A Comparative Study of Constraint-Handling Techniques in Evolutionary Constrained Multiobjective Optimization

Jia-Peng Li, Yong Wang, *Member, IEEE,* Shengxiang Yang, *Senior Member, IEEE*, and Zixing Cai, *Senior Member, IEEE*

***Abstract*—Solving constrained multiobjective optimization problems is one of the most challenging areas in the evolutionary computation research community. To solve a constrained multiobjective optimization problem, an algorithm should tackle the objective functions and the constraints simultaneously. As a result, many constraint-handling techniques have been proposed. However, most of the existing constraint-handling techniques are developed to solve test instances (e.g., CTPs) with low dimension and large feasible region. On the other hand, experimental comparisons on different constraint-handling techniques remain scarce. In view of these two issues, in this paper we first construct 18 test instances, each of which exhibits different properties. Afterward, we choose three representative constraint-handling techniques and combine them with nondominated sorting genetic algorithm II to study the performance difference on various conditions. By the experimental studies, we point out the advantages and disadvantages of different constraint-handling techniques.**

***Keywords*—Constraint-handling techniques, constrained multiobjective optimization problems, evolutionary algorithms, test instances.**

I. Introduction

Numerous real-world applications involve multiple objectives and various constraints, which can be modeled as constrained multiobjective optimization problems (CMOPs). Without loss of generality, a CMOP can be formulated as:

$$\text{Minimize } \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})),\ \vec{x} = (x_1, \ldots, x_n) \in S$$
$$\text{Subject to: } g_j(\vec{x}) \le 0,\ j = 1, ..., p$$
$$h_j(\vec{x}) = 0,\ j = p+1, ..., q$$

where $\vec{x}$ is the decision vector, $L_i \le x_i \le U_i (i \in \{1, \ldots, n\})$ is the ith decision variable, L_i and U_i are the lower and upper bounds of x_i, S is the decision space, n and m are the number of decision variables and objective functions, $f_i(\vec{x})\ (i \in \{1, \ldots, m\})$ is the ith objective function, $g_j(\vec{x})$ is the jth inequality constraint, $h_j(\vec{x})$ is the $(j-p)$th equality constraint, p is the number of inequality constraints, and $(q-p)$ is the number of equality constraints.

J.-P. Li and Z. Cai are with the School of Information Science and Engineering, Central South University, Changsha 410083, China. (e-mail: ljpcsu@csu.edu.cn; zxcai@csu.edu.cn)

Y. Wang (Corresponding Author) is with the School of Information Science and Engineering, Central South University, Changsha 410083, China, and also with the School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK. (e-mail: ywang@csu.edu.cn)

S. Yang is with the School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK. (e-mail: syang@dmu.ac.uk)

The constraint violation of individual $\vec{x}$ on the jth constraint is calculated as follows:

$$C_j(\vec{x}) = \begin{cases} \max(0, g_j(\vec{x})), & 1 \le j \le p \\ \max(0, |h_j(\vec{x})| - \delta), & p+1 \le j \le q \end{cases} \quad (1)$$

where δ is a predefined tolerance value to relax the equality constrains to a certain extent. The feasible region Ω of a CMOP is a subspace of the decision space S, and can be defined as $\Omega = \{\vec{x} \in S \mid C_j(\vec{x}) = 0, j = 1, \ldots, q\}$.

Evolutionary algorithms (EAs) are population based optimization approaches inspired by nature [1]. During the last two decades, much effort has been made to develop and understand EAs for dealing with multiobjective optimization problems [2] [3]. Constraint-handling for single-objective optimization problems has also been extensively researched [4] [5]. However, few attempts have been made to investigate evolutionary constrained multiobjective optimization [6], which is challenging and of practical interest.

For CMOPs, we cannot find a single solution to optimize all the objectives at the same time since the objectives are always in conflict with each other. Therefore, when using EAs to deal with CMOPs, we have to balance the objectives and find a set of optimal tradeoffs. In addition, it is necessary to note that EAs are unconstrained search methods that need additional mechanisms to deal with constraints when solving CMOPs. Due to the above attributes, the solution of CMOPs is very difficult and the research of CMOPs based on EAs is still in its infant stage.

When solving CMOPs by EAs, the corresponding methods are called constrained multiobjective EAs (CMOEAs). Next, we will briefly introduce some representative CMOEAs. Deb *et al*. [7] extended the feasibility rule in [4] and proposed a constrained-domination principle (CDP) to solve CMOPs. Based on the work in [7], Oyama *et al.* [8] proposed a new constraint-handling technique, which introduces the idea of nondominance and niching concepts in the objective space into the constraint space. Chafekar *et al*. [9] introduced a steady state GA to solve CMOPs. In [9], two constrained multiobjective optimization techniques are applied. One technique is to run several single-objective GAs concurrently and each GA is to optimize one objective. The other technique is to use a single-objective GA to optimize multiple objectives in a sequential order. Young [10] presented a blended rank mechanism in which every solution is assigned two Pareto dominance ranks, and both two ranks are taken into account to give the final rank of an individual. Jimenez *et al*. [11] combined the Pareto concept in multiobjective optimization

