Combining Dynamic Constrained Many-Objective Optimization with DE to Solve Constrained Optimization Problems

Xi Li[1,2], Sanyou Zeng[1(✉)], Liting Zhang[1], and Guilin Zhang[1]

[1] School of Computer Science, China University of Geosciences, Wuhan 430074, Hubei, People's Republic of China
1589441554@qq.com, sanyouzeng@gmail.com

[2] School of Information Engineering, Shijiazhuang University of Economics, Shijiazhuang 050031, Hebei, People's Republic of China

Abstract. This paper proposes a dynamic constrained many-objective optimization method for solving constrained optimization problems. We first convert a constrained optimization problem (COP) into an equivalent dynamic constrained many-objective optimization problem (DCMOP), then present many-objective optimization evolutionary algorithm with dynamic constraint handling mechanism, called MaDC, to solve the DCMOP, thus the COP is addressed. MaDC uses DE as the search engine, and reference-point-based nondominated sorting approach to select individuals to construct next population. The effectiveness of MaDC has been verified by comparing with peer algorithms.

Keywords: Constrained optimization problem · Many-objective optimization · Dynamic constraint · DE · Reference points

1 Introduction

In science and engineering disciplines, it is common to encounter a large number of constrained optimization problems (COPs). During the past decades, researchers have widely used evolutionary algorithms (EAs) to deal with COPs [1–3], and made considerable achievements. In recent years, with the development of the multi-objective and adaptive evolutionary theories and methodologies, more and more works are managed to add these fruits to solving constrained problems.

Coello first used dominance-based selection strategy to deal with constraints [4]. In [5] Coello and Mezura proposed a new version of the Niched-Pareto Genetic Algorithm (NPGA). This approach uses dominance-based selection scheme to assign fitness function value, and adopts an additional parameter called S_r to control the diversity of the population. Venkatraman and Yen [6] proposed genetic algorithm-based two-phase framework for solving COPs. In the first phase the objective function is completely disregarded, and only the constraints of the problem are focused on. In the second phase, the objective

K. Li et al. (Eds.): ISICA 2015, CCIS 575, pp. 64–73, 2016.
DOI: 10.1007/978-981-10-0356-1_7

function and satisfaction of the constraints are treated as two objectives to be simultaneously optimized. Hsieh [7] proposed an algorithm based on well-known multi-objective evolutionary algorithm, NSGA-II. The procedure, used as a hybrid constraint handling mechanism, combines ϵ-comparison method of multi-objective optimization and penalty method of constraints-handling. Yong Wang [8] presented hybrid constrained optimization EA (HCOEA), which effectively combines multi-objective optimization with global and local search models. In global model, Pareto-dominance-based tournament selection among parent and offspring and similarity measuring by Euclidean distance among individuals are used to promote population diversity; in the local model, a parallel search in subpopulations is implemented to accelerate convergence. Penalty function is a classical method used for solving COP, but the determination of penalty parameters is a difficulty. Deb [9] proposed a hybrid algorithm which combines bi-objective evolutionary approach with the penalty function methodology. The bi-objective approach provides a good estimate of the penalty parameter, and the unconstrained penalty function approach being constructed using provided penalty parameter generates the optimal solutions of overall hybrid algorithm. Zeng and Li [10,11] used not only multi-objective optimization technology but also dynamic constraint mechanism for COPs. They first convert COP to a dynamic constrained multi-objective optimization problem, then adopt a dynamic constrained multi-objective optimization algorithm to solve the problem.

In this paper, we convert the COP into the many-objective optimization problem, there are m+1 objectives (m is the number of constraints) in a problem, in other words, each constraint function is converted into a violation objective function. So we can introduced many-objective optimization technique into our method. Besides, we adopt dynamic constraint handling mechanism to deal with constraints. The proposed many-objective optimization evolutionary algorithm with dynamic constrained handling, MaDC, uses DE to generate offspring and reference-point-based nondominated sorting approach to create next parent population.

The rest of this paper is organized as follows. Section 2 introduces process of converting a COP into an equivalent dynamic constrained many-objective optimization problem (DCMOP). Section 3 describes implementation of MaDC algorithm. Experiments and results are shown in Sect. 4 to test whether the methodology is effective. Section 5 gives the conclusion.

