A General Framework of Dynamic Constrained Multiobjective Evolutionary Algorithms for Constrained Optimization

Sanyou Zeng, Ruwang Jiao, Changhe Li, *Member, IEEE*, Xi Li, and Jawdat S. Alkasassbeh

***Abstract*—A novel multiobjective technique is proposed for solving constrained optimization problems (COPs) in this paper. The method highlights three different perspectives: 1) a COP is converted into an equivalent dynamic constrained multiobjective optimization problem (DCMOP) with three objectives: a) the original objective; b) a constraint-violation objective; and c) a niche-count objective; 2) a method of gradually reducing the constraint boundary aims to handle the constraint difficulty; and 3) a method of gradually reducing the niche size aims to handle the multimodal difficulty. A general framework of the design of dynamic constrained multiobjective evolutionary algorithms is proposed for solving DCMOPs. Three popular types of multiobjective evolutionary algorithms, i.e., Pareto ranking-based, decomposition-based, and hype-volume indicator-based, are employed to instantiate the framework. The three instantiations are tested on two benchmark suites. Experimental results show that they perform better than or competitive to a set of state-of-the-art constraint optimizers, especially on problems with a large number of dimensions.**

***Index Terms*—Constrained optimization, dynamic multiobjective optimization, evolutionary computation, multiobjective optimization.**

I. Introduction

EVOLUTIONARY algorithms (EAs) have been widely employed for solving constrained optimization problems (COPs). Many constrained EAs (CEAs) have been proposed. Constraint handling techniques can be classified into the following categories: feasibility rules [1]–[3], stochastic ranking [4], [5], ϵ-constrained methods [6], novel penalty functions [7], [8], novel special operators [9], [10], and ensemble of constraint-handling techniques, and last, but not least, multiobjective technologies [11], [12]. EAs have been also employed to handle constraints for multiobjective optimization problems (MOPs) (e.g., [16]–[21]).

Multiobjective optimization has been one of the hottest research topics in the area of evolutionary computation, and many algorithms have been proposed in this area, e.g., some of the most recent ones [22]–[28]. Although some of the multiobjective techniques have been applied to COPs by transforming COPs to MOPs, the transformed MOPs are not equivalent to the original COPs. Therefore, the motivation of this paper is to provide a method for converting a COP into an equivalent dynamic constrained MOP (DCMOP) and to develop a general algorithm framework for solving DCMOPs using multiobjective EAs (MOEAs).

The work of this paper is based on our previous work [29]–[32]. The basic idea of this method was first proposed in [29], where a COP was converted into an equivalent DCMOP with two objectives: the original objective and the constraint violation. In [30], a dynamic version of the elitist nondominated sorting genetic algorithm (NSGA-II [33]) was proposed to solve DCMOPs. In [31], we converted the design of linear sparse antenna arrays problem to the constrained MOP (CMOP) and the CMOP was further converted into the DCMOP. A dynamic constrained MOEA (DCMOEA) was implemented to solve the DCMOP, consequently the original antenna design problem was solved. Recently, the COP was converted into the dynamic constrained many-objective problem, where objectives comprise the original objective and objectives of each constraint [32]. The third version of NSGA (NSGA-III) [34] for many-objective problems was employed to solve the problem in [32]. In this paper, we introduce a new objective: the niche-count for handling the multimodal difficulty. Dynamic versions of three existing best-in-class MOEAs are employed to comprehensively verify the effectiveness of the method in this paper.

The rest of this paper is organized as follows. Section II introduces some related work on multiobjective techniques for solving COPs. Section III explains procedures for converting a COP into an equivalent DCMOP. Section IV presents a framework of DCMOEA, and three instantiations of the DCMOEA framework are implemented. Experimental

Manuscript received June 17, 2016; revised September 28, 2016 and December 24, 2016; accepted December 28, 2016. Date of publication January 16, 2017; date of current version August 16, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61271140, Grant 61673355, and Grant 61203306, in part by the Hubei Provincial Natural Science Foundation of China under Grant 2015CFA010, and in part by the 111 Project under Grant B17040. This paper was recommended by Associate Editor H. Ishibuchi. *(Corresponding author: Changhe Li.)*

S. Zeng, R. Jiao, and J. S. Alkasassbeh are with the School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan 430074, China (e-mail: sanyouzeng@gmail.com; ruwangjiao@gmail.com; jawdat1983@yahoo.com).

C. Li is with the School of Automation, China University of Geosciences, Wuhan 430074, China, and also with the Hubei key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, Wuhan 430074, China (e-mail: changhe.lw@gmail.com).

X. Li is with the School of Information Engineering, Hebei GEO University, Shijiazhuang 050031, China (e-mail: lixi_sjz@foxmail.com).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2647742

results and discussions are provided in Section V. Section VI concludes this paper and discusses several future research directions.

II. Related Work

In this section, we briefly review the related work on using multiobjective techniques for handling constraints in the literature of EAs for COPs.

Constraint handling methods are one of the major types of methods for solving COPs. Mezura-Montes and Coello [11] emphasized the importance of combining feasibility rules with other mechanisms, e.g., retaining infeasible solutions which are close to the feasible region. Ray *et al.* [35] proposed an infeasibility driven EA, called infeasibility driven EA (IDEA), where the replacement process requires a proportion of infeasible solutions to persist in the population for the next generation. A new improvement of the approach can be seen in [36]. Venkatraman and Yen [37] proposed a two-step strategy for solving COPs. The first step focuses only on satisfying the problem constraints. The second step starts to optimize the objective function value.

Deb *et al.* [38] proposed a hybrid reference-point-based evolutionary multiobjective optimization (EMO) algorithm coupled with the classical sequential quadratic programming procedure for solving COPs. Deb and Datta [8] combined a bi-objective evolutionary approach with the penalty function methodology in a manner complementary to each other. Thereafter, they dynamically tuned the penalty parameters and relaxed constraints with a constant boundary [39]. A review of constraint handling techniques can be seen in [40] and [41].

Different types of optimization algorithms are also widely used to address COPs. Venter and Haftka [42] transformed a COP into a bi-objective optimization problem, then solved the problem by the particle swarm optimization (PSO). The leader selection, for most of the time, is based on the sum of constraint violation. However, the selection sometimes is based on the following choices: the original objective function, the crowding distance or Pareto dominance.

Li *et al.* [43] also transformed a COP into a bi-objective optimization problem and solved it with a PSO algorithm in which the Pareto dominance is used as a criterion in the update process of particles' personal best positions as well as in the selection of the local-best leaders in a neighborhood. The approach was tested on three engineering design problems.

Runarsson and Yao [4] proposed a stochastic ranking method based on evolution strategy (ES) using a stochastic lexicographic order to ignore constraint violations with a probability. These methods have been successfully applied to various problems. Cai and Wang's group [44] proposed a multiobjective optimization based EA for solving COPs. The authors integrated additional techniques such as differential evolution (DE) and dynamic techniques into CEAs in [45] and [46]. Reynoso-Meza *et al.* [47] proposed a spherical-pruning multiobjective DE algorithm to solve COPs. An external archive is used to store nondominated solutions.

III. Converting the COP to the Equivalent DCMOP

In the beginning of this section, we would like to clarify several statements: 1) a CEA is to solve a COP; 2) an unconstrained MOEA (UMOEA) is to solve an unconstrained MOP (UMOP); 3) a CMOEA is to solve a CMOP; and 4) a DCMOEA is to solve a DCMOP.

A. Constrained Optimization Problem

Without loss of generality, minimization optimization is assumed in this paper.

Definition 1 (COP): A general COP consists of a set of n variables, an objective function, and a set of m constraints. The optimization goal is to

$$\begin{aligned} \min\ & f(\vec{x}) \\ \text{st:}\ & \vec{g}(\vec{x}) = (g_1(\vec{x}), g_2(\vec{x}), \ldots, g_m(\vec{x})) \le \vec{0} \\ \text{where}\ & \vec{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{X} \\ & \mathbf{X} = \left\{\vec{x} | \vec{l} \le \vec{x} \le \vec{u}\right\} \\ & \vec{l} = (l_1, l_2, \ldots, l_n), \vec{u} = (u_1, u_2, \ldots, u_n) \end{aligned} \tag{1}$$

where $\vec{x}$ is the solution vector (solution for short) and $\mathbf{X}$ denotes the solution space, $\vec{l}$ and $\vec{u}$ are the lower bound and upper bound of the solution space, f is the objective function, $\vec{g} \le \vec{0}$ is the vector of constraints, and $\vec{0}$ denotes the constraint boundary.

