ε Constrained Differential Evolution Algorithm with a Novel Local Search Operator for Constrained Optimization Problems

Wenchao Yi, Xinyu Li*, Liang Gao, and Yinzhi Zhou

State Key Laboratory of Digital Manufacturing Equipment &Technology,
Huazhong University of Science and Technology, Wuhan 430074, PR China
lixinyu@mail.hust.edu.cn

Abstract. Many practical problems can be classified into constrained optimization problems (COPs). ε constrained differential evolution (εDE) algorithm is an effective method in dealing with the COPs. In this paper, ε constrained differential evolution algorithm with a novel local search operator(εDE-LS) is proposed by utilizing the information of the feasible individuals. In this way, we can guide the infeasible individuals to move into the feasible region more effectively. The performance of the proposed εDE-LS is evaluated by the 22 benchmark test functions. The experimental results empirically show that εDE-LS is highly competitive comparing with some other state-of-the-art approaches in constrained optimization problems.

Keywords: Constrained optimization problems, constraint handling technique, ε constrained differential evolution, mutation operator.

1 Introduction

Differential evolution (DE) algorithm is an efficient evolutionary algorithm, which was firstly proposed by Storn and Price [1]. During the past decade, DE algorithm shows competitive performance in solving constrained optimization problems (COPs). In real world applications, a lot of optimization problems are subjected to constraints, which can be categorized into COPs. A general COPs can be stated as follows:

$$\begin{aligned} &\min f(\vec{\mathrm{x}}) \\ &s.t.\ g_j(\vec{\mathrm{x}}) \le 0,\ j = 1,...,q \\ &\quad h_j(\vec{\mathrm{x}}) = 0,\ j = q+1,...,m \end{aligned} \tag{1}$$

Where $\vec{x} = (\mathrm{x}_1,...,\mathrm{x}_n)$ is generated within the range $L_i < x_i < U_i$. L_i and U_i denote the lower and upper bound in each dimension. $g_j(\vec{\mathrm{x}})$ denotes the *j*th inequality constraint and $h_j(\vec{\mathrm{x}})$ denotes the (*j*-*q*)th equality constraint.

* Corresponding author.

H. Handa et al. (eds.), *Proc. of the 18th Asia Pacific Symp. on Intell. & Evol. Systems – Vol. 1*,
Proceedings in Adaptation, Learning and Optimization 1, DOI: 10.1007/978-3-319-13359-1_38

The research on utilizing DE algorithm to solve COPs has attracted promising attention in recent years. A variety of constraint handling techniques have emerged to deal with the COPs. The most common used one is the penalty function method. Huang et al. [2]proposed a co-evolutionary DE algorithm, in which a special adaptive penalty function was proposed to deal with the constraints. Although the penalty function method is simple and efficient, it is still a difficult task to set a proper penalty parameter for the method.

The multiobjective technique is an efficient method in solving COPs. Wang et al. [3] introduced multiobjective technique based DE algorithm to solve COPs. An infeasible solution replacement mechanism based multiobjective approach is proposed, which mainly focus on guiding the population moving towards the promising and feasible region more efficiently. Gong et. al [4]proposed a multiobjective tenique based DE for COPs, in which multiobejctive technique based constraint handling technique was proposed. However, using the multiobjective technique to tackle the COPs is still difficult to design an effective framework in solving COPs.

The methods by adding extra rules or operator in handling COPs also attract considerable interest from the researchers. Storn [5] proposed a constraint adaptive method, which firstly make all the individuals as feasible ones by relaxing the constraints then decresing the relaxation till reach the original constraints. Lampinen et. al [6] presented three effective rules to handle the constraints, which include the feasible ones always better than the infeasible ones, the one with better fitness function value wins among the feasible ones and the one with less constraint violation wins among the infeasible ones. Among these researches, εDE method is a very effective one. εDE was proposed by Takahama et. al [7] in 2006, in which using ε constraint handling technique to deal with the constraints. The experimental results showed the εDE not only can find the feasible ones rapidly, but can achieve excellent success performance and rate as well. In Takahama et. al [7], a local search based on the information of first-order derivative was proposed. However, it is usually difficult to calculate the first-order derivative. Also, the calculation of first-order derivative is time-cost. In this paper, a local search operator is proposed. It can avoid calculating first-order derivative of constraint functions, but is more effective.

The proposed mutation operator mainly utilizing the information of the feasible and the infeasible individuals, which focus on guiding the infeasible individual move along the direction of the feasible individuals. In order to evaluate the performance of the proposed algorithm, twenty-two benchmark test function collected from the special session on the constrained real-parameter optimization of the 2006 IEEE congress on evolutionary computation are adopted in this paper.