2 Convert COP to DCMOP

This section first converts a COP into a constrained many-objective optimization problem (CMOP) which is equivalent to the COP. Then the CMOP is converted into a dynamic constrained many-objective optimization problem (DCMOP), a series of CMOPs. In this way, the COP can be solved by solving the equivalent DCMOP.

2.1 Convert COP to CMOP

Without loss of generality, minimization optimization is assumed unless specified otherwise in this paper. A constrained optimization problem (COP) can be stated as follows:

$$\begin{aligned} &min \quad y = f(\boldsymbol{x}) \\ &st: \quad \boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), ..., g_m(\boldsymbol{x})) \leq \mathbf{0} \\ &where\ \boldsymbol{x} = (x_1, x_2, ..., x_n) \in \mathbf{X} \\ &\quad\ \ \mathbf{X} = \{\boldsymbol{x} | \boldsymbol{l} \leq \boldsymbol{x} \leq \ \boldsymbol{u}\} \\ &\quad\ \ \boldsymbol{l} = (l_1, l_2, ..., l_n), \boldsymbol{u} = (u_1, u_2, ..., u_n) \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is the solution vector and $\mathbf{X}$ is the whole search space, $\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower bound and upper bound of the solution space, respectively, $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$ is the constraint and $\mathbf{0}$ is the constrained boundary. When an equality constraint $h(\boldsymbol{x}) = 0$ is involved in the COP, it is usually transformed into an inequality constraint $|h(\boldsymbol{x})| - \epsilon \leq 0$, ϵ is a positive close-to-zero number, $\epsilon = 0.0001$ in this paper.

A solution $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ is **feasible** if it satisfies the constraints conditions $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$, otherwise it is **infeasible**. A feasible set $\mathbf{S_F}$ of a COP is defined as $\mathbf{S_F} = \{\boldsymbol{x} | \boldsymbol{x} \in \mathbf{X}\ and\ \boldsymbol{x}\ is feasible\}$.

Now we would like to construct a constrained many-objective optimization problem equivalent to the COP discussed above. This can be implemented by converting the constraint function $\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), ..., g_m(\boldsymbol{x}))$ to violation objective function $\boldsymbol{\varphi}(\boldsymbol{x}) = (\varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}), ..., \varphi_m(\boldsymbol{x}))$ and inserting $\boldsymbol{\varphi}(\boldsymbol{x})$ into the COP as additional objectives without deleting the constraints $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$, i.e., a constrained many-objective optimization problem (CMOP) is constructed as follow:

$$\begin{aligned} &min\ \boldsymbol{y} = (f(\boldsymbol{x}), \varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}), ..., \varphi_m(\boldsymbol{x})) \\ &st: \ \ \boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), ..., g_m(\boldsymbol{x})) \leq \mathbf{0} \end{aligned} \tag{2}$$

where $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ is n dimension search vector, $\boldsymbol{y}$, $\boldsymbol{g}$ are functions of vector $\boldsymbol{x}$, $\varphi_i(\boldsymbol{x})$ $(i = 1, 2, \cdots, m)$ is a violation objective function converted from $g_i(\boldsymbol{x})$, the conversion is stated as:

$$\varphi_i(\boldsymbol{x}) = max\{g_i(\boldsymbol{x}), 0\}, i = 1, 2, \cdots, m \tag{3}$$

so the COP is transformed CMOP with m+1 evolution objectives and m constraint conditions.

Obviously, the CMOP in Eq. (2) has the same feasible set and the same optimal solutions as the COP in Eq. (1). Then the CMOP is equivalent to the COP, and therefore, we could solve the COP by the way of solving the CMOP by using a constrained many-objective optimization algorithm.

In multi-objective optimization, Pareto dominance is an essential relation in comparing two solution individuals. Given two solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\boldsymbol{x}_1$ is called **Pareto dominates** $\boldsymbol{x}_2$ if and only if $f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2)$ for every objective index i, and $f_j(\boldsymbol{x}_1) < f_j(\boldsymbol{x}_2)$ for at least one index j. A solution $\boldsymbol{x}^*$ is **Pareto optimal** (**non-dominated**) solution if there is no solution $\boldsymbol{x}$ such that $f(\boldsymbol{x})$ Pareto dominates $f(\boldsymbol{x}^*)$.

