A New Multi-objective Model for Constrained Optimisation

Xu, Tao; He, Jun; Shang, Changjing; Ying, Weiqin

Published in:
Advances in Computational Intelligence Systems

DOI:
10.1007/978-3-319-46562-3_6

Publication date:
2016

Citation for published version (APA):
Xu, T., He, J., Shang, C., & Ying, W. (2016). A New Multi-objective Model for Constrained Optimisation. In P. Angelov, A. Gegov, C. Jayne, & Q. Shen (Eds.), *Advances in Computational Intelligence Systems: Contributions presented at the 16th UK Workshop on Computational Intelligence, September 7-9, 2016, Lancaster UK* (pp. 71-85). [Chapter 6] (Advances in Computational Intelligence Systems; Vol. 513). Springer Nature. https://doi.org/10.1007/978-3-319-46562-3_6

A New Multi-objective Model for Constrained Optimisation

Tao Xu, Jun He, Changjing Shang and Weiqin Ying

Abstract Multi-objective optimization evolutionary algorithms have becoming a promising approach for solving constrained optimization problems in the last decade. Standard two-objective schemes aim at minimising the objective function and the degrees of violating constraints (the degrees of violating each constraint or their sum) simultaneously. This paper proposes a new multi-objective model for constrained optimization. The model keeps the standard objectives: the original objective function and the sum of the degrees of constraint violation. Besides them, other helper objectives are constructed, which are inspired from MOEA/D or Tchebycheff method for multi-objective optimization. The new helper objectives are weighted sums of the normalized original objective function and normalized degrees of constraint violation. The normalization is a major improvement. Unlike our previous model without the normalization, experimental results demonstrate that the new model is completely superior to the standard model with two objectives. This confirms our expectation that adding more help objectives may improve the solution quality significantly.

1 Introduction

Many constraint-handling techniques have been proposed in literature. The most popular constraint-handling techniques include penalty function methods, the feasibility rule, multi-objective optimization and repair methods. A detailed introduction to this topic can be found in several comprehensive surveys [1, 2, 3].

Tao Xu, Jun He, Changjing Shang
Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, United Kingdom, e-mail: \{tax2, jqh, cns\}@aber.ac.uk

Weiqin Ying
School of Software Engineering, South China University of Technology, Guangzhou, 510006, China, e-mail: yingweiqin@scut.edu.cn

This paper focuses on multi-objective optimization methods, which are regarded as one of the most promising ways for dealing with Constrained Optimization Problem (COPs) [4]. The technique is based on using multi-objective optimization evolutionary algorithms (MOEAs) for solving single-objective optimization problems. This idea can be traced back to 1990s [5] and it is also termed multi-objectivization [6]. Multi-objective methods separate the objective function and the constraint violation degrees into different fitness functions. This is unlike penalty functions, which combine them into a single fitness function. The main purpose of using multi-objective optimization is to relax the requirement of setting or fine-tuning parameters, as happens with penalty function methods.

There exist variant multi-objective methods for solving COPs. Following the taxonomy proposed in [7, 4], these methods are classified into three categories according to the number of objectives and their construction.

1. *Standard bi-objective methods:* transform the original single-objective COP into an unconstrained bi-objective optimization problem, where one objective is the original objective function and the other is a measure of constraint violations. A lot of work has been done in this category, such as [8, 9, 10, 11, 12, 13, 14, 15, 16, 17].
2. *Standard multi-objective methods:* transform the original single-objective COP into an unconstrained multi-objective optimization problem (MOP), in which the degree of each constraint violation in the COP is a separate objective in addition to the original objective. The work in this category includes [18, 19, 20, 21, 22, 23, 24].
3. *Generalized multi-objective methods:* transform the original single-objective COP into an unconstrained MOP, in which at least one objective in multi-objective optimization is different from the original objective function and the degree of constraint violations. This category includes the work such as [25, 26, 27, 28].

The multi-objective method in this paper belongs to the third category: generalized multi-objective methods. Our multi-objective model keeps the standard objective functions: the original objective function and the total degree of constraint violation. But besides them, a helper objective is added into the model. The approach is similar to weighted metric methods for multi-objective optimization. The new helper objective is the weighted sum of a normalized original objective function and normalized degrees of constraint violation. Our research question is to investigate whether adding one more helper objective is better than those with the standard model with two objectives?

This paper conducts an experimental study. In order to evaluate the performance of our new model, it is compared with the standard model with two objectives using a multi-objective differential evolution algorithm, called CMODE [29]. CMODE has been proven to be efficient in solving MOPs from COPs. This paper is a further development of our previous initial study [30]. Experimental results in [30] are not good as those in [29] partially because only a simplified version of CMODE is

implemented in [30] for solving MOPs. In this paper, a complete version of CMODE is implemented which includes an infeasible solution replacement mechanism.