with a novel diversity mechanism to sort the population. Woldesenbet and Yen [6] proposed a self-adaptive penalty function to handle constraints based on the feasible ratio in the current population. Santana-Quintero *et al*. [12] presented a hybrid CMOEA by approximating the Pareto front of a CMOP with a hybrid approach. In [12], rough set theory is implemented to maintain a good quality of the approximation. By combining CDP [7] with simulated annealing, Singh *et al*. [13] proposed a non-greedy constraint-handling technique to solve CMOPs. Jiao *et al.* [14] proposed a CMOEA based on [6], the main ideas of which are to make use of the information of the infeasible individuals and to construct some modified objective functions to guide the evolutionary process. Jan and Khanum [15] proposed two modified constraint-handling techniques by combining Tchebycheff aggregation function with the famous stochastic ranking [16] and CDP [7] to solve CMOPs. Very recently, an immune optimization algorithm is proposed by Qian *et al.* [17] to solve CMOPs. Taking both convergence and diversity into account, they divided a population into two subpopulations and applied different reproduction operators to create the next population. Recognizing that there is no single constraint-handling technique can outperform all others on different kinds of problems, Qu and Suganthan [18] proposed an ensemble of constraint-handling techniques. In [18], three constraint-handling techniques, i.e., CDP [7], self-adaptive penalty (SP) [6], and ε constrained method [19] are adopted.

It is noteworthy that most of the above methods are designed to solve the test instances (called CTPs) in [20], which were proposed by Deb *et al.* fifteen years ago. Due to the fact that CTPs usually have a large proportion of the feasible region, it is a very important topic to design CMOPs with more complex characteristics. On the other hand, the current methods are usually developed to solve CTPs with low dimension (such as two dimensions), and there are very few studies conducted on the performance comparison of different constraint-handling techniques in the community of evolutionary constrained multiobjective optimization.

Motivated by the above considerations, in this paper we firstly construct 18 test instances with different characteristics based on CTPs. Afterward, we select three representative constraint-handling techniques (CDP [7], SP [6], and adaptive tradeoff model (ATM) [21]). Then we study their performance difference on various scenarios. The main contributions of this paper can be summarized as follows:

- We design 18 new test instances (called NCTPs) based on CTPs. Compared with the original CTPs, NCTPs introduced in this paper exhibit more complex characteristics. Specifically, in NCTPs different test instances have different shapes of the Pareto front, different dimensions of the search space, and different size of the feasible region.
- Systematic experiments have been conducted on the 18 test instances to study the performance of the three representative constraint-handling techniques according to the shape of the Pareto front, the dimension of search space, and the size of the feasible region.

The rest of this paper is organized as follows. Section II briefly describes the related definitions. Section III introduces three representative constraint-handling techniques. Section IV presents the constructed test instances. Section V experimentally studies the performance of the chosen constraint-handling techniques. Section VI concludes this paper.

II. The Related Definitions of CMOPs

A CMOP include multiple objectives. In multiobjective optimization, the comparison of individuals is usually based on Pareto dominance. Next, we will introduce four related definitions.

Definition 1 (Pareto Dominance): Considering the m objective functions $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}))$ and two decision vectors $\vec{x}_v$ and $\vec{x}_u$, if $\forall i \in \{1,...,m\}$, $f_i(\vec{x}_v) \leq f_i(\vec{x}_u)$ and $\exists i \in \{1,...,m\}$, $f_j(\vec{x}_v) < f_j(\vec{x}_u)$, then $\vec{x}_v$ is said to Pareto dominate $\vec{x}_u$, denoted as $\vec{x}_v \prec \vec{x}_u$.

Definition 2 (Pareto Optimal Solution): A solution $\vec{x}_u \in \Omega$ is called a Pareto optimal solution of a CMOP if and only if $\neg\exists \vec{x}_v \in \Omega,\ \vec{x}_v \prec \vec{x}_u$.

Definition 3 (Pareto Optimal Set): The Pareto optimal set of a CMOP can be defined as $POS = \{\vec{x}_u \in \Omega \mid \neg\exists \vec{x}_v \in \Omega, \vec{x}_v \prec \vec{x}_u\}$.

Definition 4 (Pareto Front): The Pareto front of a CMOP can be defined as $PF = \{\vec{f}(\vec{x}_u) \mid \vec{x}_u \in POS\}$.

III. Three Representative Constraint-handling Techniques for CMOPs

A. Constrained-domination Principle (CDP)

CDP [7] is a simple and efficient technique to handle constrains, which compares pairwise individuals based on the following rules:

- When two feasible solutions are compared, the one Pareto dominating the other is better.
- When a feasible solution is compared with an infeasible solution, the feasible solution is better.
- When two infeasible solutions are compared, the one with smaller degree of constraint violation is better.

B. Self-adaptive Penalty (SP)

Another way to solve CMOPs is to penalize the infeasible individual with penalty function, in which the penalty term added to the objective function is based on the degree of constraint violation of the infeasible individual. Among all penalty function based methods, the self-adaptive penalty (SP) proposed in [6] is a representative one. It has two main components: the distance value and the penalty function. Firstly, the objective functions and the degree of constraint

violation of each individual $\vec{x}_j$ in the population are normalized as follows:

$$\tilde{f}_i(\vec{x}_j) = \frac{f_i(\vec{x}_j) - f_i^{\min}}{f_i^{\max} - f_i^{\min}},\ i \in \{1,\ldots,m\} \tag{2}$$

$$C(\vec{x}_j) = \frac{1}{q}\sum_{i=1}^{q}\frac{C_i(\vec{x}_j)}{C_i^{\max}} \tag{3}$$

where $f_i^{\min}$ is the minimum value of the ith objective function, $f_i^{\max}$ is the maximum value of the ith objective function, and $C_i^{\max}$ is the maximum violation of the ith constraint.