2.2 Convert CMOP to DCMOP

Many-objective evolutionary algorithm (MOEA) in solving CMOP will face the same difficulty of handling constraints as that of EA in solving COP. We know that multi-objective evolutionary algorithm in solving a multi-objective optimization problem (MOP) without constraints performs very well, if we can make the CMOP look MOP without constraints and use MOEA to overcome, we will obtain the optimal resolution. So, the key issue is to achieve a feasible population all the time, which can be addressed by adopting dynamic constraint handling technique.

First, the original constrained boundary $\mathbf{0}$ of the CMOP in Eq. (2) is largely broadened to $\boldsymbol{e}^{(0)}$ at the beginning. Then the broadened boundary $\boldsymbol{e}^{(0)}$ shrinks gradually back to $\mathbf{0}$. Each change of boundary is small enough so that the whole population is always near feasible.

This process constructs a sequence of CMOPs $\{CMOP^{(s)}\}, s = 0, 1, 2, \cdots, S$, i.e., a dynamic constrained many-objective optimization problem (DCMOP) as follows:

$$\begin{aligned}
&COMP^0 \begin{cases} min\ \boldsymbol{y} = (f(\boldsymbol{x}), \varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}), ..., \varphi_m(\boldsymbol{x})) \\ st : \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{e}^{(0)} \end{cases} \\
&COMP^1 \begin{cases} min\ \boldsymbol{y} = (f(\boldsymbol{x}), \varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}), ..., \varphi_m(\boldsymbol{x})) \\ st : \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{e}^{(1)} \end{cases} \\
&..... \\
&COMP^S \begin{cases} min\ \boldsymbol{y} = (f(\boldsymbol{x}), \varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}), ..., \varphi_m(\boldsymbol{x})) \\ st : \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{e}^{(S)} = \mathbf{0} \end{cases}
\end{aligned} \tag{4}$$

where $\boldsymbol{e}^{(s)} = (e_1^{(s)}, e_2^{(s)}, ..., e_m^{(s)}), s \in \{0, 1, 2, \cdots, S\}$, $\boldsymbol{e}^{(0)} \geq \boldsymbol{e}^{(1)} \geq \cdots \geq \boldsymbol{e}^{(S)} = \mathbf{0}$.

$\boldsymbol{e}^{(s)}$ is called **elastic constrained boundary**, and s is called **environment state**.

The initial boundary $\boldsymbol{e}^{(0)}$ on the initial state $s = 0$ needs to enable initial population $\mathbf{P}(0)$ feasible, It is set as $e_i^{(0)} = \max\limits_{\boldsymbol{x} \in \mathbf{P}(0)} \{g_i(\boldsymbol{x})\}$, $g_i(\boldsymbol{x})$ is the function value of the ith constraint, $i = 1, 2, ..., m$. On the final state $s = S$, the boundary goes back to $\mathbf{0}$, i.e., $\boldsymbol{e}^{(S)} = \mathbf{0}$. the boundary change on every environment state is modelled as follow:

$$e_i^{(s)} = A_i e^{-(\frac{s}{B_i})^2} - \varepsilon, i = 1, 2, \cdots, m \tag{5}$$

Regarding to elastic constrained boundary, if a solution satisfies inequality $\boldsymbol{g} \leq \boldsymbol{x}$, it is said to be ***e*-feasible**, otherwise, it is said to be ***e*-infeasible**. Obviously, a feasible solution is e-feasible, while an e-feasible solution might by infeasible or feasible.

Pareto-domination is defined without considering the constraints, see Subsect. 2.2. An e-constrained Pareto-domination for the DCMOP Eq. (4) is stated as follows:

Given two solutions:

- if both are e-feasible, the one which dominates the other at all objectives (involve the original objective and the violation objectives) wins;
- if one is e-feasible and the other is e-infeasible, the e-feasible solution wins;
- if both are e-infeasible, the one which dominates the other at violation objectives wins.

3 Algorithm Description

This section gives the implementation of many-objective optimization algorithm with dynamic constraints (MaDC) for solving DCMOP.

Algorithm 1. Framework of MaDC

step 1 : Initiation

1.1 Initialize parent population $\mathbf{P}(0) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_N\}$. Set global generation counter $t = 0$.