There are two new contributions in this paper. First, a normalization procedure is applied to both the original objective function and the degree of constraint violation. In this way, the original objective function and the degree of constrain violation play an equal role. Secondly, the new helper objectives are constructed from weighted sums of the original objective function and the degree of constraint violation, rather than penalty functions used in [30]. The new design is simple and easy without setting a penalty coefficient.

The rest of paper is organized as follows. Section 2 proposes a new multi-objective model for COPs. Section 3 describes CMODE, a multi-objective differential evolution algorithm for COPs. Section 4 reports experiment results and compares the performance of CMODE with different numbers of objectives. Section 5 concludes the paper.

2 A New Multi-objective Model for Constrained Optimisation

2.1 A Standard Two-Objective Model

For the sake of illustration, we consider a minimization problem which is formulated as follows:

$$\min f(\mathbf{x}), \quad \mathbf{x} = (x_1, \cdots, x_n) \in S, \tag{1}$$

$$\text{subject to} \begin{cases} g_i(\mathbf{x}) \leq 0, & i = 1, \cdots, q, \\ h_j(\mathbf{x}) = 0, & j = 1, \cdots, r, \end{cases} \tag{2}$$

where S is a bounded domain in $\mathbb{R}^n$, given by

$$S = \{\mathbf{x} \mid L_i \leq x_i \leq U_i,\ i = 1, \cdots, n\}. \tag{3}$$

L_i is the lower boundary and U_i the upper boundary. $g_i(\mathbf{x}) \leq 0$ is the ith inequality constraint while $h_j(\mathbf{x}) = 0$ is the jth equality constraints. The feasible region $\Omega \subseteq S$ is defined as:

$$\{\mathbf{x} \in S \mid g_i(\mathbf{x}) \leq 0, i = 1, \cdots, q;\ h_j(\mathbf{x}) = 0, j = 1, \cdots, r\}.$$

If an inequality constraint meets $g_i(\mathbf{x}) = 0$ (where $i = 1, \cdots, q$) at any point $\mathbf{x} \subseteq \Omega$, we say it is active at $\mathbf{x}$. All equality constraints $h_j(\mathbf{x})$ (where $j = 1, \cdots, r$) are considered active at all points of Ω.

The above single-objective COP can be transferred into a two-objective optimization problem without constraints. The first objective is to minimize the original fitness function $f(\mathbf{x})$ without considering constraints:

$$\min f(\mathbf{x}). \tag{4}$$

Notice that the optimal solution to the above minimization problem might be an infeasible solution to the original COP (1). Therefore $f(\mathbf{x})$ is only a helper fitness function because minimizing it might not lead to the optimal feasible solution.

The second objective is to minimize the degree of constraint violation. For each inequality constraint, define the degree of violating the constraint is

$$v_i^g(\mathbf{x}) = \max\{0, g_i(\mathbf{x})\},\ i = 1, \cdots, q. \tag{5}$$

For each equality constraint, define the degree of violating the constraint is

$$v_j^h(\mathbf{x}) = \max\{0, |h_j(\mathbf{x})| - \delta\},\ j = 1, \cdots, r. \tag{6}$$

where δ is a tolerance allowed for the equality constraint. Then the second objective is to minimize the sum of constraint violation degrees:

$$\min v(\mathbf{x}) = \sum_i v_i^g(\mathbf{x}) + \sum_j v_j^h(\mathbf{x}). \tag{7}$$

The above two objectives are widely used in the existing multi-objective methods for constrained optimization [4].

2.2 A New Multi-Objective Model

Besides the above two fitness functions, we may construct more helper fitness functions [30]. In this paper we consider the weighted sum of $f(\mathbf{x})$ and $v(\mathbf{x})$. The idea is similar to Tchebycheff method for multi-objective optimization and decompose-based MOEAs (MOEA/D) [31].

Given a population $P = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, let $\hat{f}(\mathbf{x})$ be a normalized value of $f(\mathbf{x})$:

$$\hat{f}(\mathbf{x}) = \frac{f(\mathbf{x}) - f_{\min}(P)}{f_{\max}(P) - f_{\min}(P)} \tag{8}$$

where $f_{\min}(P)$ and $f_{\max}(P)$ are the minimal and maximal values of $f(\mathbf{x})$ in population P respectively. In case the difference between the minimal and maximal values is zero, set $\hat{f}(\mathbf{x}) = 1$. The range of $\hat{f}(\mathbf{x})$ is $[0, 1]$.