This paper is organized as follows. We firstly will give a general introduction of DE algorithm and εDE as a foundation in Section 2. In Section 3, we will give a detailed introduction of the proposed εDE-LS algorithm, in which the framework is included. The experimental results and the comparison will be presented in Section 4. Finally, conclusions will be given in Section 5.

2 DE and εDE Algorithm

2.1 DE Algorithm

DE algorithm is an efficient but simple EAs, which can be divided into four phase, that is, initialization, mutation, crossover and selection.

During the initialization phase, the *NP* *n*-dimensional individuals $x_i^g = \left(x_{i,1}^g, ..., x_{i,n}^g\right), i = 1, ..., NP$ are generated. *g* denotes the generation number.

Then the mutation phase will be adopted to generate the mutation vectors. Several mutation operators have been proposed. The DE/rand/1/*exp* proposed by will be utilized in this paper, where *exp* denotes the exponent crossover operator. The mutation vector can be calculated as follows:

$$v_i^g = x_{r1}^g + F * (x_{r2}^g - x_{r3}^g) \tag{2}$$

Where *F* denotes the predefined scale parameter, r_1, r_2, and r_3 are three mutually different generated indexes which should be different from index *i* within the range [1,*NP*]. Then a check will be made to make sure the generated v_i^g are within the boundaries, which can be described as follows:

$$v_{i,j}^g = \begin{cases} \min\{\mathrm{U}_j, 2 * \mathrm{L}_j - v_{i,j}^g\}, \text{if } v_{i,j}^g < L_j \\ \max\{\mathrm{L}_j, 2 * \mathrm{U}_j - v_{i,j}^g\}, \text{if } v_{i,j}^g > U_j \end{cases} \tag{3}$$

Where L_j and U_j denote the lower and upper bound in *j*-th dimension.

The exponent crossover operator makes the trail vector contains a consecutive sequence of the component taken from the mutation vector. The exponent crossover operator can be given as follows:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, \textit{if } j \in \{\mathrm{k}, \langle k+1 \rangle_n, ..., \langle k+L-1 \rangle_n\} \\ x_{i,\mathrm{j}}^g, \textit{otherwise} \end{cases}, j = 1, ..., \mathrm{n} \tag{4}$$

Where $L, k \in [1, \mathrm{n}]$ are both random indexes. <*j*> is *j* if *j*<*n* and *j*=*j*-*n* if *j*>*n*.

During the selection phase, a better individual between the trail vector u_i^g and target vector x_i^g will be chosen according to their fitness function value:

$$x_i^{g+1} = \begin{cases} u_i^g, \textit{if } f(u_i^g) < f(x_i^g) \\ x_i^g, \textit{else} \end{cases} \tag{5}$$

2.2 εDE Algorithm

In the εDE algorithm, the constraint violation $\Phi(\mathrm{x}_i^g)$ is defined as the sum of all constraints:

$$\Phi(\mathrm{x}_i^g) = \sum_{j=1}^{q} \max\{0, \mathrm{g}_j(\mathrm{x}_i^g)\} + \sum_{j=q+1}^{m} h_j(\mathrm{x}_i^g) \tag{6}$$

After generating the new target vector through the DE algorithm. The ε level comparison is used in εDE algorithm to help decide which individual is better. The comparison can be given as follows:

$$\left(f_1, \Phi_1\right) <_\varepsilon \left(f_2, \Phi_2\right) \Leftrightarrow \begin{cases} f_1 < f_2, if\ \Phi_1, \Phi_2 \le \varepsilon \\ f_1 < f_2, if\ \Phi_1 = \Phi_2 \\ \Phi_1 < \Phi_2, otherwise \end{cases} \tag{7}$$

Where f is the objective fitness function value, and the ε level is set as formula given below:

$$\varepsilon(\mathrm{g}) = \begin{cases} \Phi\left(x_\theta\right), g = 0 \\ \varepsilon(0) * (1 - g / T_c)^{cp}, 0 < g < T_c \\ 0, g \ge T_c \end{cases} \tag{8}$$

Where x_θ is the top θ-th individual and we set θ=0.2*N in the paper. T_c is a predefined generation number. cp is the control parameter in ε level comparison and we set cp as 5 in the paper.

3 The Proposed εDE-LS Algorithm

Usually, in the COPs, the feasible region is continuous. The surrounding region of feasible individuals has more possibility to be feasible. So the feasible individuals can guide the infeasible ones move towards to the feasible region. Motivated by the interaction between the feasible and infeasible individuals, we design a novel local search operator "DE/current-to- feasible/2" to improve the performance of εDE algorithm. The "DE/current-to- feasible/2" is a transformation version of "DE/current-to-best/2" mutation strategy. It can be presented as follows:

$$v_i^g = x_i^g + a * \left(x_{feas\_r_1}^g - x_i^g\right) + b * \left(x_{feas\_r_2}^g - x_i^g\right) \tag{9}$$

Where $feas\_r_1$ and $feas\_r_2$ are two random indexes chosen from the feasible individual set Q. So the number of feasible individuals must be more than 2. a and b are two random generated numbers within the range [0,1]. If any dimension in v_i^g exceeds the boundary, then randomly chosen a feasible individual and makes the specific dimension in v_i^g equal to the related dimension in chosen feasible individual.