1.2 Initialize elastic constrained boundary $\boldsymbol{e} = \boldsymbol{e}^{(0)}$. Set environment state $s = 0$.

1.3 Determine reference points $\mathbf{Z}$.

step 2 : Change state

IF population is $\boldsymbol{e}$-feasible THEN reduce boundary $\boldsymbol{e} = \boldsymbol{e}^{(s+1)}$, $s = s + 1$.

step 3 : Generate offspring population

Use DE to generate offspring population $\mathbf{S}(t)$ from $\mathbf{P}(t)$ and evaluate $\mathbf{S}(t)$.

step 4 : Generate next population

Use reference-point-based nondominated sorting approach to select individuals from combined $\mathbf{S}(t)$ and $\mathbf{P}(t)$ to create next population $\mathbf{P}(t + 1)$.

step 5 : $t = t + 1$, IF s achieves final state S or t achieves $MaxG$, THEN goto *Step 6*, ELSE goto *Step2*.

step 6 : Output results.

MaDC use DE to generate offspring, and reference-point-based nondominated sorting approach to create next population. Reference-point-based nondominated sorting approach is proposed in literature [12], it is an evolutionary many- objective optimization technique of combining nondominated sorting and reference-point-based selection strategy.

Note if the algorithm could not evolve to achieve an e-feasible population on a certain state s, then it would iterate infinitely on this state. A maximal run generation $MaxG$ is set to abort the run.

The generation of offspring population in step 3 of Algorithm 1 is a combination of some genetic operators: affine mutation, crossover and uniform mutation. The detail of genetic operators is as Algorithm 2:

Algorithm 2. Generate offspring procedure

input : $\mathbf{P}(t)$, F, CR, P_m
output : Offspring population $\mathbf{S}(t)$
step 1 : $\mathbf{S}(t) = \Phi$.
step 2 : For every individual $\boldsymbol{x}_i \in \mathbf{P}(t), i = 1, 2, 3, ..., N$ do:
2.1 Affine Mutation:
$\boldsymbol{v}_i = \boldsymbol{x}_a + F(\boldsymbol{x}_b - \boldsymbol{x}_c)$
$\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}_c \in \mathbf{P}(t), a \neq b \neq c \neq i$, are selected randomly three individuals.
2.2 Crossover on $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ and $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \cdots, v_{in})$:

$$u_{ij} = \begin{cases} v_{ij} & if\ r_{rnd} < CR\ or\ j = j_{rnd} \\ x_{ij} & if\ r_{rnd} \geq CR\ or\ j \neq j_{rnd} \end{cases}$$
$$j = 1, 2, ..., n$$
$$j_{rnd} = rndInt(1, n), r_{rnd} = rndReal(0, 1)$$

2.3 Uniform Mutation on $\boldsymbol{u}_i = (u_{i1}, u_{i2}, \cdots, u_{in})$:
Change $u_{ij} = rndReal(0, 1)$ with probability P_m for $j = 1, 2, ..., n$.
step 3 : Add each $\boldsymbol{u}_i (i = 1, 2, 3, ..., N)$ into $\mathbf{S}(t)$.
step 4 : Return $\mathbf{S}(t)$.

Algorithm 3 was given as the details of creating next parent population. It uses reference-point-based nondominated sorting method to select individuals to construct the next population.

Algorithm 3. Generate next population

input : $\mathbf{Q}(t)$, $\mathbf{Z}$, N
output : $\mathbf{N}(t)$
step 1 : $\mathbf{N}(t) = \Phi$. $(\mathbf{F}_1, \mathbf{F}_2, \cdots) =$ Non-dominated-sort($\mathbf{Q}(t)$). $i = 1$.
step 2 : IF $|\mathbf{N}(t)| + |\mathbf{F}_i| < N$, THEN $\mathbf{N}(t) = \mathbf{N}(t) \cup \mathbf{F}_i$, $i = i + 1$, goto *Step 2*;
IF $|\mathbf{N}(t)| + |\mathbf{F}_i| = N$, THEN $\mathbf{N}(t) = \mathbf{N}(t) \cup \mathbf{F}_i$, goto *Step 4*;
IF $|\mathbf{N}(t)| + |\mathbf{F}_i| > N$, THEN goto *Step 3*.
step 3 : Select $N - |\mathbf{N}(t)|$ individuals from $\mathbf{F}_i$ and add them into $\mathbf{N}(t)$:
3.1 Associate each solution of $\mathbf{N}(t)$ with closest reference point by the perpendicular distance, compute the niche count of each reference point.
3.2 Select randomly a point $\mathbf{r}$ which have smallest niche count.
3.3 Let $\mathbf{I_r}$ be a set of individuals associated with $\mathbf{r}$, $\mathbf{I_r} \subseteq \mathbf{F}_i$. IF $|\mathbf{I_r}| = \Phi$, THEN remove $\mathbf{r}$ from $\mathbf{Z}$ temporarily at this generation, goto *3.2*;
ELSE: IF nichecount($\mathbf{r}$) = 0, THEN select the member $\boldsymbol{s}$ which has smallest perpendicular distance to $\mathbf{r}$; IF nichecount($\mathbf{r}$) $\neq$ 0, THEN select a member $\boldsymbol{s}$ randomly from $\mathbf{I_r}$
3.4 Add the selected member $\boldsymbol{s}$ into $\mathbf{N}(t)$, nichecount($\mathbf{r}$)= nichecount($\mathbf{r}$)+1, $\mathbf{F}_i = \mathbf{F}_i \setminus s$.
3.5 Repeat the selection above, until all $N - |\mathbf{N}(t)|$ individuals are chosen.
step 4 : Return $\mathbf{N}(t)$.