Let r_f denote the feasibility ratio of the current population, which can be calculated as follows:

$$r_f = \frac{\text{the number of feasible individuals}}{\text{the population size}} \tag{4}$$

Afterward, the ith $(i \in \{1,\ldots,m\})$ distance value $d_i(\vec{x}_j)$ and ith $(i \in \{1,\ldots,m\})$ penalty value $p_i(\vec{x}_j)$ of $\vec{x}_j$ can be expressed as follows:

$$d_i(\vec{x}_j) = \begin{cases} C(\vec{x}_j), & if\ r_f = 0 \\ \sqrt{\tilde{f}_i(\vec{x}_j)^2 + C(\vec{x}_j)^2}, & if\ r_f \neq 0 \end{cases} \tag{5}$$

$$p_i(\vec{x}_j) = (1-r_f)X_i(\vec{x}_j) + r_f Y_i(\vec{x}_j) \tag{6}$$

where

$$X_i(\vec{x}_j) = \begin{cases} 0, & if\ r_f = 0 \\ C(\vec{x}_j), & if\ r_f \neq 0 \end{cases} \tag{7}$$

$$Y_i(\vec{x}_j) = \begin{cases} 0, & if\ \vec{x}_j \text{ is feasible} \\ \tilde{f}_i(\vec{x}_j), & if\ \vec{x}_j \text{ is infeasible} \end{cases} \tag{8}$$

Finally, the fitness of $\vec{x}_j$ in the ith objective function dimension is the sum of $d_i(\vec{x}_j)$ and $p_i(\vec{x}_j)$:

$$F_i(\vec{x}_j) = d_i(\vec{x}_j) + p_i(\vec{x}_j),\ i \in \{1,\ldots,m\} \tag{9}$$

Then the population is sorted based on the m fitness functions $F_1, F_2, \ldots, F_m$ via the nondominated sorting [7].

C. Adaptive Tradeoff Model (ATM)

ATM introduces an important idea, i.e., dividing the evolutionary process into three situations by the feasibility proportion of the current population. In different situations, ATM adopts different techniques to cope with the constraints.

1) The infeasible situation: There is no feasible solution in the current population. ATM converts a CMOP with m objectives and q constraints into a unconstrained MOP with (m+1) objectives, by considering the constraint violation as an additional objective. Then the nondominated sorting [7] is applied, and half of the individuals with less constraint violations in the first layer are chosen and removed from the population. Afterward, the remaining individuals in the population are implemented the same process until a desirable number of individuals is obtained.

2) The semi-feasible situation: There exist both infeasible and feasible solutions in the current population. Assume that Z is the current population, and then Z is divided into the feasible group and the infeasible group based on equation (3):

$$Z_1 = \{\vec{x} \in Z \mid C(\vec{x}) = 0\} \tag{10}$$

$$Z_2 = \{\vec{x} \in Z \mid C(\vec{x}) > 0\} \tag{11}$$

Firstly, ATM uses the following transformed objective function to penalize the infeasible individuals:

$$f_i'(\vec{x}_j) = \begin{cases} f_i(\vec{x}_j), & if\ \vec{x}_j \text{ is feasible} \\ \max\{\varphi * f_{b\_i} + (1-\varphi) * f_{w\_i}, f_i(\vec{x}_j)\}, & if\ \vec{x}_j \text{ is infeasible} \end{cases} \tag{12}$$

where φ denotes the feasibility proportion of the last population, $f_{b\_i} = \min_{\vec{x}\in Z_1} f_i(\vec{x})$, and $f_{w\_i} = \max_{\vec{x}\in Z_1} f_i(\vec{x})$.

Then, ATM normalizes the transformed objective function and the constraint violation:

$$\tilde{f}_i(\vec{x}_j) = \frac{f_i'(\vec{x}_j) - \min_{\vec{x}\in Z} f_i'(\vec{x})}{\max_{\vec{x}\in Z} f_i'(\vec{x}) - \min_{\vec{x}\in Z} f_i'(\vec{x})},\ i \in \{1,\ldots,m\} \tag{13}$$

$$\tilde{C}(\vec{x}_j) = \begin{cases} 0, & if\ \vec{x}_j \in Z_1 \\ \dfrac{C(\vec{x}_j) - \min_{\vec{x}\in Z_2} C(\vec{x})}{\max_{\vec{x}\in Z_2} C(\vec{x}) - \min_{\vec{x}\in Z_2} C(\vec{x})}, & if\ \vec{x}_j \in Z_2 \end{cases} \tag{14}$$

The final fitness of $\vec{x}_j$ in the ith objective function dimension is the sum of $\tilde{f}_i(\vec{x}_j)$ and $\tilde{C}(\vec{x}_j)$:

$$F_i(\vec{x}_j) = \tilde{f}_i(\vec{x}_j) + \tilde{C}(\vec{x}_j),\ i \in \{1,\ldots,m\} \tag{15}$$

Like SP, the m fitness functions $F_1, F_2, \ldots, F_m$ are used to sort the population by the nondominated sorting [7].

3) The feasible phase: In this phase, all solutions in the population are feasible. Thus, ATM directly sorts the population by making use of the nondominated sorting [7].