So in εDE-LS, DE algorithm is used to generate offspring, and ε constrained algorithm is used to choose better individual survive into next generation. For those infeasible solutions, if the number of feasible individuals is less than 2, the local search phase is skipped. If the number of feasible individuals is more than 2, then for each infeasible individual, local search operator is used. Then the ε level comparison is adopted to choose a better one between the individual and offspring generated by local search operator.. The framework of the proposed εDE-LS algorithm can be given as follows:

εDE-LS algorithm
Initialize the individuals $\{x_1^0,...,x_{NP}^0\}$
Initialize the ε level value, ε=ε(0)
While *Fes*<*maxFES*
{
    Mutation phase
    Crossover phase
    Using ε level comparison in selection phase
    Fes=*Fes*+*NP*;
    Store the ε-feasible ones in *Q;* Store the other individuals as infeasible ones in *W*
    If the number of feasible individuals in *Q* is bigger than 2
        For *i*=1: *size (W)*
            using DE/current-to-feasible/2 to generate $V_i'^g$
            For *j*=1:*n*
                If $V_{i,j}'^g$ exceeds the boundary
                $V_{i,j}'^g = x_{feas\_r_3,j}^{g+1}$
                End If
            End For
            choose better one between $V_i'^g$ and x_i^{g+1} to survive into next generation
            Fes=*Fes*+1;
        End for
    End If
    Update ε level value;
}

Fig. 1. The pesudocode of εDE-LS algorithm

4 Experimental Results

4.1 Parameter Settings

The twenty-two benchmark test functions are collected from Liang et. al [8] are adopted in evaluating the performance of the proposed algorithm. The detailed information about the benchmark can be referred to Liang et. al [8]. The parameter settings of the proposed algorithm are as follows:

Table 1. Parameter setings of εDE-LS algorithm

popsize	40,40,100*
MaxFES	$5\times10^3, 5\times10^4, 5\times10^5$
θ	0.2
T_c	0.2* *MaxFES* /*popsize*
cp	5
F	[0.5,1.0]
CR	[0.9,1.0]

*The popsize is 40, 40 and 100 with 5×10^3, 5×10^4, 5×10^5 finess evaluations (FES),respectively.

4.2 Performance of εDE-LS Algorithm

25 independent runs are conducted for the test benchmark functions with 5×10^3, 5×10^4, 5×10^5 FES, respectively.The torlerance value δ for the equality constraints is set as 0.0001. The best, median, worst, mean and standard deviation of the error value ($f(\vec{x})-f(\vec{x}^*)$), where $f(\vec{x}^*)$ is the best objective fitness function value for each benchmark test function that ever known. *c* is the number of the violated constraints at the median solution : the three numbers refers to the constraints bigger than 1, between 0.01 and 1.0 and between 0.0001and 0.01, respectively. *v* is mean value of the violations of all the constraints at the median solution. The number in parentheses after best, median and worst solutions is the number of violated constraints.

Table 2. Function error values achieved when FES=5×10^3, FES=5×10^4 and FES=5×10^5 for function G01-G05

		G01	G02	G03	G04	G05
5×10^3	Best	1.2042E+00	3.5286E-01	-2.4983E-01	8.2965E-01	5.1278E+03
	Median	2.6309E+00	3.9958E-01	9.6055E-01	3.3045E+00	5.2755E+03
	Worst	5.5134E+00	4.7644E-01	1.0005E+00	9.5916E+00	5.7907E+03
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,3,0
	v	0	0	0	0	2.1310E-02
	Mean	2.7307E+00	4.0586E-01	6.4434E-01	4.0203E+00	5.3362E+03
	Std	7.9984E-01	2.9802E-02	4.9011E-01	2.0836E+00	1.8425E+02

Table 2. (*Continued*)