Note that an equality constraint $h(\vec{x}) = 0$ is converted into an inequality constraint, i.e., $|h(\vec{x})| - \delta \le 0$ for numerical computation where δ is a positive close-to-zero number, and $\delta = 0.0001$ is used in this paper.

Definition 2 (Feasible): A solution $\vec{x} \in \mathbf{X}$ is said to be feasible, if $\vec{g}(\vec{x}) \le \vec{0}$; otherwise $\vec{x}$ is said to be infeasible.

Definition 3 (Feasible Set): The feasible set $\mathbf{X_F}$, is defined as the set of all feasible solutions $\vec{x} \in \mathbf{X}$.

B. Conversion of the COP to the DCMOP

The motivation of this paper is to adopt the constraint-violation and niche count to solve COPs. Therefore, we first construct a DCMOP based on a COP.

1) Constraint-Violation: The constraint-violation is employed to handle the constraint difficulty. The violation of a constraint is usually evaluated by

$$G_i(\vec{x}) = \max\{g_i(\vec{x}), 0\}, i = 1, 2, \ldots, m. \tag{2}$$

Definition 4 (Constraint-Violation): The constraint-violation of a solution is defined as the average of the normalized violations of all constraints of the solution

$$\text{cv}(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m}\frac{G_i(\vec{x})}{\max\limits_{\vec{x}\in\mathbf{P}(0)}\{G_i(\vec{x})\}} \tag{3}$$

where $\mathbf{P}(0)$ is the initial population. Note that the constraint-violation is fair for every constraint since the normalization is based on the initial population which is initialized uniformly over the solution space. If the normalization is based on the current population like [48], bias might be blindly introduced

to constraints since solutions in the current population would not uniformly distribute over the solution space any more.

2) Niche-Count: The niche-count, suggested by Goldberg and Richardson [49], is employed to prevent the population from getting trapped in local optima. The niche-count is defined as follows.

Definition 5 (Niche-Count) [49]: Given $\mathbf{U} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_L\}$ (the union of the parent and offspring populations) and σ (the niche radius), the niche-count of $\vec{x} \in \mathbf{U}$ is defined by

$$\text{nc}(\vec{x}|\mathbf{U}, \sigma) = \sum_{i=1, \vec{x}_i \neq \vec{x}}^{L} \text{sh}(\vec{x}, \vec{x}_i) \tag{4}$$

where the sharing function between two solutions $\vec{x}_1 = (x_{11}, x_{12}, \ldots, x_{1n})$ and $\vec{x}_2 = (x_{21}, x_{22}, \ldots, x_{2n})$, is defined as follows with a niche: a hyper-sphere of radius σ:

$$\text{sh}(\vec{x}_1, \vec{x}_2) = \begin{cases} 1 - \left(\frac{d(\vec{x}_1, \vec{x}_2)}{\sigma}\right) & d(\vec{x}_1, \vec{x}_2) \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $d(\vec{x}_1, \vec{x}_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$ is used in this paper.

3) sDCMOP: This paper aims to use multiobjective techniques to handle the constraint and multimodal difficulties. Therefore, we introduce the constraint-violation $\text{cv}(\vec{x})$ and the niche-count $\text{nc}(\vec{x}|\mathbf{U}, \sigma)$ to a COP in (1) to obtain a CMOP as follows:

$$\begin{aligned} \min \ & (f(\vec{x}), \text{cv}(\vec{x}), \text{nc}(\vec{x}|\mathbf{U}, \sigma)) \\ \text{st:} \ & \vec{g}(\vec{x}) \leq \vec{0}. \end{aligned} \tag{6}$$

The original objective $f(\vec{x})$ in (1) is still one of the objectives in (6). Minimizing the constraint-violation $\text{cv}(\vec{x})$ will push all solutions in the population to feasible regions. Actually, the value of $\text{cv}(\vec{x})$ is zero for any feasible solution. Minimizing the niche-count $\text{nc}(\vec{x}|\mathbf{U}, \sigma)$ will disperse the population with a dispersal distance σ between each other to maintain a certain diversity. Actually, the minimum value of $\text{nc}(\vec{x}|\mathbf{U}, \sigma)$ is also zero, which means that none except for the $\vec{x}$ itself exists in the niche.

The CMOP obtained by (6) is not equivalent to the original COP in (1), although the optimal solution of the COP belongs to the Pareto optimal solution of the CMOP. The CMOEA would face the same difficulty of constraint handling as a CEA does.

To address this issue, we reformulate the CMOP into the DCMOP as follows.

Definition 6 (DCMOP): A dynamic constrained multiobjective problem is defined by

$$\text{CMOP}^{(s)} \begin{cases} \min(f(\vec{x}), \text{cv}(\vec{x}), \text{nc}(\vec{x}|\mathbf{U}, \sigma^{(s)})) \\ \text{st: } \vec{g}(\vec{x}) \leq \vec{\varepsilon}^{(s)} \end{cases} \tag{7}$$

where $\sigma^{(0)} > \sigma^{(1)} > \cdots > \sigma^{(S)} = 0$, $\vec{\varepsilon}^{(s)} = (\varepsilon_1^{(s)}, \varepsilon_2^{(s)}, \ldots, \varepsilon_m^{(s)})$, $s = 0, 1, \ldots, S$, $\vec{\varepsilon}^{(0)} > \vec{\varepsilon}^{(1)} > \cdots > \vec{\varepsilon}^{(S)} = \vec{0}$, S is a given number of environmental changes, $\sigma^{(s)}$ is *the dynamic niche radius*, $\vec{\varepsilon}^{(s)}$ is *the dynamic constraint boundary*, and s is *the environmental state*. An environmental change denotes a reduction of the constraint boundary and the niche radius from state s to state $s + 1$.

The relationship between the original COP in (1) and the DCMOP in (7) is discussed as follows. The feasible set of $\text{CMOP}^{(s-1)}$ includes the feasible set of $\text{CMOP}^{(s)}$ ($s = 1, 2, \ldots, S$). The Pareto optimal set of each $\text{CMOP}^{(s)}$ ($s = 0, 1, \ldots, S$) includes the optimal solution of the original COP. At the final state S, $\vec{\varepsilon}^{(S)} = \vec{0}$, and $\sigma^{(S)} = 0$, we have $\text{cv}(\vec{x}) = 0$ and $\text{nc}(\vec{x}, \mathbf{U}|0) = 0$. Then the final problem $\text{CMOP}^{(S)}$ in the DCMOP in (7) has the following simplified form:

$$\begin{aligned} \min \ & (f(\vec{x}), 0, 0) \\ \text{st:} \ & \vec{g}(\vec{x}) \leq \vec{0}. \end{aligned} \tag{8}$$

The Pareto optimal set for $\text{CMOP}^{(S)}$ is a one-solution-set. The one-solution is exactly the optimal solution of the original COP. In this sense, the $\text{CMOP}^{(S)}$ is called equivalent to the original COP (denoted as $\text{CMOP}^{(S)} \cong \text{COP}$). Therefore, we say that *the DCMOP is equivalent to the COP*, which is denoted by $\text{DCMOP} \cong \text{COP}$.

We set the initial constraint boundary to a large enough value so that all individuals are feasible in the initial population for $\text{CMOP}^{(0)}$. Then a CMOEA can be directly used to solve the $\text{CMOP}^{(0)}$. This is because that UMOEAs are effective to solve UMOPs and a CMOEA with a feasible population is actually equivalent to a UMOEA. A slight reduction of the constraint boundary from $\vec{\varepsilon}^{(s)}$ to $\vec{\varepsilon}^{(s+1)}$ will keep most solutions in the population still feasible so that the performance of the CMOEA will not be significantly impacted. Gradually reducing the constraint boundary from a big enough initial value to zero would be helpful to handle the constraint difficulty.

