-01±9.96E-01	2.91E+00±3.26E+00	1.13E+01±6.63E+00	1.05E+00±2.20E+00	6.91E+00±3.43E+00	1.66E+01±9.97E+00
	(*s*=4)	5.23E-01±7.77E-01	7.03E+00±6.76E+00	1.61E+01±8.24E+00	1.01E+00±1.57E+00	9.82E+00±3.55E+00	1.59E+01±8.30E+00
	(*s*=5)	5.61E-01±8.31E-01	4.96E+00±5.31E+00	1.37E+01±8.31E+00	1.24E+00±1.72E+00	9.58E+00±6.10E+00	1.27E+01±7.99E+00
	(*s*=6)	1.54E+00±2.49E+00	8.00E+00±8.39E+00	1.36E+01±9.14E+00	3.02E+00±4.59E+00	1.18E+01±4.78E+00	1.69E+01±7.38E+00
Instance 6	(*s*=1)	3.62E-01±6.32E-01	1.87E+00±1.49E+00	4.78E+00±3.11E+00	4.59E-01±6.50E-01	5.23E+00±2.87E+00	9.34E+00±5.80E+00
	(*s*=2)	2.89E-01±5.74E-01	2.55E+00±1.93E+00	5.89E+00±4.25E+00	6.41E-01±8.39E-01	5.01E+00±2.75E+00	1.05E+01±5.36E+00
	(*s*=3)	4.18E-01±4.00E-01	2.26E+00±1.47E+00	6.24E+00±3.76E+00	1.31E+00±2.16E+00	5.92E+00±2.74E+00	1.09E+01±5.64E+00
	(*s*=4)	3.98E-01±7.54E-01	2.21E+00±1.15E+00	7.35E+00±3.52E+00	1.10E+00±1.83E+00	5.16E+00±2.25E+00	9.32E+00±4.60E+00
	(*s*=5)	5.35E-01±1.02E+00	2.51E+00±1.78E+00	8.22E+00±4.45E+00	9.66E-01±1.48E+00	5.29E+00±2.83E+00	1.05E+01±6.66E+00
	(*s*=6)	5.35E-01±6.52E-01	2.42E+00±1.79E+00	8.38E+00±4.76E+00	1.39E+00±1.55E+00	5.94E+00±3.26E+00	1.27E+01±7.16E+00

Thus, infeasible solutions are always deleted during the evolution. As a result, it is difficult for the three compared algorithms to obtain the global optimum.

5.3.6. *The number of feasible regions changes in different environments*

In complex real-world environments, the number of feasible regions may change in different environments. In fact, it is easy to generalize our test suite to have this characteristic by revising some relevant parameter settings. Next, we introduce two additional test instances and their settings of l and a_k are summarized in Table 22.

Test instance 7 has three different kinds of feasible regions, which change slightly in the decision space due to the tracking of some fixed peaks. In the first ten environments, the feasible region tracks a fix peak ($a_1 = 1$); in the second ten environments, the feasible regions track two fix peaks ($a_1 = 1$ and $a_2 = 6$); and in the third ten environments, the feasible regions track three fix peaks ($a_1 = 1$, $a_2 = 6$, and $a_3 = 10$).

Test instance 8 also has three different kinds of feasible regions, which change drastically in the decision space due to the tracking of the highest peaks. In the first ten environments, the feasible region tracks the highest peak (a_1 is the index of the highest peak); in the second ten environments, the feasible regions track the two highest peaks (a_1 and a_2 are the indexes of the two highest peaks); and in the third ten environments, the feasible regions track the three highest peaks (a_1, a_2, and a_3 are the indexes of the three highest peaks).

Table 23 records the mean and standard deviation (abbreviated as "Mean OE" and "Std Dev") of the offline error provided by the three compared algorithms. It is clear from Table 23 that the performance of CPSO is better than that of the two competitors for DCOPs with different kinds of feasible regions. The above phenomenon can be explained as follows. Test instance 7 and test instance 8 somehow like a special case of the test instance with multiple feasible regions. As explained in Section 5.3.2, in the case of multiple feasible regions, CPSO achieves the best performance.

Table 22
Settings of l and a_k ($k = 1, \ldots, l$) of test instance 7, test instance 8, and test instance 9.