IV. Constructed Test Instances

The widely used test instances in evolutionary constrained multiobjective optimization are CTPs [20], which can be divided in two parts. The first part is CTP1, in which the number of constraints is changeable, but with the increase of constraints the shape of the Pareto front almost remains the same. The second part is CTP2-CTP7. The shapes of the Pareto front of these test instances are tunable and controlled by six parameters, but these test instances only have one constraint and the size of the feasible region is usually large.

To study the performance of the three chosen constraint-handling techniques in Section III on different conditions, the above drawbacks of test instances should be addressed. In this paper, a set of new constrained test instances (called NCTPs) is constructed based on CTP2-CTP7. In NCTPs, the following new characteristics have been added:

- In most of papers, $g(\vec{x})$ in the second objective function of CTP2-CTP7 is often set to $(1+x_2)$. To increase the difficulty, in this paper $g(x)$ has been set to the Ronsenbrock function [22]:

$$g(\vec{x}) = \sum_{i=2}^{n-1}(100(x_{i+1} - x_i^2)^2 + (1-x_i)^2) \tag{16}$$

TABLE I
Detailed information of the 18 test instances, where "Type I" denotes that the Pareto front is discontinuous (see Fig. 1), "Type II" denotes that the Pareto front consists of both discontinuous and continuous parts (see Fig. 2), and "Type III" denotes that the Pareto front is continuous (see Fig. 3).

Test instance	Parameter value	Constraint	Properties		
			Shape of the Pareto front	Dimension of the decision vector	Size of the feasible region
NCTP-1	$\theta=-0.2\pi,\ a=0.2,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=-0.5,z_2=4$	C_1 and C_2	Type I	10D/30D	Small (<0.1%)
NCTP-2	$\theta=-0.2\pi,\ a=0.75,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=-0.5,z_2=4$	C_1 and C_2	Type I	10D/30D	Small (<0.1%)
NCTP-3	$\theta=-0.2\pi,\ a=2,\ b=10,\ c=1,\ d=6,\ e=1,z_1=-0.5,z_2=6$	C_1 and C_2	Type I	10D/30D	Small (<0.1%)
NCTP-4	$\theta=-0.2\pi,\ a=0.2,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=-0.5$	C_1	Type I	10D/30D	Large (>99.9%)
NCTP-5	$\theta=-0.2\pi,\ a=0.75,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=-0.5$	C_1	Type I	10D/30D	Large (>99.9%)
NCTP-6	$\theta=-0.2\pi,\ a=2,\ b=10,\ c=1,\ d=6,\ e=1,z_1=-0.5$	C_1	Type I	10D/30D	Large (>99.9%)
NCTP-7	$\theta=-0.2\pi,\ a=0.2,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=1,z_2=4$	C_1 and C_2	Type II	10D/30D	Small (<0.1%)
NCTP-8	$\theta=-0.2\pi,\ a=0.75,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=1,z_2=4$	C_1 and C_2	Type II	10D/30D	Small (<0.1%)
NCTP-9	$\theta=-0.2\pi,\ a=2,\ b=10,\ c=1,\ d=6,\ e=1,z_1=1,z_2=6$	C_1 and C_2	Type II	10D/30D	Small (<0.1%)
NCTP-10	$\theta=-0.2\pi,\ a=0.2,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=1$	C_1	Type II	10D/30D	Large (>99.9%)
NCTP-11	$\theta=-0.2\pi,\ a=0.75,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=1$	C_1	Type II	10D/30D	Large (>99.9%)
NCTP-12	$\theta=-0.2\pi,\ a=2,\ b=10,\ c=1,\ d=6,\ e=1,z_1=1$	C_1	Type II	10D/30D	Large (>99.9%)
NCTP-13	$\theta=-0.2\pi,\ a=0.2,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=2,z_2=4$	C_1 and C_2	Type III	10D/30D	Small (<0.1%)
NCTP-14	$\theta=-0.2\pi,\ a=0.75,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=2.5,z_2=4$	C_1 and C_2	Type III	10D/30D	Small (<0.1%)
NCTP-15	$\theta=-0.2\pi,\ a=2,\ b=10,\ c=1,\ d=6,\ e=1,z_1=4,z_2=6$	C_1 and C_2	Type III	10D/30D	Small (<0.1%)
NCTP-16	$\theta=-0.2\pi,\ a=0.2,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=2$	C_1	Type III	10D/30D	Large (>99.9%)
NCTP-17	$\theta=-0.2\pi,\ a=0.75,\ b=10,\ c=1,\ d=0.5,\ e=1,z_1=2.5$	C_1	Type III	10D/30D	Large (>99.9%)
NCTP-18	$\theta=-0.2\pi,\ a=2,\ b=10,\ c=1,\ d=6,\ e=1,z_1=4$	C_1	Type III	10D/30D	Large (>99.9%)

Meanwhile, the decision space has been changed from $[0,1]^n$ to $[0,5]^n$.

- An additional constraint ($C_2(\vec{x})$) has been added to adjust the size of the feasible region.
- An additional parameter (z_1) has been added to the second objective function to control the proportion of the continuous part of the Pareto front.
- $f_1(\vec{x})$ has been changed to $\sqrt{f_1(\vec{x})}$ in the second objective, with the aim of making the unconstrained Pareto front a curve rather than a line segment.