Let $\hat{v}(\mathbf{x})$ be a normalized value of $v(\mathbf{x})$:

$$\hat{v}(\mathbf{x}) = \frac{v(\mathbf{x}) - v_{\min}(P)}{v_{\max}(P) - v_{\min}(P)} \tag{9}$$

where $v_{\min}(P)$ and $v_{\max}(P)$ are the minimal and maximal values of $v(\mathbf{x})$ in a population P respectively. In case the difference between the minimal and maximal values is zero, set $\hat{v}(\mathbf{x}) = 1$. The range of $\hat{v}(\mathbf{x})$ is $[0, 1]$.

Similar to MOEA/D [31] and Tchebycheff method for multi-objective optimization, we construct $K+1$ objectives as follows. Let K be a positive integer and choose $K+1$ weights

$$w_i = \frac{i}{K}, \quad i = 0, 1, \cdots, K.$$

Then $K+1$ helper fitness functions are constructed in the form

$$f_i(\mathbf{x}) = w_{i-1}\hat{f}(\mathbf{x}) + (1 - w_{i-1})\hat{v}(\mathbf{x}),\ i = 1, \cdots, K+1, \tag{10}$$

In summary, the original constrained optimization problem is transferred into a $K+1$-objective optimization problem:

$$\min f_i(\mathbf{x}), \quad i = 1, \cdots, K+1. \tag{11}$$

If $K = 1$, it is equivalent to the standard model with two objectives. If $K \geq 2$, the model potentially may include many objectives inside. However, it sufficient to consider the simplest case of $K = 2$, which is a three-objective optimization problem:

$$\min \begin{cases} f_1(\mathbf{x}) = f(\mathbf{x}), \\ f_2(\mathbf{x}) = v(\mathbf{x}), \\ f_3(\mathbf{x}) = \frac{1}{2}\hat{f}(\mathbf{x}) + \frac{1}{2}\hat{v}(\mathbf{x}). \end{cases} \tag{12}$$

It is the combination of the standard two fitness functions and a helper objective which is the average of $\hat{f}$ and $\hat{v}$. Our ultimate aim is to find an optimal feasible solution through finding the Pareto front. It is obvious that the optimal feasible solution to the original COP (1) is on the Pareto front.

It must be mentioned that the above normalization is our new contribution, which is not used in MOEA/D [31]. Our experiments show that it plays an extremely important role in improving the performance. Without the normalization, the value of f sometimes is much larger than v and then it always dominates v. With the normalization, f and v play an equal role.

3 Multi-objective Differential Evolution for Constrained Optimisation

3.1 MOEAs

Many MOEAs have been proposed for solving MOPs. They can be classified into two categories: one aims to evolve the non-domination set and eventually to reach Pareto optimal set, such as the non-dominate sorting genetic algorithm [32] and strength Pareto evolutionary algorithm [33]. Another category focuses on solving a series of scalar optimization problems, such as the vector evaluated genetic algo-

rithm [34] and MOEAs based on decomposition [31]. The algorithm below gives a general description for the MOEAs based on the dominance relation.

1: initialize a set of solutions;
2: evaluate the values of $f_i, i = 1, \cdots, K+1$ for each solution;
3: select non-dominated solutions and construct an initial population P_0;
4: **for** $t = 0, 1, 2 \cdots, \cdots$ **do**
5: generate a children population C_t from the parent population P_t;
6: evaluate the values of $f_i, i = 1, \cdots, K+1$ for each solution;
7: select non-dominated solutions in $P_t \cup C_t$ and obtain the next generation population P_{t+1}.
8: **end for**

3.2 Constrained Multi-objective Differential Evolution

A MOEA based on differential evolution (DE), called CMODE [29], is chosen to solve the above MOP (11). Different from normal MOEAs, CMODE is specially designed for solving constrained optimization problems. Hence it is expected that CMODE is efficient in solving the MOP (11). CMODE [29] originally is applied to solving a bi-objective optimization problem which consists of only two functions: f_1 and f_2. However, it is easy to extend the existing CMODE to MOPs. The algorithm is described as follows.

Require: μ: population size;
λ: the number of individuals involved in DE operations
$FES_{\max}$: the maximum number of fitness evaluations
1: randomly generate an initial population P_0 with population size μ;
2: evaluate the values of f and v for each individual in the initial population, and then calculate the value of f_i where $i = 1, \cdots, m$;
3: set $FES = \mu$; // FES denotes the number of fitness evaluations;
4: set $A = \emptyset$; //A an archive to store the infeasible individual with the lowest degree of constraint violation;
5: **for** $t = 1, \cdots, FES_{\max}$ **do**
6: choose λ individuals (denoted by Q) from population P_t;
7: let $P' = P_t \setminus Q$;
8: for each individual in set Q, an offspring is generated by using DE mutation and crossover operations as explained in Section 3.3. Then λ children (denoted by C) are generated from Q;
9: evaluate the values of f and v for each individual in C and then obtain the value of f_i where $i = 1, \cdots, m$;
10: set $FES = FES + \lambda$;
11: identify all nondominated individuals in C (denoted by R);
12: **for** each individual $\mathbf{x}$ in R **do**
13: find all individual(s) in Q dominated by $\mathbf{x}$;
14: randomly replace one of these dominated individuals by $\mathbf{x}$;