The population will distribute more broadly when the niche radius σ is increased, and vice versa when σ is decreased. Therefore, gradually reducing the niche radius from an initial value to zero is helpful to handle the multimodal difficulty.

This way, the difficulties of the constraint handling and multimodal handling for a COP will be transferred to a DCMOEA. By doing so, an advantage is that it does not need any extra effort for a DCMOEA to handle the difficulties. Therefore, the conversion does not make the original problem harder. Actually, it provides an alternative way to solve a COP by DCMOEAs. In addition, in this way any existing UMOEA can be treated as a DCMOEA and it can be directly applied to the COP.

To distinguish the concepts for the original COP, we borrow the $\vec{\varepsilon}$ constraint concept [6] for the DCMOP as follows.

Definition 7 ($\vec{\varepsilon}$feasible, $\vec{\varepsilon}$infeasible): A solution $\vec{x} \in \mathbf{X}$ is said to be $\vec{\varepsilon}$feasible regarding the dynamic constraint boundary $\vec{\varepsilon} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m)$, if $\vec{g}(\vec{x}) \leq \vec{\varepsilon}$; otherwise $\vec{x}$ is said to be $\vec{\varepsilon}$infeasible.

Pareto domination is defined only on the feasible set and it does not apply to infeasible solutions. For this reason, an $\vec{\varepsilon}$-constrained Pareto domination is defined on the whole solution space as follows.

Definition 8 ($\vec{\varepsilon}$-Constrained Pareto Domination): $\forall \vec{a}, \vec{b} \in \mathbf{X}$:

1) if both are $\vec{\varepsilon}$feasible, then the Pareto domination is used;
2) if one is $\vec{\varepsilon}$feasible and the other $\vec{\varepsilon}$infeasible, then the $\vec{\varepsilon}$feasible one wins;
3) if both are $\vec{\varepsilon}$infeasible, then the one with a smaller constraint-violation cv wins.

Note that Takahama and Sakai [6], [50] also proposed a ε-constrained method. However, the method converts a COP to an equivalent dynamic COP rather than an equivalent DCMOP.

C. When to Change the Environment

There are two important issues for the use of a DCMOEA for solving COPs. One issue is that solutions in a population should be $\vec{\varepsilon}$feasible as many as possible for the effectiveness of a UMOEA. The other issue is that the DCMOEA should reach the final problem CMOP$^{(S)}$($\cong$COP) as quickly as possible since the DCMOEA in this paper is to solve the COP ($\cong$CMOP$^{(S)}$) rather than the whole DCMOP. To address these two issues, we change the environment as follows.

1) Set a large enough value for the initial constraint boundary $\vec{\varepsilon}^{(0)}$ so that the initial population is $\vec{\varepsilon}$feasible and set an initial niche $\sigma^{(0)}$ at state $s = 0$ (CMOP$^{(0)}$) at generation $t = 0$.
2) Reduce the constraint boundary slightly from $\vec{\varepsilon}^{(0)}$ to $\vec{\varepsilon}^{(1)}$ and reduce the niche radius slightly from $\sigma^{(0)}$ to $\sigma^{(1)}$ (i.e., change the problem from CMOP$^{(0)}$ to CMOP$^{(1)}$ at state $s = 1$).
3) Use solutions obtained from CMOP$^{(0)}$ as initial solutions for CMOP$^{(1)}$, then evolve the solutions till they all become $\vec{\varepsilon}$feasible.
4) Repeat the reducing-evolving process till s reaches the final state $s = S$.

Note that although an algorithm is required to evolve until the population becomes $\vec{\varepsilon}$feasible for the current CMOP, it in fact always searches for the global optimum of the original problem as the global optimum is always in the Pareto-optimal set of every CMOP$^{(s)}$.

D. How to Change the Environment

Inspired by the simulated annealing algorithm [51], where the acceptance probability and the annealing program usually use an exponential function, the reduction of the constraint boundary $\vec{\varepsilon}$ and the niche radius σ also employ an exponential function for each change as follows:

$$\varepsilon_i^{(s)} = A_i e^{-\left(\frac{s}{B_i}\right)^{\text{cp}}} - \delta, i = 1, 2, \ldots, m \tag{9}$$

$$\sigma^{(s)} = Ce^{-(\frac{s}{D})^{\text{cp}}} - \delta \tag{10}$$

where δ is a positive close-to-zero number for the precision requirement ($\delta = 1e-8$ in this paper) and cp is a control parameter of the decreasing level. A_i, B_i ($i = 1, 2, \ldots, m$), C and D are constant parameters to be determined.

Let us first determine the values of A_i and B_i in (9). In order to satisfy an $\vec{\varepsilon}$feasible initial parent population $\mathbf{P}(0)$, the maximum violation of a constraint in $\mathbf{P}(0)$ is selected as the initial constraint boundary $\varepsilon_i^{(0)}$. Therefore, we have the initial constraint boundary $\varepsilon_i^{(0)} = \max_{\vec{x}\in\mathbf{P}(0)}\{G_i(\vec{x})\}$, $i = 1, 2, \ldots, m$ at $s = 0$ [see (2) for $G_i(\vec{x})$], and the final constraint boundary is $\vec{\varepsilon}^{(S)} = \vec{0}$ at $s = S$. From the initial and final states of (9), we obtain a group of two equations

$$\begin{cases} \varepsilon_i^{(0)} = A_i - \delta & s = 0 \\ 0 \quad = A_i e^{-\left(\frac{S}{B_i}\right)^{\text{cp}}} - \delta & s = S \end{cases} \tag{11}$$

where $i = 1, 2, \ldots, m$. We can obtain A_i and B_i from (11) by

$$\begin{cases} A_i = \varepsilon_i^{(0)} + \delta \\ B_i = \dfrac{S}{\sqrt[\text{cp}]{ln\left(\frac{\varepsilon_i^{(0)}+\delta}{\delta}\right)}}. \end{cases} \tag{12}$$

Then we determine the values of C and D in (10). Suppose there are L solutions in the union of the parent and offspring populations $\mathbf{U}$, and the average space, which a solution takes, is the initial niche size, i.e., $[\prod_{i=1}^{n}(u_i - l_i)/L]$. Therefore, we have the initial niche radius $\sigma^{(0)} = (1/2)\sqrt[n]{[(2n\prod_{i=1}^{n}(u_i - l_i))/L\pi]}$ at $s = 0$, and the final niche radius is $\sigma^{(S)} = 0$ at $s = S$. Similarly, we obtain the values of C and D by

$$\begin{cases} C = \sigma^{(0)} + \delta \\ D = \dfrac{S}{\sqrt[\text{cp}]{ln\left(\frac{\sigma^{(0)}+\delta}{\delta}\right)}}. \end{cases} \tag{13}$$

Algorithm 1 Algorithm Framework

Step 1: Initialization
- *1.1* Initialize a parent population and set the global generation counter $t = 0$.
- *1.2* Initialize the niche radius $\sigma = \sigma^{(0)}$ and the constraint boundary $\vec{\varepsilon} = \vec{\varepsilon}^{(0)}$, and set the problem state $s = 0$ and the local generation counter $t_s = 0$.

Step 2: **IF** the population is $\vec{\varepsilon}$feasible **THEN** reduce $\sigma = \sigma^{(s+1)}$ and $\vec{\varepsilon} = \vec{\varepsilon}^{(s+1)}$, update population $\vec{\varepsilon}$-feasibility and other components, $s = s + 1$, $t_s = 0$.
ELSE $t_s = t_s + 1$.

Step 3: Generate the offspring population and select the next parent population.

Step 4: $t = t + 1$.

Step 5: **IF** s reaches the final state S or t reaches *AbortingT* **THEN** go to *Step 6*, **ELSE** go back to *Step 2*.

Step 6: Output results.

IV. Algorithm Implementation

A. Algorithm Framework

The framework of a DCMOEA for DCMOPs is shown in Algorithm 1.