The proposed test instances can be described as follows:

$$\begin{aligned}
&\text{Minimize } f_1(\vec{x}) = x_1\\
&\text{Minimize } f_2(\vec{x}) = g(\vec{x})\left(1-\frac{\sqrt{f_1(\vec{x})}}{g(\vec{x})}\right)+z_1\\
&\text{subject to } C_1(\vec{x}) = \cos(\theta)(f_2(\vec{x})-e)-\sin(\theta)f_1(\vec{x})\\
&\qquad\qquad \geq a\,|\sin(b\pi(\sin(\theta)(f_2(\vec{x})-e)+\cos(\theta)f_1(\vec{x}))^c)|^d\\
&\qquad C_2(\vec{x}) = f_2(\vec{x}) < -0.73f_1(\vec{x})+z_2
\end{aligned} \tag{17}$$

where θ, a, b, c, d, and e are used to control the topology of the Pareto front [20], z_1 is used to control the proportion of the continuous part of the Pareto front, and z_2 is used to control the size of the feasible region.

Based on the above formulation, 18 test instances have been designed in this paper to test the performance of the three representative constraint-handling techniques. The details of these test instances have been presented in Table I, and the objective spaces of these test instances have been presented in Fig. 1, Fig. 2, and Fig. 3.

V. Experimental Study

A. Experimental Setup

In this section, we choose the nondominated sorting genetic algorithm II (NSGA-II) [7] as the multiobjective optimization framework, due to its high robustness and ease of implementation. Then the three representative constraint-handling techniques (CDP, SP, and ATM) introduced in Section III are combined with NSGA-II to solve the 18 test instances proposed in Section IV. For all the experiments, the parameters were set as follows: the population size was 100, the crossover rate was 0.8, the mutation rate was $1/n$, and the maximum number of generation was 500. Besides, the tournament selection, simulated binary crossover, and polynomial mutation were adopted as the genetic operators in each simulation.

As presented in Table I, this paper intends to study how the following three properties of the 18 test instances influence the performance of a constraint-handling technique for CMOPs.

- The shape of Pareto front
- The dimension of decision vector
- The size of feasible region

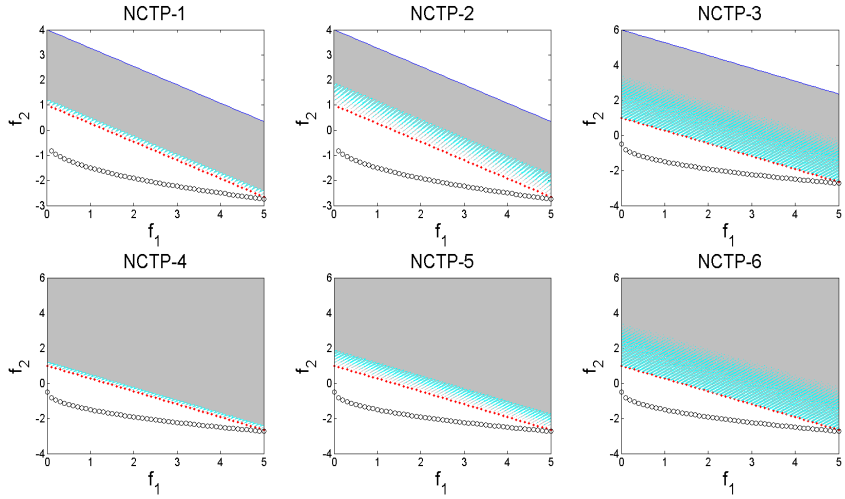

Fig. 1. The objective spaces of six test instances in type I, where the shade indicates the feasible region in the objective space, the black circle line indicates the unconstrained Pareto front, and the red dotted line indicates the constrained Pareto front.

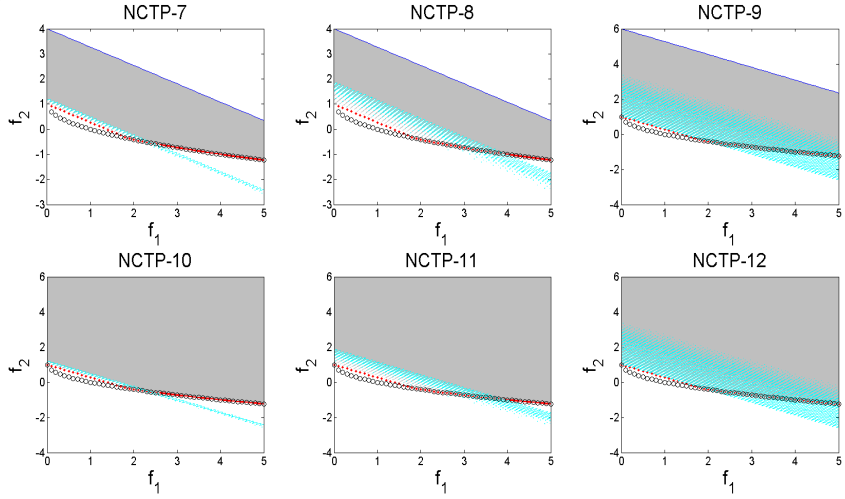

Fig. 2. The objective spaces of six test instances in type II, where the shade indicates the feasible region in the objective space, the black circle line indicates the unconstrained Pareto front, and the red dotted line indicates the constrained Pareto front.