15: **end for**
16: let $P_{t+1} = P' \cup Q$;
17: **if** no feasible solution exists in R **then**
18: identify the infeasible solution $\mathbf{x}$ in R with the lowest degree of constraint violation and add $\mathbf{x}$ to A;
19: **end if**
20: **if** $\mod(t,k) = 0$ **then**
21: execute the infeasible solution replacement mechanism and set $A = \emptyset$;
22: **end if**
23: **end for**
24: **return** the best found solution

The algorithm is explained step-by-step in the following. At the beginning, an initial population P_0 is chosen at random, where all initial vectors are chosen randomly from $[L_i, U_i]^n$.

At each generation, parent population P_t is split into two groups: one group with λ parent individuals that are used for DE operations (set Q) and the other group (set P') with $\mu - \lambda$ individuals that are not involved in DE operations. DE operations are applied to λ selected children (set Q) and then generate λ children (set C).

Selection is based on the dominance relation. First nondominated individuals (set R) are identified from children population C. Then these individual(s) will replace the dominated individuals in Q (if exists). As a result, population set Q is updated. Merge population set Q with those parent individuals that are involved in DE operation (set P') together and form the next parent population P_{t+1}. The procedure repeats until reaching the maximum number of evaluations. The output is the best found solution by DE.

The infeasible solution replacement mechanism is that, provided that a children population is composed of only infeasible individuals, the "best" child, who has the lowest degree of constraint violation, is stored into an archive. After a fixed interval of generations, some randomly selected infeasible individuals in the archive will replace the same number of randomly selected individuals in the parent population.

3.3 Differential Evolution

The mutation and crossover operators used in CMODE comes from DE. DE is arguably one of the most powerful stochastic real-parameter optimization algorithms in current use [35]. There exist several variants of DE. The original DE algorithm [36] is utilized in this paper. A population P_t is represented by μ n-dimensional vectors:

$$P_t = \{\mathbf{x}_{1,t}, \cdots, \mathbf{x}_{\mu,t}\}, \tag{13}$$

$$\mathbf{x}_{i,t} = (x_{i,1,t}, x_{i,2,t}, \cdots, x_{i,n,t}), i = 1, 2, \cdots, \mu, \tag{14}$$

where t represents the generation counter. Population size μ does not change during the minimization process. The initial vectors are chosen randomly from $[L_i, U_i]^n$. The formula below shows how to generate an initial individual $\mathbf{x} = (x_1, \cdots, x_n)$ at random:

$$x_i = L_i + (U_i - L_i) \times rand, \quad i = 1, \cdots, n, \tag{15}$$

where *rand* is the random number $[0, 1]$.

Three operations are used in the DE [36]: mutation, crossover and selection, which are described as follows.

- *Mutation:* for each target $\mathbf{x}_{i,t}$ where $i = 1, \cdots, n$, a mutant vector

  $$\mathbf{v}_{i,t} = (v_{i,1,t}, v_{i,2,t}, \cdots, v_{i,n,t})$$

  is generated by

  $$\mathbf{v}_{i,t} = \mathbf{x}_{r1,t} + F \cdot (\mathbf{x}_{r2,t} - \mathbf{x}_{r3,t}) \tag{16}$$

  where random indexes $r1, r2, r3 \in \{1, \cdots, \mu\}$ are mutually different integers. They are also chosen to be different from the running index i. F is a real and constant factor from $[0, 2]$ which controls the amplification of the differential variation $(\mathbf{x}_{r2,t} - \mathbf{x}_{r3,t})$. In case $\mathbf{v}_{i,t}$ is out of the interval $[L_i, U_i]$, the mutation operation is repeated until $\mathbf{v}_{i,t}$ falls in $[L_i, U_i]$.
- *Crossover:* in order to increase population diversity, crossover is also used in DE. The trial vector $\mathbf{u}_{i,t}$ is generated by mixing the target vector $\mathbf{x}_{i,t}$ with the mutant vector $\mathbf{v}_{i,t}$. Trial vector $\mathbf{u}_{i,t} = (u_{i,1,t}, u_{i,2,t}, \cdots, u_{i,n,t})$ is constructed as follows:

  $$u_{i,j,t} = \begin{cases} v_{i,j,t}, & \text{if } rand_j(1,0) \le C_r \text{ or } j = j_{rand}, \\ x_{i,j,t}, & \text{otherwise,} \end{cases} \quad j = 1, \cdots, n, \tag{17}$$

  where $rand_j(0, 1)$ is a uniform random number from $[0, 1]$. Index j_{rand} is randomly chosen from $\{1, \cdots, n\}$. $C_r \in [0, 1]$ denotes the crossover constant which has to be determined by the user. In addition, the condition "$j = j_{rand}$" is used to ensure the trial vector $\mathbf{u}_{i,t}$ gets at least one parameter from vector $\mathbf{v}_{i,t}$.
- *Selection:* a greedy criterion is used to decide which offspring generated by mutation and crossover should be selected to population P_{t+1}. Trail vector $\mathbf{u}_{i,t}$ is compared to target vector $\mathbf{x}_{i,t}$, then the better one will be reserved to the next generation.