Test Instance	l	a_k ($k = 1, \ldots, l$)
Test Instance 7	1, 2, 3	$a_1 = 1, t = 1, \ldots, 10$.
		$a_1 = 1$ and $a_2 = 6, t = 11, \ldots, 20$.
		$a_1 = 1, a_2 = 6$, and $a_3 = 10, t = 21, \ldots, 30$.
Test Instance 8	1, 2, 3	a_1 is the index of the highest peak, $t = 1, \ldots, 10$.
		a_1 and a_2 are the indexes of the two highest peaks, $t = 11, \ldots, 20$.
		a_1, a_2, and a_3 are the indexes of the three highest peaks, $t = 21, \ldots, 30$.
Test Instance 9	1	$a_1 = 1, t = 1$.
		$a_1 = 2, t = 2$.
		⋮
		$a_1 = 10, t = 10$.

Table 23
The mean and standard deviation of the offline error for test instance 7 and test instance 8. The experiments were implemented on 10D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance	Shift Length	CPSO (10D) Mean OE±Std Dev	LTFR-DSPSO (10D) Mean OE±Std Dev	DyCODE (10D) Mean OE±Std Dev
Test Instance 7	s=1	3.10E-01±7.07E-01	6.62E+00±6.54E+00	7.47E+00±6.33E+00
	s=2	9.24E-01±1.38E+00	5.49E+00±4.50E+00	8.44E+00±5.30E+00
	s=3	1.11E+00±1.56E+00	6.14E+00±4.50E+00	7.88E+00±4.88E+00
	s=4	1.07E+00±1.09E+00	6.74E+00±5.43E+00	9.87E+00±5.18E+00
	s=5	9.13E-01±1.39E+00	4.33E+00±4.17E+00	8.64E+00±4.70E+00
	s=6	9.28E-01±1.44E+00	6.09E+00±5.37E+00	9.66E+00±4.72E+00
Test Instance 8	s=1	5.15E-01±8.51E-01	1.34E+00±6.79E-01	2.70E+00±1.55E+00
	s=2	9.70E-01±1.42E+00	1.69E+00±8.81E-01	3.51E+00±1.85E+00
	s=3	6.05E-01±7.67E-01	1.14E+00±6.59E-01	3.94E+00±2.16E+00
	s=4	1.31E+00±1.32E-01	1.70E+00±1.00E+00	3.64E+00±2.04E+00
	s=5	5.33E-01±1.05E-01	1.79E+00±1.23E+00	4.63E+00±2.94E+00
	s=6	6.96E-01±8.23E-01	1.66E+00±1.05E+00	4.02E+00±2.00E+00

Table 24
The mean and standard deviation of the offline error for test instance 9. The experiments were implemented on 10D. The best and second best mean offline error values among all the algorithms in each case are highlighted in gray and light gray, respectively.

Test Instance	Shift Length	CPSO (10D) Mean OE±Std Dev	LTFR-DSPSO (10D) Mean OE±Std Dev	DyCODE (10D) Mean OE±Std Dev
Test Instance 9	s=1	1.28E+00±1.77E+00	1.84E-02±2.00E-02	2.70E-02±5.78E-01
	s=2	1.65E+00±2.14E+00	2.75E-02±1.85E-02	2.83E-02±8.08E-01
	s=3	1.84E+00±2.32E+00	3.18E-02±1.88E-02	1.30E-01±4.18E-01
	s=4	1.84E+00±2.53E+00	2.66E-02±1.74E-02	1.87E-01±4.53E-01
	s=5	1.69E+00±2.36E+00	2.72E-02±1.61E-02	3.97E-02±1.58E-01
	s=6	1.61E+00±2.39E+00	2.33E-02±1.21E-02	4.02E-02±9.95E-01

5.3.7. *Static objective function with dynamic constraints*

In some real-world environments, it may exist static objective function with dynamic constraints. Similarly, it is simple to generalize our test suite to have this characteristic by revising some relevant parameter settings. Next, we introduce another test instance and its settings of l and a_k are summarized in Table 22.

Test instance 9 has a single feasible region, the objective function of which is static. In the first environment, the feasible region tracks the first peak ($a_1 = 1$); in the second environment, the feasible region tracks the second peak ($a_1 = 2$); in the third environment, the feasible region tracks the third peak ($a_1 = 3$), and so fourth.