Note that without steps 1.2 and 2, the DCMOEA framework is the same as a common CMOEA. Also, the CMOEA can be obtained by replacing the Pareto domination in the UMOEA with the $\vec{\varepsilon}$constrained Pareto domination (Definition 8). If a COP has no feasible solution or an algorithm could not find an $\vec{\varepsilon}$feasible population at a certain state, then the algorithm will iterate infinitely at this state. Therefore, a large *AbortingT* is used to terminate the run in this case.

Algorithm 2 DE/rand/1/bin Operator

Input: Parents $\{\vec{p}_i, \vec{p}_a, \vec{p}_b, \vec{p}_c\}$
Output: Offspring $\vec{q}_i$
Step 1: Affine Mutation on $\vec{p}_a, \vec{p}_b, \vec{p}_c$:
$\vec{v}_i = \vec{p}_a + F(\vec{p}_b - \vec{p}_c)$.
Step 2: Crossover on $\vec{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ and $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{in})$:
j_{rnd}= *rndInt*$(1, n)$, r_{rnd}= *rndReal*$(0, 1)$.
$$q_{ij} = \begin{cases} v_{ij} \text{ if } r_{rnd} < CR \text{ or } j = j_{rnd} \\ p_{ij} \text{ if } r_{rnd} \geq CR \text{ and } j \neq j_{rnd} \end{cases}$$
$j = 1, 2, \ldots, n$
Step 3: Uniform mutation on $\vec{q}_i = (q_{i1}, q_{i2}, \ldots, q_{in})$:
Change $q_{ij} = rndReal(l_j, u_j)$ with a probability p_m for $j = 1, 2, \ldots, n$.
Step 4: Output $\vec{q}_i$.

B. Instances of DCMOEA

The instantiation of the DCMOEA (Algorithm 1) requires:
1) an initial parent population in step 1.1;
2) the generation of the offspring population and the selection of the new parent population in step 3;
3) and an update operator in step 2.

Three state-of-the-art MOEAs: 1) one Pareto ranking-based algorithm NSGA-II [33]; 2) one decomposition-based algorithm MOEA/D M2M [52]; and 3) one hypervolume-based algorithm HypE [53], are selected to instantiate the DCMOEA, respectively. The three algorithms are based on Pareto domination for problems without constraints. For DCMOPs in this paper, the Pareto domination is replaced with $\vec{\varepsilon}$constrained Pareto domination (see Definition 8).

Note that the third version of NSGA: NSGA-III [34] was designed for problems with many objectives (more than three objectives), the DCMOPs in this paper have only three objectives. Therefore, we adopts NSGA-II instead of NSGA-III in this paper.

For a fair performance comparison, the DE/rand/1/bin operator [54] is used for all the three algorithms.

If the vector $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ obtained in step 1 in Algorithm 2 is invalid, we will truncate invalid components of $\vec{v}_i$ as follows:

$$v_{ij} = \begin{cases} l_j & \text{if } v_{ij} < l_j \\ u_j & \text{if } v_{ij} > u_j, j = 1, 2, \ldots, n. \end{cases} \quad (14)$$

For clarity, dynamic constrained versions of the above three algorithms are denoted as DCNSGAII-DE, DCM2M-DE, and DCHypE-DE, respectively. In the three algorithms, the initialization of the parent population (step 1.1 of Algorithm 1), the generation of the offspring population and the selection of the next parent population (step 3 of Algorithm 1) are referred directly to NSGA-II [33], MOEA/D-M2M [52], and HypE [53], respectively, except that the generation of new solutions are replaced by Algorithm 2.

V. Experimental Results and Discussions

A. Benchmark Problems

Two sets of benchmark problems were used to evaluate the performance of the three instantiations of DCMOEAs. They are $setG = \{g01, g02, \ldots, g024\}$ [55] and $setC = \{c01 - D10, c02 - D10, \ldots, c18 - D10\} \bigcup \{c01 - D30, c02 - D30, \ldots, c18 - D30\}$ (18 problems each with 10-D and 30-D) [56] proposed by Suganthan's group for the competitions of the IEEE CEC2006 and the IEEE CEC2010, respectively.

B. Parameters Settings of DCMOEAs

Parameter settings of DCMOEAs are as follows.
1) The parent population size and the offspring population size: $N = 100$.
2) The settings of the DE operator in Algorithm 2: Scaling factor F = rndReal$(0, 1)$, crossover probability CR $= 0.9$, and mutation probability $p_m = 0.01$. Note that F is randomly selected in this paper due to the tradeoff between simplicity and adaptability.
3) The two constant parameters in (9) and (10): $\delta = 1e-8$, cp $= 2$.
4) The number of runs: 25.
5) Aborting evaluations FEs $= 1\,000\,000$.
6) The number of the environmental changes: $S = 2400$ for problems in *setG*, $S = 2000$ and $S = 6000$ for problems in *setC* with 10-D and 30-D, respectively.

To test the statistical significance, the Friedman's test is implemented to sort the performance of all compared algorithms and the Wilcoxon's rank sum test at significance level $\alpha = 0.05$ and $\alpha = 0.1$ is also implemented for all peer algorithms in this paper.

C. Experimental Results on setG

For the test suite of *setG*, the three DCMOEAs (DCNSGAII-DE, DCM2M-DE, and DCHypE-DE) are compared with five state-of-the-art methods: 1) CMODE [45]; 2) εADE [6]; 3) DPDE [12]; 4) ECHT-ARMOR-DE1 [9]; and 5) ECHT-ARMOR-DE2 [9] with the ECHT technique [57]. The three DCMOEAs proposed in this paper are similar to CMODE in terms of the use of multiobjective techniques, and they are similar to εADE in terms of the use of dynamic techniques. The best, average, worst, and standard deviation of the objective function values obtained over 25 runs are used to compare the performance.

Table I shows the problems where the algorithms get trapped in local optima over 25 runs, where the value in the parenthesis denotes the number of runs for an algorithm converging to local optima. From the results, all the three DCMOEAs successfully locate the global optimum for each run on all the problems. However, the other algorithms all get trapped in local optima: CMODE [45] on $g02$, $g21$, and $g23$, εADE [6] on $g17$, $g18$, and $g23$, DPDE [12] on $g02$ and $g23$, and ECHT-ARMOR-DE1-2 [9] on $g02$.

The Friedmans test in Table II shows that DCNSGAII-DE ranks the first, DCM2M-DE ranks the fifth and DCHypE-DE ranks the last over all the peer algorithms. Statistical results of the Wilcoxon-test are shown in Table III where R^+, R^- denote the sum of ranks. $R^+ > R^-$ means that the proposed algorithm is better than the compared algorithm and vice versa. "No" represents no significant difference

TABLE I
PROBLEMS WHERE CMODE [45], εADE [6], DPDE [12], AND ECHT-ARMOR-DE1-2 [9] GET TRAPPED IN LOCAL OPTIMA ON *setG* OVER 25 RUNS

Algorithms	DCMOEAs	CMODE	εADE	DPDE	ECHT-ARMOR-DE1-2
Problems	none	g02(1), g21(7), g23(5)	g17(3), g18(4), g23(4)	g02(-), g23(-)	g02(-)

The sign '-' means that the corresponding results of DPDE and ECHT-ARMOR-DE1-2 are not available in [12] and [9].