4 Experiments and Results

4.1 Experimental Settings

Experiments are used to compare the performance of CMODE on the standard model with two objective and our new model with three objectives. The two-

objective optimization problem is

$$\min \begin{cases} f_1(\mathbf{x}) = f(\mathbf{x}), \\ f_2(\mathbf{x}) = v(\mathbf{x}), \end{cases} \tag{18}$$

This problem is the same as that in [29]. CMODE in the experiments of [29] is implemented using MATLAB language. We take experimental results (see Table 2 and Table 3) directly taken from [29].

The three-objective optimization problem is given as follows

$$\min \begin{cases} f_1(\mathbf{x}) = f(\mathbf{x}), \\ f_2(\mathbf{x}) = v(\mathbf{x}), \\ f_3(\mathbf{x}) = \frac{1}{2}\hat{f}(\mathbf{x}) + \frac{1}{2}\hat{v}(\mathbf{x}). \end{cases} \tag{19}$$

CMODE for three-objective optimization is is implemented using C++ language in our experiments. The C++ program of our work can be found in [40]. But the parameter setting of CMODE is the same as that in [29].

Thirteen benchmark functions were employed as the instances to perform experiments. These benchmarks have been used to test the performance of MOEAs for constrained optimization in [37] and are a part of benchmark collections in IEEE CEC 2006 special session on constrained real-parameter optimization [38]. Their detailed information is provided in Table 1, where n is the number of decision variables, LI stands for the number of linear inequalities constraints, NE the number of nonlinear equality constraints, NI nonlinear inequalities constraints. ρ denotes the ratio between the sizes of the entire search space and feasible search space and a is the number of active constraints at the optimal solution.

Table 1 Summary of 13 Benchmark Functions

Fcn	n	Type of f	ρ	LI	NE	NI	a	$f(\mathbf{x}^*)$
g01	13	quadratic	0.0003%	9	0	0	6	-15.0000000000
g02	20	nonlinear	99.9965%	1	0	1	1	-0.8036191041
g03	10	nonlinear	0.0000%	0	1	0	1	-1.0005001000
g04	5	quadratic	29.9356%	0	0	6	2	-30665.5386717833
g05	4	nonlinear	0.0000%	2	3	0	3	5126.4967140071
g06	2	nonlinear	0.0064%	0	0	2	2	-6961.8138755802
g07	10	quadratic	0.0003%	3	0	5	6	24.3062090682
g08	2	nonlinear	0.8640%	0	0	2	0	-0.0958250414
g09	7	nonlinear	0.5256%	0	0	4	2	680.6300573744
g10	8	linear	0.0005%	3	0	3	3	7049.2480205287
g11	2	quadratic	0.0000%	0	1	0	1	0.7499000000
g12	3	quadratic	0.0197%	0	0	9^3	0	-1.0000000000
g13	5	nonlinear	0.0000%	0	3	0	3	0.0539415140

CMODE contains several parameters which are the population size μ, the scaling factor F in mutation, the crossover control parameter Cr. Usually, F is set within $[0,1]$ and mostly from 0.5 to 0.9; Cr is also chosen from $[0,1]$ and higher values

can produce better results in most cases. In our experiments, F is randomly chosen between 0.5 and 0.6, Cr is randomly chosen from 0.9 to 0.95. Set $\lambda = 5$, and k=22. The population size $\mu = 180$. The tolerance value δ for the equality constraints was set to 0.0001. The maximum number of fitness evaluations $FES_{\max}$ is set to two values: $5 \cdot 10^4$ and $FES_{\max} = 5 \cdot 10^5$. As suggested in [38], 25 independent runs are set for each benchmark function.

4.2 Experimental Results

Table 2 and Table 3, taken from [29], shows the result of function error values achieved by CMODE with two helper functions f_1, f_2 on thirteen benchmark functions. Table 4 and Table 5 is our result of function error values achieved by CMODE using three helper functions f_1, f_2, f_3 on thirteen benchmark functions. Within $5 \cdot 10^4$ and $5 \cdot 10^5$ fitness evaluations, CMODE can produce very close to or even better than "known" optimal solutions by three helper functions f_1, f_2, f_3. The results obtained by CMODE with three helper functions f_1, f_2, f_3 within $5 \cdot 10^4$ are much more outstanding than which obtained by CMODE with two helper functions f_1, f_2 among all 13 benchmark functions apart from g10 and g13. Analogously, the results achieved by CMODE with three helper functions f_1, f_2, f_3 within $5 \cdot 10^5$ are much better in g03-10 and g13 whereas only worse in g02. Therefore CMODE using f_1, f_2, f_3 achieves remarkably better performance than that using f_1, f_2.