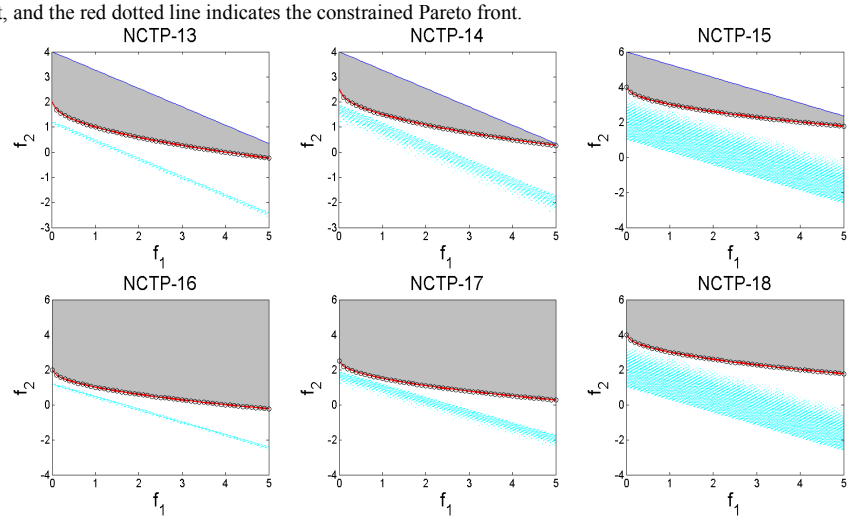

Fig. 3. The objective spaces of six test instances in type III, where the shade indicates the feasible region in the objective space, the black circle line indicates the unconstrained Pareto front, and the red dotted line indicates the constrained Pareto front.

In order to study the effect of the second property, we tested two scenarios: n=10 and n=30. For each scenario, 100 independent runs were implemented for each test instance.

According to our observation, the performance of an algorithm significantly degenerates with the increase of the dimension of the decision vector. More importantly,

TABLE II
Test instances included in each case: 10D-S includes the test instances with n=10 and small feasible regions, 30D-S includes the test instances with n=30 and small feasible regions, 10D-L includes the test instances with n=10 and large feasible regions, and 30D-L includes the test instances with n=30 and large feasible regions.

Type	Type I				Type II				Type III			
Case	10D-S	30D-S	10D-L	30D-L	10D-S	30D-S	10D-L	30D-L	10D-S	30D-S	10D-L	30D-L
Test instance	NCTP-1		NCTP-4		NCTP-7		NCTP-10		NCTP-13		NCTP-16	
	NCTP-2		NCTP-5		NCTP-8		NCTP-11		NCTP-14		NCTP-17	
	NCTP-3		NCTP-6		NCTP-9		NCTP-12		NCTP-15		NCTP-18	

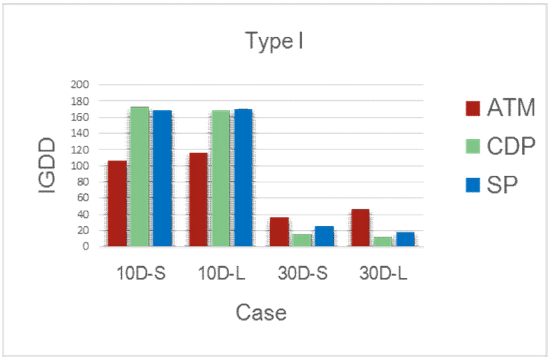

Fig. 4. The experimental results of the four cases in type I

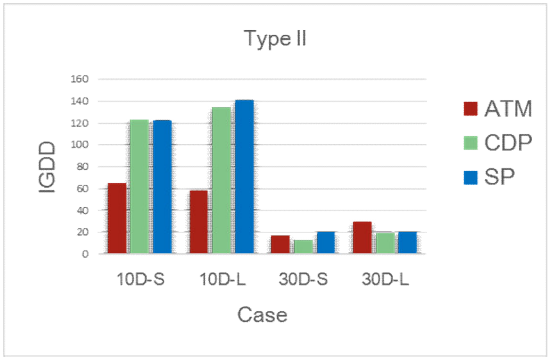

Fig. 5. The experimental results of the four cases in type II

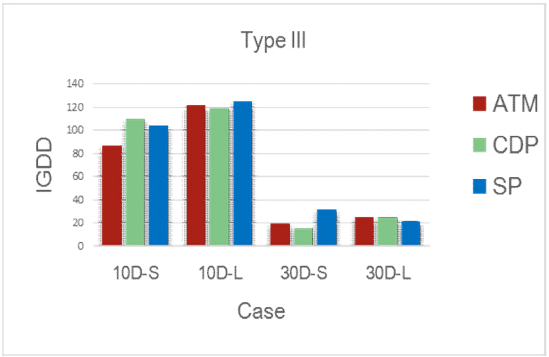

Fig. 6. The experimental results of the four cases in type III

sometimes an algorithm cannot find any feasible solution on the nine test instances with a small feasible region when the evolution halts. The inverted generational distance (IGD) [23] is a widely used indicator to evaluate the performance of an algorithm for multiobjective optimization. Note that if an algorithm cannot find any feasible solution, we cannot obtain the IGD value. Thus, IGD might not be suitable to assess the performance of an algorithm for CMOPs with a small feasible region. To this end, we propose a revised IGD indicator, called IGD distribution indicator (IGDD) in this paper. The IGDD value of an algorithm is computed as follows:

- Step 1): Calculate the IGD value in each run. As a result, we obtain 100 IGD values in 100 runs.
- Step 2): Define two ranges of the IGD value: $range_1 = (0, 0.25]$ and $range_2 = (0.25, 0.5]$, compute the number (denoted as n_1) of the IGD value belonging to $range_1$, and compute the number (denoted as n_2) of the IGD value belonging to $range_2$. Note that $n_1 + n_2 \le 100$.
- Step 3): Calculate the IGDD value by the following equation:

$$\text{IGDD} = \alpha n_1 + (1-\alpha) n_2 \tag{18}$$

where α is an coefficient that is used to adjust the weight of n_1 and n_2. Since the IGD value in $range_1$ is better than that in $range_2$, n_1 should be put more emphasis. In this paper, α was set to 0.8.