TABLE II
RANKS OF DCNSGAII-DE, DCM2M-DE, DCHYPE-DE, CMODE [45], εADE [6], DPDE [12], ECHT-ARMOR-DE1, AND ECHT-ARMOR-DE2 [9] BY THE FRIEDMAN'S TEST ON *setG*

Algorithm	Ranking
DCNSGAII-DE	4.18
ECHT-AROMOR-DE2	4.32
ECHT-AROMOR-DE1	4.36
CMODE	4.43
DCM2M-DE	4.48
DPDE	4.52
εADE	4.84
DCHypE-DE	4.86

TABLE III
RESULTS OF THE MULTIPLE-PROBLEM WILCOXON'S TEST OF DCNSGAII-DE, DCM2M-DE, AND DCHYPE-DE AGAINST CMODE [45], εADE [6], DPDE [12], ECHT-ARMOR-DE1, AND ECHT-ARMOR-DE2 [9], RESPECTIVELY, ON *setG*

Algorithm	R^+	R^-	α=0.05	α=0.1
DCNSGAII-DE vs CMODE	10	5	No	No
DCNSGAII-DE vs εADE	14	7	No	No
DCNSGAII-DE vs DPDE	8	7	No	No
DCNSGAII-DE vs ECHT-ARMOR-DE1	6	4	No	No
DCNSGAII-DE vs ECHT-ARMOR-DE2	5	5	No	No
Algorithm	R^+	R^-	α=0.05	α=0.1
DCM2M-DE vs CMODE	10	5	No	No
DCM2M-DE vs εADE	14	7	No	No
DCM2M-DE vs DPDE	8	7	No	No
DCM2M-DE vs ECHT-ARMOR-DE1	7	8	No	No
DCM2M-DE vs ECHT-ARMOR-DE2	7	8	No	No
Algorithm	R^+	R^-	α=0.05	α=0.1
DCHypE-DE vs CMODE	19	17	No	No
DCHypE-DE vs εADE	17	11	No	No
DCHypE-DE vs DPDE	17	19	No	No
DCHypE-DE vs ECHT-ARMOR-DE1	10	18	No	No
DCHypE-DE vs ECHT-ARMOR-DE2	9	19	No	No

R^+, R^- represent the sum of ranks, $R^+ > R^-$ means that the algorithm of this paper is better than the compared algorithm and vice versa, "No" means no significant difference between two compared algorithms.

between two compared algorithms. We can see that there is no significant difference among all the compared algorithms. The detailed results of all the algorithms on problems in *setG* are provided in the Appendix due to the space limitation.

From the results in this section, the three instantiations (DCNSGAII-DE, DCM2M-DE, and DCHypE-DE) are competitive to the state-of-the-art algorithms referred in this paper on the test suite of *setG*. An interesting observation is that the three instantiations do not get trapped in local optima on all the problems in *setG* while CMODE, εADE, DPDE, ECHT-ARMOR-DE1-2 all get trapped in local optima on some problems.

TABLE IV
RANKS OF DCNSGAII-DE, DCM2M-DE, DCHYPE-DE, CMODE [45], εADE [6], AIS [58], AND ECHT-ARMOR-DE [9] BY THE FRIEDMAN'S TEST ON PROBLEMS $c01$–$c18$ WITH 10-D

Algorithm	Ranking
ECHT-ARMOR-DE	2.86
DCM2M-DE	3.64
DCNSGAII-DE	3.89
AIS	4.06
DCHypE-DE	4.17
εADE	4.56
CMODE	4.83

TABLE V
RESULTS OF THE MULTIPLE-PROBLEM WILCOXON'S TEST OF DCNSGAII-DE, DCM2M-DE, AND DCHYPE-DE AGAINST CMODE [45], εADE [6], AIS [58], AND ECHT-ARMOR-DE [9], RESPECTIVELY, ON $c01$–$c18$ WITH 10-D

Algorithm	R^+	R^-	α=0.05	α=0.1
DCNSGAII-DE vs CMODE	135	36	Better	Better
DCNSGAII-DE vs εADE	126	45	No	Better
DCNSGAII-DE vs AIS	99	72	No	No
DCNSGAII-DE vs ECHT-ARMOR-DE	57	79	No	No
Algorithm	R^+	R^-	α=0.05	α=0.1
DCM2M-DE vs CMODE	120	51	No	No
DCM2M-DE vs εADE	116	37	No	Better
DCM2M-DE vs AIS	96	75	No	No
DCM2M-DE vs ECHT-ARMOR-DE	58	78	No	No
Algorithm	R^+	R^-	α=0.05	α=0.1
DCHypE-DE vs CMODE	133	38	Better	Better
DCHypE-DE vs εADE	123	48	No	No
DCHypE-DE vs AIS	94	77	No	No
DCHypE-DE vs ECHT-ARMOR-DE	58	78	No	No

R^+, R^- represent the sum of ranks, $R^+ > R^-$ means that the algorithm of this paper is better than the compared algorithm and vice versa, "No" means no significant difference between two compared algorithms.

D. Experimental Results on setC

In this section, we compare DCNSGAII-DE, DCM2M-DE, and DCHypE-DE against four state-of-the-art methods: two similar methods (CMODE [45] and εADE [6]) to the DCMOEA, and two recently proposed algorithms (AIS [58] and ECHT-ARMOR-DE [9]) on the test suite of *setC*.

Tables IV and V show the statistical test results for problems with $D = 10$ based on the Friedman's test and the multiple-problem Wilcoxon's test, respectively. Compared with CMODE, εADE, and AIS, the three instantiations achieve higher R^+ values than R^- values as shown in Table V, which means that the three instantiations all perform better than CMODE, εADE, and AIS. DCNSGAII-DE, DCM2M-DE, and DCHypE-DE achieve lower R^+ values than R^- values with the Wilcoxon's test in comparison with ECHT-ARMOR-DE. However, the Wilcoxon's test shows that there is no significant difference between them, i.e., DCNSGAII-DE, DCM2M-DE, and DCHypE-DE are competitive to ECHT-ARMOR-DE.

Tables VI and VII show the statistical test results for problems with $D = 30$ based on the Friedman's test and the multiple-problem Wilcoxon's test, respectively. Table VI shows that DCNSGAII-DE, DCM2M-DE and DCHypE-DE achieve the best ranking. In addition, from Table VII, it can be seen that DCNSGAII-DE, DCM2M-DE, and DCHypE-DE obtain higher R^+ values than R^- values in all the cases. Furthermore, DCNSGAII-DE, DCM2M-DE, and DCHypE-DE all significantly outperform CMODE and εADE. Due to