The mean and standard deviation (abbreviated as "Mean OE" and "Std Dev") of the offline error provided by the three compared algorithms are summarized in Table 24. From Table 24, LTFR-DSPSO outperforms the two competitors for DCOPs with static objective function and dynamic constraints. This phenomenon can be attributed to the fact that test instance 9 somehow likes a mixture of feasible regions with slight changes and a single feasible region. As explained in Section 5.3.2 and Section 5.3.3, in the case of feasible regions with slight changes, LTFR-DSPSO performs better than the two competitors, and in the case of a single feasible region, LTFR-DSPSO has good performance.

Remark 1. From all the above experimental analysis, we can give the following comments:

- In the case of $D = 10$, LTFR-DSPSO performs the best; in the case of $D = 20$, CPSO is superior to the other two algorithms; and in the case of $D = 30$, DyCODE has the best performance.
- When the feasible regions change slightly, LTFR-DSPSO outperforms the two competitors, and CPSO ranks the first when the feasible regions change drastically.
- DyCODE achieves the best performance on DCOPs with single feasible region and CPSO performs the best on DCOPs with multiple feasible regions.
- The three compared algorithms are insensitive to the change of the shift length s. With the increase of s, the performance of the three compared algorithms drops slightly.

6. Conclusion

In the evolutionary computation research community, dynamic constrained optimization is still in its infant stage. Therefore, it is urgent to design a standard test suite to advance the development of this area. A standard test suite can not only attract more researchers to pay more attention to this area, but also be used to design powerful algorithms to solve DCOPs. Through extensive investigations, we found that existing test functions of DCOPs could not exhibit multi-facet properties to test the performance of a DCOEA. Moreover, the algorithm comparisons in the area remain scarce. Based on the above consideration, this paper constructed a test suite for dynamic constrained optimization, which contains the following five characteristics, i.e., scalability, adjustability, multi-modality, different change severities of feasible regions, and the global and local optima known. We also designed six test instances on the basis of the proposed test suite.

Thereafter, the performance of three DCOEAs was evaluated and compared on the six test instances. One of these DCOEAs is DyCODE proposed in this paper. The experimental results demonstrated that none of them is able to efficiently solve these six test instances, and that different algorithms show their strengths and weaknesses on different types of test instances. For DCOPs with a single feasible region, DyCODE performs the best. For DCOPs with multiple feasible regions, CPSO achieves the best performance. LTFR-DSPSO outperforms the other two competitors when the feasible regions change slightly. When the feasible regions change drastically, CPSO obtains the best performance. With the increase of dimension, the performance of CPSO and LTFR-DSPSO drops faster than that of DyCODE for the reason that they have some strategies related to the dimension.

The future work will be carried out from the following four aspects:

- In this paper, the constraints of our test suite are still relatively simple, and we only consider convex constraints. In the future, it is necessary to design concave, nonlinear, and mixed constraints.
- To ensure a fair comparison among three DCOEAs, the feasibility-based rule [32] is adopted in CPSO and DyCODE. We will try other kinds of constraint-handling techniques, such as stochastic ranking [45], the ϵ constrained method [46], and multiobjective optimization-based methods [47].
- Through the experimental comparisons, we ascertain which kind of strategy suits which type of DCOPs. Inspired by Ref. [48], we will hybridize CPSO, LTFR-DSPSO, and DyCODE based on their contributions to design more excellent and stable algorithms in the future.
- There are a lot of DCOPs in engineering fields, such as adaptive walking of humanoid robots [49], dynamic controller design [50], vehicle routing problems [51], ship scheduling problems [2], and so on. In the future, we intend to carry out the real-world applications of dynamic constrained optimization.

Acknowledgment

The authors would like to thank Prof. Changhe Li and Dr. Chenyang Bu for kindly providing the source codes of CPSO and LTFR-DSPSO, respectively.

This work was supported in part by the Innovation-Driven Project of Central South University under Grant 2018CX010, in part by the National Natural Science Foundation of China under Grant 61673397, in part by the Hunan Provincial Natural Science Fund for Distinguished Young Scholars (Grant No. 2016JJ1018), and in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS06.