The IGDD indicator can be used to measure the convergence towards the Pareto front. The higher the IGDD value, the better the performance of an algorithm.

B. Comparative Study

For the sake of clarity, the comparative study is based on the three different types of shapes of the Pareto front introduced in Table I:

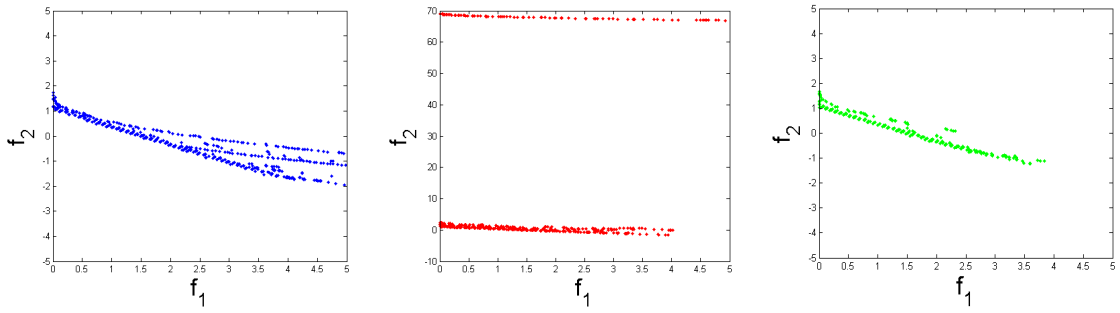

Fig. 7. 10 final Pareto fronts obtained by ATM, CDP, and SP (from left to right) on NCTP-4 in type I, which belongs to 30D-L.

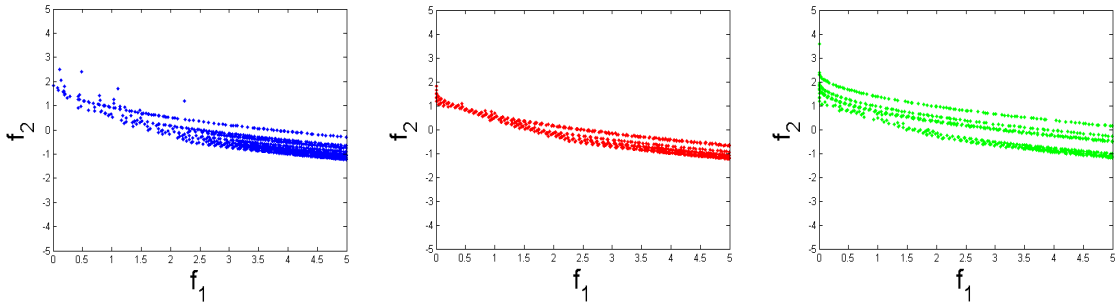

Fig. 8. 10 final Pareto fronts obtained by ATM, CDP, and SP (from left to right) on NCTP-11 in type II, which belongs to 10D-L.

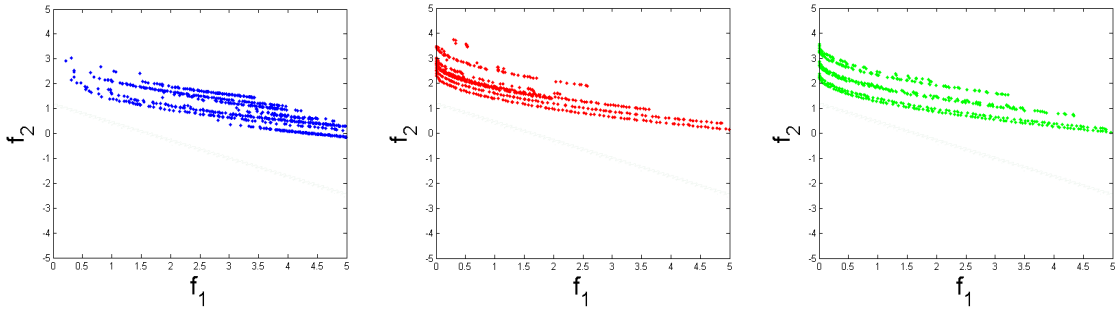

Fig. 9. 10 final Pareto fronts obtained by ATM, CDP, and SP (from left to right) on NCTP-13 in type III, which belongs to 30D-S.

- "Type I" denotes that the Pareto front is discontinuous (NCTP-1-NCTP-6).
- "Type II" denotes that the Pareto front consists of both discontinuous part and continuous part (NCTP-7-NCTP-12).
- "Type III" denotes that the Pareto front is continuous (NCTP-13-NCTP-18).

Based on the dimension of the decision vector and the size of the feasible region, each type can be divided into four cases (10D-S, 30D-S, 10D-L, and 30D-L) as shown in Table II. The IGDD values in the same case of each type are added together to represent the overall performance of a constraint-handling technique (see Figs. 4-6). For example, the experimental result of ATM in "10D-S" of Fig. 4 denotes the sum of the IGDD values on NCTP-1 with 10D, NCTP-2 with 10D, and NCTP-3 with 10D.