Table 2 Function Error Values Achieved by CMODE with two fitness functions f_1 and f_2 When $FES = 5 \times 10^4$, $FES = 5 \times 10^5$ for Test Functions g01-g13

FES	5×10^4				
	Best	Median	Worst	Mean	Std
g01	3.3935E-02	7.0387E-02	1.6472E-01	7.8005E-02	3.2951E-02
g02	1.6609E-01	2.0310E-01	2.6952E-01	2.0225E-01	2.7097E-02
g03	1.4327E-03	7.1021E-03	1.9600E-02	7.2294E-03	4.2851E-03
g04	1.4489E-03	1.3733E-02	4.4097E-02	1.7592E-02	1.1460E-02
g05	1.7750E-08	1.4309E-07	5.2276E-07	1.6482E-07	1.1939E-07
g06	2.1464E-08	2.4520E-07	1.4298E-06	4.4921E-07	4.2333E-07
g07	1.3917E-01	2.4984E-01	3.4206E-01	2.3893E-01	5.3937E-02
g08	8.1968E-11	1.1650E-08	2.8863E-07	3.4410E-08	6.3509E-08
g09	7.7915E-04	1.7339E-03	7.3459E-03	2.4239E-03	1.5793E-03
g10	1.7843E+01	2.9539E+01	5.1087E+01	3.0954E+01	7.1813E+00
g11	1.1792E-10	1.6769E-09	9.0309E-09	1.7298E-09	1.7427E-09
g12	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g13	8.3118E-09	3.8234E-08	1.8106E-07	6.0691E-08	5.5890E-08

In summary, our experimental results confirm that the performance of CMODE with three helper functions is better than that of that with only two helper functions in most benchmark functions.

Table 3 Function Error Values Achieved by CMODE with two fitness functions f_1 and f_2 When $FES = 5 \times 10^4$, $FES = 5 \times 10^5$ for Test Functions g01-g13

FES	5×10^5				
	Best	Median	Worst	Mean	Std
g01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g02	4.1726E-09	1.1372E-08	1.1836E-07	2.0387E-08	2.4195E-08
g03	2.3964E-10	1.1073E-09	2.5794E-09	1.1665E-09	5.2903E-10
g04	7.6398E-11	7.6398E-11	7.6398E-11	7.6398E-11	2.6382E-26
g05	1.8190E-12	1.8190E-12	1.8190E-12	1.8190E-12	1.2366E-27
g06	3.3651E-11	3.3651E-11	3.3651E-11	3.3651E-11	1.3191E-26
g07	7.9783E-11	7.9793E-11	7.9811E-11	7.9793E-11	7.6527E-15
g08	8.1964E-11	8.1964E-11	8.1964E-11	6.3596E-18	0.0000E+00
g09	9.8225E-11	9.8225E-11	9.8111E-11	9.8198E-11	4.9554E-14
g10	6.2755E-11	6.2755E-11	6.3664E-11	6.2827E-11	2.5182E-13
g11	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g12	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g13	4.1897E-11	4.1897E-11	4.1897E-11	3.6230E-15	0.0000E+00

Table 4 Function Error Values Achieved by CMODE with three fitness functions f_1, f_2 and f_3 When $FES = 5 \times 10^4$ for Test Functions g01-g13

FES	5×10^4				
	Best	Median	Worst	Mean	Std
g01	1.2837E-04	1.0211E-03	1.5566E-03	8.4569E-04	6.0161E-04
g02	9.0046E-03	1.9675E-02	9.0575E-02	3.8730E-02	2.9003E-02
g03	9.0574E-06	1.5975E-05	6.0624E-05	2.205E-05	1.526E-05
g04	3.3203E-06	1.5458E-05	5.7923E-05	2.5182E-05	2.0877E-05
g05	5.1929E-08	6.5710E-08	2.4226E-07	1.1910E-07	6.8873E-08
g06	-1.6371E-11	-4.5475E-12	9.5497E-11	3.1541E-11	4.5013E-11
g07	4.0450E-03	1.0116E-02	3.3372E-02	1.4698E-02	9.7322E-03
g08	2.7756E-17	5.5511E-17	1.7581E-12	2.8138E-13	6.4448E-13
g09	6.8854E-06	1.1737E-04	6.8578E-04	2.4306E-04	2.4293E-04
g10	1.3143E+01	3.8723E+01	7.9194E+01	3.8438E+01	2.2326E+01
g11	4.2189E-15	4.8739E-14	3.3029E-13	8.7055E-14	1.1019E-13
g12	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g13	1.3433E-09	6.8162E-09	3.6096E-07	3.5494E-08	9.6116E-08

5 Conclusions

This paper proposes a new multi-objective model for solving constrained optimization problems. Besides the standard model with two objectives: to minimize the original objective function and the sum of degrees of constraint violation, other helper fitness functions are constructed from weighted sums of the normalized original objective and the normalized degree of constraint violation.