References

[1] T.T. Nguyen, Continuous Dynamic Optimisation Using Evolutionary Algorithms. (PhD thesis), University of Birmingham, 2011.
[2] K. Mertens, T. Holvoet, Y. Berbers, The DynCOAA algorithm for dynamic constraint optimization problems, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, 2006, pp. 1421–1423.
[3] N. Jin, M. Termansen, K. Hubacek, J. Holden, M. Kirkby, Adaptive farming strategies for dynamic economic environment, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 1213–1220.
[4] J.A. D Atkin, E.K. Burke, J.S. Greenwood, D. Reeson, On-line decision support for take-off runway scheduling with uncertain taxi times at london heathrow airport, J. Sched. 11 (5) (2008) 323.
[5] J.J. Pantrigo, A. Sánchez, A.S. Montemayor, A. Duarte, Multi-dimensional visual tracking using scatter search particle filter, Pattern Recognit. Lett. 29 (8) (2008) 1160–1174.
[6] C. Sonntag, W. Su, O. Stursberg, S. Engell, Optimized start-up control of an industrial-scale evaporation system with hybrid dynamics, Contr. Eng. Pract. 16 (8) (2008) 976–990.
[7] P. Mitra, G.K. Venayagamoorthy, Real time implementation of an artificial immune system based controller for a dstatcom in an electric ship power system, in: 2008 IEEE Industry Applications Society Annual Meeting, IEEE, 2008, pp. 1–8.
[8] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the 1999 Congress on Evolutionary Computation, vol. 3, IEEE, 1999, pp. 1875–1882.
[9] T.T. Nguyen, X. Yao, Benchmarking and solving dynamic constrained problems, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 690–697.
[10] H. Richter, Memory design for constrained dynamic optimization problems, Appl. Evolut. Comput. (2010) 552–561.
[11] C.A. Liu, New dynamic constrained optimization PSO algorithm, in: Fourth International Conference on Natural Computation, ICNC'08, vol. 7, IEEE, 2008, pp. 650–653.
[12] C. Bu, W. Luo, L. Yue, Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies, IEEE Trans. Evol. Comput. 21 (1) (2017) 14–33.
[13] Z. Zhang, S. Yue, M. Liao, F. Long, Danger theory based artificial immune system solving dynamic constrained single-objective optimization, Soft Comput. 18 (1) (2014) 185–206.
[14] T.T. Nguyen, X. Yao, Continuous dynamic constrained optimization the challenges, IEEE Trans. Evol. Comput. 16 (6) (2012) 769–786.
[15] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, J. Appl. Mech. 41 (8) (2006).
[16] R. Mallipeddi, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the Cec 2010 Competition on Constrained Real-Parameter Optimization, Nanyang Technological University, Singapore, 24, 2010.
[17] H.K. Singh, A. Isaacs, T.T. Nguyen, T. Ray, X. Yao, Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 3127–3134.
[18] K. Alam, T. Ray, S.G. Anavatti, Practical application of an evolutionary algorithm for the design and construction of a six-inch submarine, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 2825–2832.
[19] V.S. Aragón, S.C. Esquivel, C.A. Coello Coello, Artificial immune system for solving dynamic constrained optimization problems, in: Metaheuristics for Dynamic Optimization, Springer, 2013, pp. 225–263.
[20] V.S. Aragón, S.C. Esquivel, C.A. Coello Coello, Artificial immune system for solving global optimization problems, Intel. Artif. 14 (46) (2010) Revista Iberoamericana de Inteligencia Artificial.
[21] K. Pal, C. Saha, S. Das, C.A. Coello Coello, Dynamic constrained optimization with offspring repair based gravitational search algorithm, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 2414–2421.
[22] M.Y. Ameca-Alducin, E. Mezura-Montes, N. Cruz-Ramirez, Differential evolution with combined variants for dynamic constrained optimization, in: 2014 IEEE Congress on Evolutionary Computation, IEEE, 2014, pp. 975–982.
[23] M.Y. Ameca-Alducin, E. Mezura-Montes, N. Cruz-Ramírez, A repair method for differential evolution with combined variants to solve dynamic constrained optimization problems, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, 2015, pp. 241–248.
[24] H.G. Cobb, An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-dependent Nonstationary Environments, Technical report, Naval Research Lab Washington DC, 1990.
[25] J.J. Grefenstette, Genetic algorithms for changing environments, in: International Conference on Parallel Problem Solving from Nature (PPSN), vol. 2, 1992, pp. 137–144.
[26] H. Richter, S. Yang, Memory based on abstraction for dynamic fitness functions, Appl. Evolut. Comput. (2008) 596–605.
[27] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Trans. Evol. Comput. 10 (4) (2006) 440–458.
[28] C.A. Liu, New method for solving a class of dynamic nonlinear constrained optimization problems, in: The Sixth International Conference on Natural Computation (ICNC), vol. 5, IEEE, 2010, pp. 2400–2402.
[29] H. Richter, F. Dietel, Solving dynamic constrained optimization problems with asynchronous change pattern, in: European Conference on the Applications of Evolutionary Computation, Springer, 2011, pp. 334–343.
[30] X. Lu, K. Tang, X. Yao, Speciated evolutionary algorithm for dynamic constrained optimisation, in: International Conference on Parallel Problem Solving from Nature, Springer, 2016, pp. 203–213.
[31] S. Yang, C. Li, A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments, IEEE Trans. Evol. Comput. 14 (6) (2010) 959–974.
[32] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2) (2000) 311–338.
[33] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[34] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[35] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[36] Y. Wang, H. Liu, H. Long, Z. Zhang, S. Yang, Differential evolution with a new encoding mechanism for optimizing wind farm layout, IEEE Trans. Ind. Inf. 14 (3) (2018) 1040–1054.
[37] Y. Wang, B. Xu, G. Sun, S. Yang, A two-phase differential evolution for uniform designs in constrained experimental domains, IEEE Trans. Evol. Comput. 21 (5) (2017) 665–680.
[38] Z. Liu, Y. Wang, S. Yang, K. Tang, An adaptive framework to tune the coordinate systems in nature-inspired optimization algorithms, IEEE Trans. Cybern. 49 (4) (April 2019) 1403–1416.
[39] Y. Wang, D. Yin, S. Yang, G. Sun, Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints, IEEE Trans. Cybern. 49 (5) (May 2019) 1642–1656.
[40] B.C. Wang, H.X. Li, J.P. Li, Y. Wang, Composite differential evolution for constrained evolutionary optimization, IEEE Trans. Syst., Man, Cybern.: Systems (99) (2018) 1–14.
[41] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution–an updated survey, Swarm Evolut. Comput. 27 (2016) 1–30.
[42] J. Del Ser, E. Osaba, D. Molina, X.S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A. Coello Coello, F. Herrera, Bio-inspired computation: where we stand and what's next, Swarm Evolut. Comput. 48 (2019) 220–250.
[43] Y. Wang, Z. Cai, A dynamic hybrid framework for constrained evolutionary optimization, IEEE Trans. Syst., Man, Cybern., Part B (Cybern.) 42 (1) (2012) 203–217.