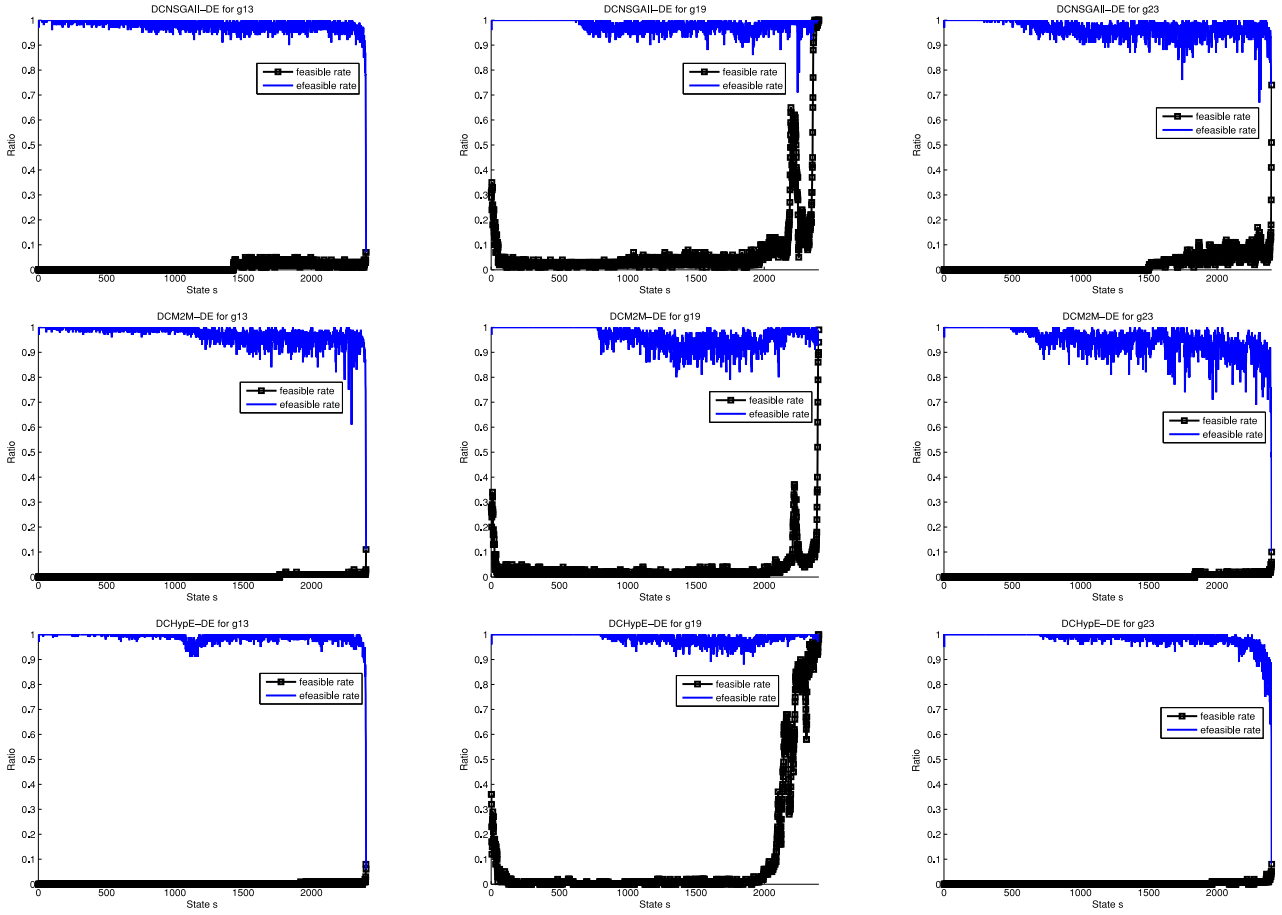


Fig. 1. Progress of the $\vec{\varepsilon}$feasible ratio and the feasible ratio over environmental changes. The problem with (a) equality constraints, (b) inequality constraints, and (c) equality and inequality constraints.

TABLE VI
RANKS OF DCNSGAII-DE, DCM2M-DE, DCHYPE-DE, CMODE [45], εADE [6], AIS [58], AND ECHT-ARMOR-DE [9] BY THE FRIEDMAN'S TEST ON PROBLEMS $c01$–$c18$ WITH 30-D

Algorithm	Ranking
DCHypE-DE	3.08
DCNSGAII-DE	3.36
DCM2M-DE	3.67
ECHT-ARMOR-DE	3.86
AIS	3.94
CMODE	4.72
εADE	5.36

TABLE VII
RESULTS OF THE MULTIPLE-PROBLEM WILCOXON'S TEST OF DCNSGAII-DE, DCM2M-DE, AND DCHYPE-DE AGAINST CMODE [45], εADE [6], AIS [58] AND ECHT-ARMOR-DE [9], RESPECTIVELY, ON PROBLEMS $c01$–$c18$ WITH 30-D

Algorithm	R^+	R^-	α=0.05	α=0.1
DCNSGAII-DE vs CMODE	134	37	Better	Better
DCNSGAII-DE vs εADE	126	27	Better	Better
DCNSGAII-DE vs AIS	105	66	No	No
DCNSGAII-DE vs ECHT-ARMOR-DE	94	59	No	No
Algorithm	R^+	R^-	α=0.05	α=0.1
DCM2M-DE vs CMODE	130	41	Better	Better
DCM2M-DE vs εADE	131	40	Better	Better
DCM2M-DE vs AIS	99	72	No	No
DCM2M-DE vs ECHT-ARMOR-DE	87	66	No	No
Algorithm	R^+	R^-	α=0.05	α=0.1
DCHypE-DE vs CMODE	133	38	Better	Better
DCHypE-DE vs εADE	139	32	Better	Better
DCHypE-DE vs AIS	105	66	No	No
DCHypE-DE vs ECHT-ARMOR-DE	94	59	No	No

R^+, R^- represent the sum of ranks, $R^+ > R^-$ means that the algorithm of this paper is better than the compared algorithm and vice versa, "No" means no significant difference between two compared algorithms.

the space limitation, detailed results of all the algorithms on problems in *setC* are provided in the Appendix.

We conclude that DCNSGAII-DE, DCM2M-DE, DCHypE-DE are better than or competitive to the state-of-the-art algorithms tested in this paper on problems in *setC*, especially in the case of higher dimensions.

E. Investigation of the Change of $\vec{\varepsilon}$feasibility/feasibility

To investigate the behaviors of a DCMOEA when the problem changes, we define *$\vec{\varepsilon}$feasible ratio* and *feasible ratio* at state s as the ratio of $\vec{\varepsilon}$feasible and feasible solutions to all the solutions in the initial population after the environment changes from state $s-1$ to s, respectively. Due to the space limitation, the ratios of the feasibility/$\vec{\varepsilon}$feasibility of the population are provided only for three typical problems: $g13$, $g19$, and $g23$. $g13$ has only equality constraints, $g19$ has only inequality constraints, and $g23$ has both inequality and equality constraints.

Fig. 1 shows the changes of the two ratios for DCNSGAII-DE, DCM2M-DE and DCHypE-DE, where the

left, middle, and right columns are results for $g13$, $g19$, and $g23$, respectively, and the top, middle, and bottom rows are results for DCNSGAII-DE, DCM2M-DE, and DCHypE-DE, respectively.

The results show that the $\vec{\varepsilon}$feasible ratios of the population over the environmental changes for the three instantiations are almost always above 90% on all the three problems. This indicates that the three instantiations search mainly in the $\vec{\varepsilon}$feasible set. Together with the results obtained in Sections V-C and V-D, we can conclude that a small percentage of $\vec{\varepsilon}$infeasible solutions in the population does not impact much on the effectiveness of the three algorithms.

The feasible ratios of the population in the three instantiations are almost always 0% over the state changes for $g13$ and $g23$. This is because the ratios between the feasible set and the whole solution space are almost 0% for these two problems. For the problem $g19$, the ratio of the feasible area over the whole solution space is 33.4761%. However, the feasible ratios of the population for this problem are also almost always very low (about 3%). It seems that the three instantiations search in a small part of the feasible set. This might be a limitation of the DCMOEA.

F. Effect of the Niche-Count on the Global Search

The objective of the niche-count $\text{nc}(\vec{x}|\mathbf{U}, \sigma^{(s)})$ in (7) is assumed to be helpful for the global search. To verify this assumption, a comparison experiment is conducted on the DCMOP with and without the niche-count objective. Without $\text{nc}(\vec{x}|\mathbf{U}, \sigma^{(s)})$, a three-objective DCMOP is degenerated to a bi-objective DCMOP as follows:

$$\text{CMOP}^{(s)} \quad \begin{cases} \min \ \vec{y} = (f(\vec{x}), \text{cv}(\vec{x})) \\ \text{st: } \vec{g}(\vec{x}) \le \vec{\varepsilon}^{(s)} \end{cases} \quad (15)$$

$s = 0, 1, \ldots, S.$

Like the three-objective DCMOP, the bi-objective DCMOP is also equivalent to the original COP, thus the three instantiations of the DCMOEA (DCNSGAII-DE, DCM2M-DE, and DCHypE-DE) can be applied to solve the bi-objective DCMOP.

The problem $g02$ in *setG* was selected for the experimental study since it has a huge number of local optima. The number of runs was set to 100. Table VIII shows the number of runs where DCNSGAII-DE, DCM2M-DE, and DCHypE-DE get trapped in local optima, with and without the niche-count objective.

From Table VIII we can see that without $\text{nc}(\vec{x}|\mathbf{U}, \sigma^{(s)})$, DCNSGAII-DE, DCM2M-DE, and DCHypE-DE get trapped in local optima for 11, 9, 26 times, respectively, over 100 runs. On the contrary, they all successfully find the global optimum over all runs with the niche-count objective. Therefore, we can conclude that the niche-count objective is helpful for the global search.

G. Effect of Varying the Environmental Change Level

The parameter cp in the exponential functions in (9) and (10), is to control the environmental change level. Different values of cp have different impact on the shape of the exponential functions. Fig. 2 plots the curves of the exponential functions with cp $= 1, 2, 3$ for $S = 2000$, where σ_0/ε_0 was set to 1.0.