The new model is compareed with the standard model using the same CMODE for solving MOPs. Experimental results show that CMODE with three fitness func-

Table 5 Function Error Values Achieved by CMODE with three fitness functions f_1, f_2 and f_3 When $FES = 5 \times 10^5$ for Test Functions g01-g13

FES	5×10^5				
	Best	Median	Worst	Mean	Std
g01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g02	1.4301E-08	2.5307E-02	6.9972E-02	2.5188E-02	1.8275E-02
g03	4.4936E-13	3.5689E-12	2.3674E-11	5.2828E-12	5.0102E-12
g04	-7.2760E-12	-7.2760E-12	-7.2760E-12	-7.2760E-12	0.0000E+00
g05	-1.8190E-12	-1.8190E-12	-1.8190E-12	-1.8190E-12	0.0000E+00
g06	-1.6371E-11	-1.6371E-11	-1.6371E-11	-1.6371E-11	0.0000E+00
g07	-1.4566E-13	8.6366E-12	7.3598E-09	7.4716E-10	1.8301E-09
g08	2.7756E-17	2.7756E-17	2.7756E-17	2.7756E-17	0.0000E+00
g09	-1.1369E-13	-1.1369E-13	-1.1369E-13	-1.1369E-13	0.0000E+00
g10	-5.4570E-12	-3.6380E-12	5.0022E-11	-6.1846E-13	1.2234E-11
g11	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g12	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
g13	-1.9429E-16	-1.9429E-16	-1.9429E-16	-1.9429E-16	0.0000E+00

tions obtains remarkable better performance than that with the standard two fitness functions [29] on most benchmark functions (12/13). This confirms our expectation that adding more helper functions may significantly improve the performance of MOEAs for COPs. The new model is extremely encouraging since our method is able to compete with other leading methods [37, 13, 39, 29]. Our next step is to test it on more benchmarks functions and to make a full size comparison with other leading methods.

Acknowledgement This work was partially supported by EPSRC under Grant No. EP/I009809/1.

References

1. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation **4**(1) (1996) 1–32
2. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering **191**(11-12) (2002) 1245–1287
3. Mezura-Montes, E., Coello Coello, C.A.: Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm and Evolutionary Computation **1**(4) (2011) 173–194
4. Segura, C., Coello, C.A.C., Miranda, G., León, C.: Using multi-objective evolutionary algorithms for single-objective optimization. 4OR **11**(3) (2013) 201–228
5. Louis, S.J., Rawlins, G.: Pareto optimality, GA-easiness and deception. In: Proceedings of 5th International Conference on Genetic Algorithms, Morgan Kaufmann (1993) 118–123
6. Knowles, J.D., Watson, R.A., Corne, D.W.: Reducing local optima in single-objective problems by multi-objectivization. In: Evolutionary Multi-Criterion Optimization, Springer (2001) 269–283