[44] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, Swarm Evolut. Comput. 6 (2012) 1–24.
[45] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Trans. Evol. Comput. 4 (3) (2000) 284–294.
[46] T. Takahama, S. Sakai, Constrained optimization by the constrained differential evolution with an archive and gradient-based mutation, in: 2010 IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–9.
[47] Z. Cai, Y. Wang, A multiobjective optimization-based evolutionary algorithm for constrained optimization, IEEE Trans. Evol. Comput. 10 (6) (2006) 658–675.
[48] J. Zou, Q. Li, S. Yang, J. Zheng, Z. Peng, T. Pei, A dynamic multiobjective evolutionary algorithm based on a dynamic evolutionary environment model, Swarm Evolut. Comput. 44 (2019) 247–259.
[49] C. Liu, Q. Chen, D. Wang, CPG-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots, IEEE Trans. Syst., Man, Cybern., Part B (Cybern.) 41 (3) (2011) 867–880.
[50] X. Huang, K. Hamad, S. Amit, X. Zhang, Variant PID controller design for autonomous visual tracking of oil and gas pipelines via an unmanned aerial vehicle, in: 2017 17th International Conference on Control, Automation and Systems (ICCAS), IEEE, 2017, pp. 368–372.
[51] P. Ioannou, A. Chassiakos, H. Jula, R. Unglaub, Dynamic Optimization of Cargo Movement by Trucks in Metropolitan Areas with Adjacent Ports, METRANS Transportation Center, University of Southern California, Los Angeles, CA, 2002. 90089:00–15.