TABLE VIII
NUMBER OF RUNS, WHERE DCNSGAII-DE, DCM2M-DE, AND DCHYPE-DE WITH AND WITHOUT $\text{nc}(\vec{x}|\mathbf{U}, \sigma^{(s)})$, GET TRAPPED ON $g02$ OVER 100 RUNS

Algorithm	without $nc(\vec{x}\|\mathbf{U}, \sigma^{(s)})$	with $nc(\vec{x}\|\mathbf{U}, \sigma^{(s)})$
DCNSGAII-DE	11	0
DCM2M-DE	9	0
DCHypE-DE	26	0

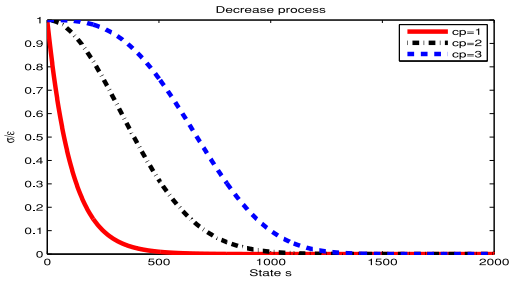


Fig. 2. Changes of the ε/σ function with different values of cp.

From Fig. 2, the objective values of all the three curves decrease quickly at the early stage and very slow at the late stage. This indicates that the DCMOEA focuses on exploration for feasible solutions in the early stage, while it will focus on exploitation for high quality solutions in the late stage.

To test the sensitivity of the DCMOEA to the value of cp, we run DCNSGAII-DE 100 times on the problem $g02$ with cp $= 1, 2, 3$. The results show that DCNSGAII-DE gets trapped in local optima for 36 times over 100 runs with cp $= 1$ while it does not get trapped with cp $= 2$ and cp $= 3$. DCNSGAII-DE obtains better results with cp $= 2$ than that with cp $= 3$, therefore cp $= 2$ is used in this paper.

Note that other functions, e.g., polynomial, linear, or logarithmic ones, could also be used instead of the exponential one. We will study the effect of choosing different functions in our future work.

H. Aborting the DCMOEA

If a problem has no feasible solution or an algorithm fails to find feasible solutions, the algorithm will iterate infinitely at a certain state. In this abnormal case, the DCMOEA should stop when the number of function evaluations reaches the maximum function evaluations (1 000 000 evaluations in this paper). The problem $g20$ in *setG* has no feasible solution, the DCMOEAs abort after 1 000 000 evaluations. The DCMOEAs also fail to find feasible solutions on the problem $g22$ in *setG*, $c11-10D$, $c12-10D$, $c11-30D$, $c12-30D$, $c17-30D$ in *setC* for some runs. In those cases, the average evaluations used by the DCMOEAs are much larger than the given evaluations.

I. Computational Complexity of the DCMOEAs

It takes at least one generation for a DCMOEA to obtain an $\vec{\varepsilon}$feasible population when the problem state changes, i.e., the

number of local generations $T_s \geq 1$. Therefore, the total number of global generations T will be greater than S. However, the experimental results in this paper show that T_s is one at almost every state s, i.e., T is very close to S ($T \approx S$).

Considering the running time of a DCMOEA for one generation, the complexity of the algorithm is $O(N^2)$ where N is the population size. The extra computational cost is the calculation of violation-constraints and niche-count with the time complexity of $O(mN)$ and $O(nN^2)$, respectively, where m and n are the number of constraints and the number of dimensions.

The overall computational complexity is $O(N^2+nN^2+mN)$ for DCNSGAII-DE, DCM2M-DE, and DCHypE-DE. Notably, the experimental results show that DCNSGAII-DE runs the fastest, where the computational time of DCM2M-DE is about 1.5 times of DCNSGAII-DE and the computational time of DCHypE-DE is about 4 times of DCNSGAII-DE.

VI. Conclusion

This paper proposes a method for converting a COP into an equivalent DCMOP with three objectives: 1) the original objective; 2) a constraint-violation; and 3) a niche-count. In the DCMOP, the constraint boundary and the niche radius gradually decrease as the environmental state increases. Gradually reducing the constraint boundary helps to handle the constraint difficulty, and gradually reducing the niche size helps to address the multimodal difficulty. This paper also proposes a general framework for the DCMOEA. In the framework, any existing UMOEA can be directly applied to the COP.

Three instantiations of the DCMOEA are implemented by the dynamic versions of three existing state-of-the-art MOEAs. Experimental results show that they are better than or competitive to the state-of-the-art algorithms tested in this paper.

We would like to pursue several topics in the future.

1) Each constraint violation can be treated as an objective individually. A COP then can be equivalently transformed to a dynamic constrained many-objective optimization problem. Recent studies on many-objective optimization problems can be employed to solve the constructed problem.
2) A CMOP can be equivalently transformed to a DCMOP. Then this methodology can be employed to solve the CMOP.
3) A multimodal optimization problem can be equivalently transformed to a dynamic MOP. Then this methodology can be employed to solve the multimodal optimization problem.
4) The DCMOEA might not perform well on COPs having the global optimum inside the constraint boundary. In this case, more feasible solutions should be preserved in the population. Further studies are needed.

Appendix

The detailed results of Tables II, IV, and VI in this paper are provided in the supplementary file.

References

[1] T. Ray and K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 386–396, Aug. 2003.

[2] Y. Wang, B.-C. Wang, H.-X. Li, and G. G. Yen, "Incorporating objective function information into the feasibility rule for constrained evolutionary optimization," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2938–2952, Dec. 2016.

[3] M. de Castro Rodrigues, B. S. L. P. de Lima, and S. Guimarães, "Balanced ranking method for constrained optimization problems using evolutionary algorithms," *Inf. Sci.*, vol. 327, pp. 71–90, Jan. 2016.

[4] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 284–294, Sep. 2000.

[5] R. Mallipeddi, P. N. Suganthan, and B. Y. Qu, "Diversity enhanced adaptive evolutionary programming for solving single objective constrained problems," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 2106–2113.

[6] T. Takahama and S. Sakai, "Efficient constrained optimization by the ϵ constrained adaptive differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–8.

[7] J. Xiao, J. Xu, Z. Shao, C. Jiang, and L. Pan, "A genetic algorithm for solving multi-constrained function optimization problems based on KS function," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 4497–4501.

[8] K. Deb and R. Datta, "A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–8.

[9] W. Gong, Z. Cai, and D. Liang, "Adaptive ranking mutation operator based differential evolution for constrained optimization," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 716–727, Apr. 2015.

[10] N. M. Hamza, D. L. Essam, and R. A. Sarker, "Constraint consensus mutation-based differential evolution for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 447–459, Jun. 2016.

[11] E. Mezura-Montes and C. A. C. Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 1–17, Feb. 2005.

[12] W.-F. Gao, G. G. Yen, and S.-Y. Liu, "A dual-population differential evolution with coevolution for constrained optimization," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1108–1121, May 2015.

[13] J. Sun, J. M. Garibaldi, Y. Zhang, and A. Al-Shawabkeh, "A multi-cycled sequential memetic computing approach for constrained optimisation," *Inf. Sci.*, vols. 340–341, pp. 175–190, May 2016.

[14] A. Maesani, G. Iacca, and D. Floreano, "Memetic viability evolution for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 125–144, Feb. 2016.

[15] K. Yu, X. Wang, and Z. Wang, "Constrained optimization based on improved teaching-learning-based optimization algorithm," *Inf. Sci.*, vols. 352–353, pp. 61–78, Jul. 2016.

[16] F. Jimenez, A. F. Gomez-Skarmeta, G. Sanchez, and K. Deb. "An evolutionary algorithm for constrained multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Honolulu, HI, USA, 2002, pp. 1133–1138.

[17] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multiobjective evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 514–525, Jun. 2009.

[18] H. K. Singh, T. Ray, and W. Smith, "C-PSA: Constrained Pareto simulated annealing for constrained multi-objective optimization," *Inf. Sci.*, vol. 180, no. 13, pp. 2499–2513, 2010.