7. Mezura-Montes, E., Coello, C.A.C.: Constrained optimization via multiobjective evolutionary algorithms. In Knowles, J., Corne, D., Deb, K., Chair, D., eds.: Multiobjective Problem Solving from Nature. Springer Berlin Heidelberg (2008) 53–75
8. Surry, P.D., Radcliffe, N.J.: The COMOGA method: constrained optimization by multiobjective genetic algorithms. Control and Cybernetics **26** (1997) 391–412
9. Zhou, Y., Li, Y., He, J., Kang, L.: Multi-objective and MGG evolutionary algorithm for constrained optimization. In: Proceedings of 2003 IEEE Congress on Evolutionary Computation, Canberra, Australia, IEEE Press (2003) 1–5
10. Wang, Y., Liu, D., Cheung, Y.M.: Preference bi-objective evolutionary algorithm for constrained optimization. In: Computational Intelligence and Security. Springer (2005) 184–191
11. Venkatraman, S., Yen, G.G.: A generic framework for constrained optimization using genetic algorithms. IEEE Transactions on Evolutionary Computation **9**(4) (2005) 424–435
12. Deb, K., Lele, S., Datta, R.: A hybrid evolutionary multi-objective and SQP based procedure for constrained optimization. In Kang, L., Liu, Y., Zeng, S., eds.: Advances in Computation and Intelligence. Springer (2007) 36–45
13. Wang, Y., Cai, Z., Guo, G., Zhou, Y.: Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Transactions on Systems, Man, and Cybernetics, Part B **37**(3) (2007) 560–575
14. Ray, T., Singh, H., Isaacs, A., Smith, W.: Infeasibility driven evolutionary algorithm for constrained optimization. In Mezura-Montes, E., ed.: Constraint-Handling in Evolutionary Optimization. Volume 198. Springer Berlin Heidelberg (2009) 145–165
15. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach. IEEE Transactions on Evolutionary Computation **18**(4) (2014) 602–622
16. Zapotecas Martinez, S., Coello Coello, C.: A multi-objective evolutionary algorithm based on decomposition for constrained multi-objective optimization. In: Proceedings of 2014 IEEE Congress on Evolutionary Computation, IEEE (2014) 429–436
17. Gao, W.F., Yen, G.G., Liu, S.Y.: A dual-population differential evolution with coevolution for constrained optimization. IEEE Transactions on Cybernetics **45**(5) (2015) 1094–1107
18. Coello, C.A.C., Mezura-Montes, E.: Handling constraints in genetic algorithms using dominance-based tournaments. In: Adaptive Computing in Design and Manufacture V. Springer (2002) 273–284
19. Jiménez, F., Gómez-Skarmeta, A.F., Sánchez, G.: How evolutionary multiobjective optimization can be used for goals and priorities based optimization. In: Primer Congreso Espanol de Algoritmos Evolutivos y Bioinspirados (AEB'02), Mérida, Espana, Universidad de Extremadura (2002) 460–465
20. Kukkonen, S., Lampinen, J.: Constrained real-parameter optimization with generalized differential evolution. In: Proceedings of 2006 IEEE Congress on Evolutionary Computation, IEEE (2006) 207–214
21. Gong, W., Cai, Z.: A multiobjective differential evolution algorithm for constrained optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, IEEE (2008) 181–188
22. Ray, T., Kang, T., Chye, S.K.: An evolutionary algorithm for constrained optimization. In: Proceedings of 2000 Genetic and Evolutionary Computation Conference, San Francisco, Morgan Kaufmann (2000) 771–777
23. Aguirre, A.H., Rionda, S.B., Coello Coello, C.A., Lizárraga, G.L., Montes, E.M.: Handling constraints using multiobjective optimization concepts. International Journal for Numerical Methods in Engineering **59**(15) (2004) 1989–2017
24. Liang, J.J., Suganthan, P.: Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In: Proceedings of 2006 IEEE Congress on Evolutionary Computation, IEEE (2006) 9–16
25. Watanabe, S., Sakakibara, K.: Multi-objective approaches in a single-objective optimization environment. In: Proceedings of 2005 IEEE Congress on Evolutionary Computation. Volume 2., IEEE (2005) 1714–1721

26. Reynoso-Meza, G., Blasco, X., Sanchis, J., Martinez, M.: Multiobjective optimization algorithm for solving constrained single objective problems. In: Proceedings of 2010 IEEE Congress on Evolutionary Computation. (July 2010) 1–7
27. Chowdhury, S., Dulikravich, G.S.: Improvements to single-objective constrained predator–prey evolutionary optimization algorithm. Structural and Multidisciplinary Optimization **41**(4) (2010) 541–554
28. Jia, L., Zeng, S., Zhou, D., Zhou, A., Li, Z., Jing, H.: Dynamic multi-objective differential evolution for solving constrained optimization problem. In: Evolutionary Computation (CEC), 2011 IEEE Congress on. (June 2011) 2649–2654
29. Wang, Y., Cai, Z.: Combining multiobjective optimization with differential evolution to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation **16**(1) (2012) 117–134
30. Xu, T., He., J.H.: Multi-objective differential evolution with helper functions for constrained optimization. In: Proceedings of UKCI 2015. (2015) (accepted).
31. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6) (2007) 712–731
32. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197
33. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation **3**(4) (1999) 257–271
34. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985. (1985) 93–100
35. Das, S., Suganthan, P.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation **15**(1) (Feb 2011) 4–31
36. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4) (1997) 341–359
37. Cai, Z., Wang, Y.: A multiobjective optimization-based evolutionary algorithm for constrained optimization. IEEE Transactions on Evolutionary Computation **10**(6) (2006) 658–675
38. Liang, J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C.C., Deb, K.: Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University (2006)
39. Mallipeddi, R., Suganthan, P.N.: Ensemble of constraint handling techniques. IEEE Transactions on Evolutionary Computation **14**(4) (2010) 561–579
40. Tao Xu, Weiqin Ying: newSMODE. `https://drive.google.com/file/d/0B57WgWIwWDmkS1d4Z0Y5RzhZWlU/view?usp=sharing`. (2016)