From the experimental results of Figs. 4-6, we can give the following comments:

- *Type I:* As shown in Fig. 4, CDP and SP perform similarly in all the four cases. For the test instances with 10D, CDP and SP performs much better than ATM, regardless of the size of feasible region. However, ATM outperforms CDP and SP on the test instances with 30D. Fig. 7 further illustrates this phenomenon. This figure shows that the solutions obtained by ATM are more close to the Pareto front and have a better distribution. Compared with ATM, some solutions found by CDP are far away from the Pareto front, and the solutions obtained by SP are not well distributed.
- *Type II:* Similar to type I, in type II CDP and SP surpass ATM on the test instances with low dimension (10D). Fig. 8 provides the performance comparison of ATM, CDP, and SP on NCTP-11 with 10D. However, ATM and SP are slightly better than CDP on the test instances with 30D and small feasible regions. Moreover, ATM ranks the first on the test instances with 30D and large feasible regions.
- *Type III*: In the case of 10D-S, CDP is better than SP and ATM. In terms of 30D-S, SP ranks the first, followed by ATM, which is further illustrated in Fig. 9. With regard to 10D-L and 30D-L, the three

constraint-handling techniques show comparable performance.

- Overall, the increase of the number of decision variables poses a grand challenge to the three constraint-handling techniques. In comparison with the IGDD values of the 10D test instances, the IGDD values of the 30D test instances decrease significantly.

VI. Conclusion and Future Work

According to the preliminary experimental comparison in Section V, the following conclusions can be made according to the IGDD values of the three constraint-handling techniques:

- CDP and SP show comparable performance on all test instances, and they outperform ATM on test instances with 10D.
- ATM exhibits competitive performance on test instances with 30D.

It is necessary to note that the above conclusions are obtained by experiments. In the future, we will further analyze the advantages and disadvantages of the constraint-handling techniques on different kinds of CMOPs in principle. On the other hand, we will design new constraint-handling techniques to tackle the 18 test instances developed in this paper.

The Matlab source code can be downloaded from Y. Wang's homepage: http://ist.csu.edu.cn/YongWang.htm

Acknowledgments

This work was supported in part by the Innovation-driven Plan in Central South University (No. 2015CXS012 and No. 2015CX007), in part by the National Natural Science Foundation of China under Grant 61273314, in part by the EU Horizon 2020 Marie Sklodowska-Curie Individual Fellowships (Project ID: 661327), in part by the Engineering and Physical Sciences Research Council of UK under Grant EP/K001310/1, in part by the Hunan Provincial Natural Science Fund for Distinguished Young Scholars (Grant No. 2016JJ1018), and in part by the Program for New Century Excellent Talents in University under Grant NCET-13-0596.

References

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. IOP Publishing Ltd., 1997.

[2] K. Deb, *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[3] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic, 2002.

[4] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311-338, 2000.

[5] Z. Cai and Y. Wang, "A multiobjective optimization based evolutionary algorithm for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 658-675, 2006.

[6] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 514-525, 2009.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[8] A. Oyama, K. Shimoyama, and K. Fujii "New constraint-handling method for multi-objective multi-constraint evolutionary optimization and its application to space plane design," in *Evolutionary and Deterministic Techniques for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2005)*, Munich, Germany, 2005, pp. 1-13.

[9] D. Chafekar, J. Xuan, and K. Rasheed, "Constrained multi-objective optimization using steady state genetic algorithms," in *Proc. Genetic and Evol. Comput. Conf.*, 2003, pp. 813-824.

[10] N. Young, "Blended ranking to cross infeasible regions in constrained multiobjective problems," in *Proceedings of International Conference on Computional Intellignece for Modeling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 2005, pp. 191-196.

[11] F. Jimenez, A. F. Gomez-Skarmeta, G. Sanchez, and K. Deb, "An evolutionary algorithm for constrained multi-objective optimization," in *Proc. CEC*, 2002, pp. 1133-1138.

[12] L. V. Santana-Quintero, A. G. Hernández-Díaz, J. Molina, C. A. Coello Coello, and R. Caballero, "DEMORS: A hybrid multi-objective optimization algorithm using differential evolution and rough set theory for constrained problems," *Computers & Operations Research*, vol. 37, no. 3, pp. 470-480, 2010.

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

[14] L. Jiao, J. Luo, R. Shang, and F. Liu, "A modified objective function method with feasible-guiding strategy to solve constrained multi-objective optimization problems," *Applied Soft Computing*, vol. 14, pp. 363-380, 2014.

[15] M. A. Jan and R. A. Khanum, "A study of two penalty-parameterless constraint handling techniques in the framework of MOEA/D," *Applied Soft Computing*, vol. 13, no. 1, pp. 128-148, 2013.

[16] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284-294, 2000.

[17] S. Qian, Y. Ye, B. Jiang, and J. Wang, "Constrained multiobjective optimization algorithm based on immune system model," *IEEE Transactions on Cybernetics*, 2015, in press.

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

[19] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. CEC*, 2006, pp. 1-8.

[20] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, 2001, pp. 284-298.

[21] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng. "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80-92, 2008.

[22] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107-125, 2008.

[23] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117-132, 2003.