[19] B. Y. Qu and P. N. Suganthan, "Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods," *Eng. Optim.*, vol. 43, no. 4, pp. 403–416, 2011.

[20] H.-L. Liu and D. Wang, "A constrained multiobjective evolutionary algorithm based decomposition and temporary register," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 3058–3063.

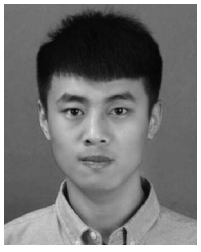
[21] M. Miyakawa, K. Takadama, and H. Sato, "Controlling selection areas of useful infeasible solutions for directed mating in evolutionary constrained multi-objective optimization," *Ann. Math. Artif. Intell.*, vol. 76, nos. 1–2, pp. 25–46, 2016.

[22] K. Li, K. Deb, Q. Zhang, and Q. Zhang, "Efficient nondomination level update method for steady-state evolutionary multiobjective optimization," *IEEE Trans. Cybern.*, no. 99, pp. 1–12, 2016, doi: 10.1109/TCYB.2016.262100.

[23] S. B. Gee, K. C. Tan, and C. Alippi, "Solving multiobjective optimization problems in unknown dynamic environments: An inverse modeling approach," *IEEE Trans. Cybern.*, no. 99 pp. 1–12, 2016, doi: 10.1109/TCYB.2016.2602561.
[24] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, no. 99 pp. 1–14, 2016, doi: 10.1109/TCYB.2016.2600577.
[25] X. Cai, Z. Yang, Z. Fan, and Q. Zhang, "Decomposition-based-sorting and angle-based-selection for evolutionary multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, no. 99 pp. 1–14, 2016, doi: 10.1109/TCYB.2016.2586191.
[26] A. Zhou and Q. Zhang, "Are all the subproblems equally important? Resource allocation in decomposition-based multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 52–64, Feb. 2016.
[27] K. Deb and M. Abouhawwash, "An optimality theory-based proximity measure for set-based multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 515–528, Aug. 2016.
[28] Z. He and G. G. Yen, "Many-objective evolutionary algorithm: Objective space reduction and diversity improvement," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 145–160, Feb. 2016.
[29] S. Zeng *et al.*, "Dynamic constrained multi-objective model for solving constrained optimization problem," in *Proc. IEEE Congr. Evol. Comput.*, New Orleans, LA, USA, 2011, pp. 2041–2046.
[30] L. Jia *et al.*, "Dynamic multi-objective differential evolution for solving constrained optimization problem," in *Proc. IEEE Congr. Evol. Comput.*, New Orleans, LA, USA, 2011, pp. 2649–2654.
[31] W. Dong *et al.*, "Linear sparse arrays designed by dynamic constrained multi-objective evolutionary algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 3067–3072.
[32] X. Li, S. Zeng, S. Qin, and K. Liu, "Constrained optimization problem solved by dynamic constrained NSGA-III multiobjective optimizational techniques," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, 2015, pp. 2923–2928.
[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[34] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.
[35] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, "Infeasibility driven evolutionary algorithm for constrained optimization," in *Constraint-Handling in Evolutionary Optimization*. Heidelberg, Germany: Springer, 2009, pp. 145–165.
[36] H. K. Singh, M. Asafuddoula, and T. Ray, "Solving problems with a mix of hard and soft constraints using modified infeasibility driven evolutionary algorithm (IDEA-M)," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 983–990.
[37] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 9, no. 4, pp. 424–435, Aug. 2005.
[38] K. Deb, S. Lele, and R. Datta, "A hybrid evolutionary multi-objective and SQP based procedure for constrained optimization," in *Proc. Int. Symp. Intell. Comput. Appl.*, Wuhan, China, 2007, pp. 36–45.
[39] R. Datta and K. Deb, "Individual penalty based constraint handling using a hybrid bi-objective and penalty function approach," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 2720–2727.
[40] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm Evol. Comput.*, vol. 1, no. 4, pp. 173–194, 2011.
[41] C. A. C. Coello, "Constraint-handling techniques used with evolutionary algorithms," in *Proc. Companion Publ. Gen. Evol. Comput. Conf. (GECCO)*, Portland, OR, USA, Jul. 2015, pp. 367–389.
[42] G. Venter and R. T. Haftka, "Constrained particle swarm optimization using a bi-objective formulation," *Struct. Multidisciplinary Optim.*, vol. 40, nos. 1–6, pp. 65–76, 2010.
[43] L. D. Li, X. Li, and X. Yu, "A multi-objective constraint-handling method with PSO algorithm for constrained engineering optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 1528–1535.
[44] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 658–675, Dec. 2006.
[45] Y. Wang and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 117–134, Feb. 2012.
[46] Y. Wang and Z. Cai, "A dynamic hybrid framework for constrained evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 203–217, Feb. 2012.
[47] G. Reynoso-Meza, X. Blasco, J. Sanchis, and M. A. Martinez, "Multiobjective optimization algorithm for solving constrained single objective problems," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–7.
[48] R. Datta and K. Deb, "An adaptive normalization based constrained handling methodology with hybrid bi-objective and penalty function approach," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[49] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Gen. Algorithms Gen. Algorithms Appl.*, Cambridge, MA, USA, 1987, pp. 41–49.
[50] T. Takahama and S. Sakai, "Efficient constrained optimization by the ϵ constrained differential evolution with rough approximation using kernel regression," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 1334–1341.
[51] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, *Metaheuristics for Hard Optimization, Simulated Annealing, Tabu Search, Evolutionary and Genetic Algorithms, Ant Colonies, Methods and Case Studies*. Heidelberg, Germany: Springer, 2006.
[52] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 450–455, Jun. 2014.
[53] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Mar. 2011.
[54] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[55] J. J. Liang *et al*. (2006). *Problem Definitions and Evaluation Criteria for the CEC2006 Special Session on Constrained Real-Parameter Optimization*. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/
[56] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization," School Elect. Electron. Eng., Nanyang Technol. Univ., Singapore, Tech. Rep., 2010.
[57] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 561–579, Aug. 2010.
[58] W. Zhang, G. G. Yen, and Z. He, "Constrained optimization via artificial immune system," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 185–198, Feb. 2014.

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

He has been a Professor with the China University of Geosciences, Wuhan, since 2004. His current research interests include evolutionary computation with machine learning for solving problems with constraints, multiobjective, dynamic environments, and expensive costs, especially the antenna design problems.

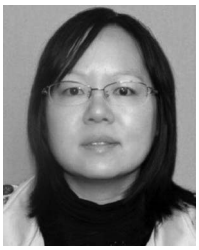


Ruwang Jiao is currently pursuing the M.Sc. degree in computer science with the China University of Geosciences, Wuhan, China.

His current research interests include evolutionary algorithms for solving optimization problems with constraints, especially the antenna design problems.

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

He has been an Associate Professor with the China University of Geosciences since 2011. He is the Vice Chair of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments. His current research interests include evolutionary algorithms with machine leaning, swarm intelligence, multimodal optimization, and dynamic optimization.



Xi Li received the B.Sc. degree in computer science from the Ocean University of China, Qingdao, China, in 2000, and the M.Sc. degree from the Hebei University of Science and Technology, Shijiazhuang, China, in 2008. She is currently pursuing the Ph.D. degree with the China University of Geosciences, Wuhan, China.

She is a Lecturer with Hebei GEO University, Shijiazhuang. Her current research interests include evolutionary computation with machine learning. She is currently involved in multiobjective/constrained/dynamic evolutionary optimization researches.

Jawdat S. Alkasassbeh was born in 1983 in Jordan. He received the B.Sc. degree in communications engineering from the Department of Electrical Engineering, Faculty of Engineering, Mutah University, Al-Karak, Jordan, in 2006, and the master's degree in communications engineering from the University of Jordan, Amman, Jordan, in 2011. He is currently pursuing the Ph.D. degree with the School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan, China.

His current research interests include evolutionary algorithms for antenna design, power reduction of mobile communication mechanisms, digital wireless communication systems, radio link design, and adaptive modulation techniques.