4 Experiments and Results

In this section, we apply our proposed methodology to a number of benchmark problems proposed in Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization [13], online available: http://www.ntu.edu.sg/home/epnsugan/. The 24 test instances are minimization problems. The detail of the test problems refers to [13].

4.1 Determination of Reference Points and Algorithm Parameters

The proposed algorithm uses a predefined set of reference points to ensure diversity of many-objective optimization. We use determination method presented in [12] that places points on a normalized hyper-planean $(M-1)$-dimensional unit simplex—which is equally inclined to all objective axes and has an intercept of one on each axis. If p divisions are considered along each objective, the total number of reference points (H) in an M-objective problem is given by:

$$H = C^{p}_{M+p-1} \tag{6}$$

For example, in a three-objective problem $(M = 3)$, if six divisions $(p = 6)$ are chosen for each objective axis, $H = 28$ reference points will be created [12]. When there are many objectives $(M \geq 5)$, one layer of reference points is not appropriate. For eight-objective problems, even if we use $p = 8$ (to have exactly one intermediate reference point), it requires 5040 reference points. To avoid such a situation, we use two layers of reference points (boundary layer and inside layer) in many-objective problems.

The population size N is set the number of reference points. Table 1 shows the number of chosen reference points (H) and corresponding population sizes.

Table 1. Number of reference points and corresponding population size

M	p of boun.	p of insi.	H	popsize	M	p of boun.	p of insi.	H	popsize
2	90	0	91	91	8	2	2	72	72
3	12	0	91	91	9	2	2	90	90
4	6	0	84	84	10	2	2	110	110
5	4	2	85	85	14	2	1	119	119
6	3	2	77	77	39	1	1	78	78
7	3	1	91	91	–	–	–	–	–

Other parameters are as follow:

Number of repeats: 25.

In the offspring generation procedure (Algorithm 2), the scaling factor $F = 0.5$, crossover rate $CR = 0.9$, uniform mutation probability $P_m = 0.01$.

Table 2. Function values obtained by MADC, SAMO-DE, ECHT-EP2, DE-DPS, HCOEA and DCMOEA

Pro.	Crit.	MaDC	SAMO-EA	ECHT-EP2	DE-DPS	HCOEA	DCMOEA
g01	best	**−15.0000**	−15.0000	−15.0000	−15.0000	−15.0000	−15.0000
	Avg.	**−15.0000**	−15.0000	−15.0000	−15.0000	−15.0000	−15.0000
g02	best	**−0.8036191**	−0.8036191	−0.8036191	−0.8036190	−0.803241	−0.8036191
	Avg.	−0.8010908	−0.7987352	−0.7998220	−0.8036189	−0.801258	−0.7969470
g03	best	**−1.0005**	−1.0005	−1.0005	−1.0005	−1.0005	−1.0005
	Avg.	**−1.0005**	−1.0005	−1.0005	−1.0005	−1.0005	−1.0005
g04	best	**−30665.5386**	−30665.5386	−30665.5386	−30665.5386	−30665.5386	−30665.5386
	Avg.	−30665.5386	−30665.5386	−30665.5386	−30665.5386	−30665.5386	−30665.5386
g05	best	**5126.4967**	5126.497	5126.497	5126.497	5126.498	5126.498
	Avg.	**5126.4967**	5126.497	5126.497	5126.497	5148.960	5126.498
g06	best	**−6961.8138**	−6961.8138	−6961.8138	−6961.8138	−6961.8138	−6961.8138
	Avg.	**−6961.8138**	−6961.8138	−6961.8138	−6961.8126	−6961.8138	−6961.8138
g07	best	**24.3062**	24.3062	24.3062	24.3062	24.3062	24.3062
	Avg.	24.3063	24.3096	24.3063	24.3062	24.307	24.3064
g08	best	**−0.095825**	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825
	Avg.	**−0.095825**	−0.095825	−0.095825	−0.095825	−0.095825	−0.093491
g09	best	**680.630**	680.630	680.630	680.630	680.630	680.630
	Avg.	**680.630**	680.630	680.630	680.630	680.630	680.630
g10	best	**7049.249**	7049.249	7049.249	7049.248	7049.287	7049.248
	Avg.	7049.304	7059.813	7049.249	7059.248	7049.525	7049.248
g11	best	**0.7499**	0.7499	0.7499	0.7499	0.750	0.75
	Avg.	**0.7499**	0.7499	0.7499	0.7499	0.750	0.75
g12	best	**−1.000**	−1.000	−1.000	−1.000	−1.000	−1.000
	Avg.	**−1.000**	−1.000	−1.000	−1.000	−1.000	−1.000
g13	best	**0.05394**	0.05394	0.05394	0.05394	0.05395	0.05395
	Avg.	**0.05394**	0.05394	0.05394	0.81702	0.05395	0.05395
g14	best	**−47.76488**	−47.76488	−47.7649	−47.76488	–	–
	Avg.	−47.76395	−47.68115	−47.7648	−47.76488	–	–
g15	best	**961.71502**	961.71502	961.71502	961.71502	–	–
	Avg.	**961.71502**	961.71502	961.71502	962.13142	–	–
g16	best	**−1.905155**	−1.905155	−1.905155	−1.905155	–	–
	Avg.	**−1.905155**	−1.905155	−1.905155	−1.905155	–	–
g17	best	**8853.5338**	8853.5397	8853.5397	8862.6287	–	–
	Avg.	**8853.5338**	8853.5397	8853.5397	8934.8675	–	–
g18	best	**−0.866025**	−0.866025	−0.866025	−0.866025	–	–
	Avg.	−0.866024	−0.866024	−0.866025	−0.866025	–	–
g19	best	**32.65559**	32.65559	32.6591	32.65559	–	–
	Avg.	32.65564	32.75734	32.6623	32.65559	–	–
g21	best	**193.72451**	193.72451	193.7246	193.72451	–	–
	Avg.	**193.72451**	193.77137	193.7348	193.72451	–	–
g23	best	−400.0451	−396.1657	−398.9731	−400.0551	–	–
	Avg.	−395.8492	−360.8176	−373.2178	−395.6745	–	–
g24	best	**−5.508013**	−5.508013	−5.508013	−5.508013	–	–
	Avg.	**−5.508013**	−5.508013	−5.508013	−5.508013	–	–

The ε in Eq. 5 was set to 0.000 000 1.

The number of environment changes was set $S = 240000/N$.

The maximal run generation $MaxG = 10000$, if a problem has no feasible solutions or the algorithm could not find feasible solutions, the algorithm would

abort after evolving 10 000 generations. Problems g20 and g22 could not find feasible solution.

4.2 Results and Comparison

The detailed results of MaDC are provided in Table 2, along with that of the state-of-the-art algorithms such as: (1) self-adaptive multioperator differential evolution (SAMO-DE) [14]; (2) ensemble of constraint handling techniques based on evolutionary programming (ECHT-EP2) [15]; (3) differential evolution with dynamic parameters selection (DE-DPS) [3]; (4) hybrid constrained optimization evolutionary algorithm (HCOEA) [8]; (5) dynamic constrained multi-objective evolutionary algorithm (DCMOEA) [10]. All algorithms solved 22 test problems, except HCOEA and DCMOEA, in which only the 13 test problems were solved.

From Table 2, MaDC was able to obtain the optimal solutions for all problems except g23. The algorithm SAMO-DE, ECHT-EP2, DE-DPS were able to obtain the optimal solutions for 20, 19, 20 problems, respectively. The algorithm HCOEA and DEMOEA obtained the optimal solutions for 9, 12 out of 13 problems. In regard to the average results, MaDC is superior to SAMO-DE, ECHT-EP2, DE-DPS, HCOEA and DEMOEA for eight, four, three, three, two text problems, respectively. It can be seen that our proposed method performs better than or is competitive to state-of-the-art algorithms.

5 Conclusion

In this paper, we have suggested a many-objective optimization algorithm with dynamic constraint mechanism for solving constrained optimization problem. We first construct an equivalent dynamic constrained many-objective optimization problem to the COP, then adopt MaDC algorithm to solve the DCMOP, thus the COP is solved. Dynamic technology is implemented by setting an elastic boundary for the constrained problem, and the trade-off between the population diversity and accuracy is mainly handled by reference-point-based nondominated sorting method. The proposed algorithm is tested by a number of benchmark problems. Experimental results show that it is competitive to state-of-the-art algorithms referred in this paper. The future work should be: (1) Retaining more feasible solutions to improve the performance of the algorithm in each evolutionary generation by adopting other selection strategy; (2) Introducing other better many-objective optimization technique in the algorithm; (3) Using dynamic parameters selection mechanism in DE to speed up the convergence of the algorithm; (4) To explore other candidates of the dynamic environment.

Acknowledgment. This work was supported by the National Natural Science Foundation of China and other foundations(No.s: 61271140, 61203306, 2012001202, 61305086).

References

1. Mezura-Montes, E., Coello, C.A.C.: Constraint handling in nature-inspired numerical optimization: Past, present and future. Swarm Evol. Comput. **1**(4), 173–194 (2011)
2. Kramer, O.: A review of constraint-handling techniques for evolution strategies. In: Applied Computational Intelligence and Soft Computing, vol. 2010 (2010)
3. Sarker, R.A., Elsayed, S.M., Ray, T.: Differential evolution with dynamic parameters selection for optimization problem. IEEE Trans. Evol. Comput. **18**(5), 689–707 (2014)
4. Coello, C.A.C.: Constraint-handling using an evolutionary multi-objective optimization technique. Civil Eng. Environ. Syst. **17**, 319–346 (2000)
5. Coello, C.A.C., Mezura-Montes, E.: Constraint-handling in genetic algorithms through the use of dominance-based tournament election. Adv. Eng. Inform. **16**(3), 193–203 (2002)
6. Venkatraman, S., Yen, G.G.: A generic framework for constrained optimization using genetic algorithms'. IEEE Trans. Evol. Comput. **9**(4), 424–435 (2005)
7. Hsieh, M., Chiang, T., Fu, L.: A hybrid constraint handling mechanism with differential evolution for constrained multiobjective optimization. In: IEEE Congress on Evolutionary Computation, pp. 1785–1792 (2011)
8. Wang, Y., Cai, Z., Guo, G., Zhou, Y.: Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Trans. Syst. Man Cybern. **37**(3), 560–575 (2007)
9. Deb, K., Datta, R.: A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach. In: IEEE Congress on Evolutionary Computation, pp. 165–172 (2010)
10. Zeng, S., Chen, S., Zhao, J., Zhou, A., Li, Z., Jing, H.: Dynamic constrained multiobjective model for solving constrained optimization problem. In: IEEE Congress on Evolutionary Computation, pp. 2041–2046 (2011)
11. Li, X., Zeng, S., Qin, S., Liu, K.: Constrained optimization problem solved by dynamic constrained NSGA-III multiobjective optimizational techniques. In: IEEE Congress on Evolutionary Computation, pp. 2923–2928 (2015)
12. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014)
13. Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.A.C., Deb, K.: Problem definitions and evaluation criteria for the CEC2006 special session on constrained real-parameter optimization (2006). http://www.ntu.edu.sg/home/epnsugan/
14. Elsayed, S.M., Sarker, R.A., Essam, D.L.: Multi-operator based evolutionary algorithms for solving constrained optimization problems. Comput. Oper. Res. **38**(12), 1877–1896 (2011)
15. Mallipeddi, R., Suganthan, P.N.: Ensemble of constraint handling techniques. IEEE Trans. Evol. Comput. **14**(4), 561–579 (2010)