5×10^4	Best	1.4440E-07	4.1605E-04	2.1900E-06	-3.6380E-12	-1.8190E-12
	Median	6.8327E-07	1.1172E-02	8.9556E-02	-3.6380E-12	-9.0949E-13
	Worst	3.1932E-06	6.2807E-02	2.7449E-01	-3.6380E-12	4.5045E-06
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	9.9408E-07	1.8072E-02	1.0027E-01	-3.6380E-12	2.0085E-07
	Std	8.2529E-07	1.7363E-02	8.1116E-02	0.0000E+00	9.0155E-07
5×10^5	Best	0.0000E+00	8.6703E-10	-2.8866E-15	-3.6380E-12	-1.8190E-12
	Median	0.0000E+00	9.3817E-09	-2.6645E-15	-3.6380E-12	-18190E-12
	Worst	0.0000E+00	3.7095E-08	-2.4425E-15	-3.6380E-12	-1.8190E-12
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	0.0000E+00	1.1279E-08	-2.6024E-15	-3.6380E-12	-1.8190E-12
	Std	0.0000E+00	9.4316E-09	1.6367E-16	0.0000E+00	0.0000E+00

Table 3. Function error values achieved when FES= 5×10^3, FES= 5×10^4 and FES= 5×10^5 for function G06-G10

		G06	G07	G08	G09	G10
5×10^3	Best	6.9833E-04	1.5394E+01	4.1633E-17	3.1226E+00	1.2622E+03
	Median	4.1042E-03	2.1236E+01	5.5511E-17	5.5926E+00	2.1678E+03
	Worst	4.6504E-02	3.6400E+01	6.9389E-17	1.1330E+01	4.8200E+03
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	8.5336E-03	2.2857E+01	5.4401E-17	5.8105E+00	2.2786E+03
	Std	1.0454E-02	6.3786E+00	5.5511E-18	1.8589E+00	7.5188E+02
5×10^4	Best	-1.6371E-11	1.1864E-04	2.7756E-17	-1.1369E-13	1.1531E-02
	Median	-1.6371E-11	2.8898E-04	4.1633E-17	1.1369E-13	5.5891E-02
	Worst	-1.6371E-11	1.2395E-03	4.1633E-17	2.2737E-13	7.8631E+01
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	-1.6371E-11	3.7846E-04	3.7192E-17	6.8212E-14	8.1563E+00
	Std	0.0000E+00	2.7710E-04	6.6071E-18	8.0389E-14	1.7592E+01
5×10^5	Best	-1.6371E-11	-1.4566E-13	2.7756E-17	-2.2737E-13	-7.2760E-12
	Median	-1.6371E-11	2.8422E-14	2.7756E-17	-1.1369E-13	-4.5475E-12
	Worst	-1.6371E-11	2.9488E-13	2.7756E-17	-1.1369E-13	1.4508E-08
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	-1.6371E-11	-1.5632E-14	2.7756E-17	-1.5916E-13	2.0654E-09
	Std	0.0000E+00	1.0766E-13	0.0000E+00	5.6843E-14	4.3554E-09

Table 4. Function error values achieved when FES= 5×10^3, FES= 5×10^4 and FES= 5×10^5 for function G11-G15

		G11	G12	G13	G14	G15
5×10^3	Best	1.6943E-10	8.6228E-05	5.3633E-01	-4.7264E+01	2.2286E-03
	Median	8.1219E-05	7.5430E-03	9.7924E-01	-4.2850E+01	9.6215E+02
	Worst	6.9003E-02	4.2052E-01	1.9416E+00	-3.9904E+01	9.6535E+02
	c	0,0,0	0,0,0	0,3,0	0,3,0	0,0,1
	v	0	0	2.0749E-01	1.1952E-01	3.6741E-04
	Mean	7.7345E-03	5.0767E-06	9.9447E-01	-4.3113E+01	7.7047E+02
	Std	1.5332E-02	1.0478E-05	2.5999E-01	2.0303E+00	3.9265E+02
5×10^4	Best	0.0000E+00	0.0000E+00	1.6771E-02	5.6943E-07	-1.1369E-13
	Median	0.0000E+00	0.0000E+00	5.2714E-01	4.1865E-06	-1.1369E-13
	Worst	0.0000E+00	0.0000E+00	9.2521E-01	2.1939E-04	-1.1369E-13
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	0.0000E+00	0.0000E+00	5.3646E-01	2.2823E-05	-1.1369E-13
	Std	0.0000E+00	0.0000E+00	3.0524E-01	5.2442E-05	0.0000E+00
5×10^5	Best	0.0000E+00	0.0000E+00	-2.2204E-16	1.4211E-14	-1.1369E-13
	Median	0.0000E+00	0.0000E+00	-2.2204E-16	1.4211E-14	-1.1369E-13
	Worst	0.0000E+00	0.0000E+00	-1.9429E-16	2.1316E-14	-1.1369E-13
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	0.0000E+00	0.0000E+00	-2.1649E-16	1.7053E-14	-1.1369E-13
	Std	0.0000E+00	0.0000E+00	1.1331E-17	3.5527E-15	0.0000E+00

Table 5. Function error values achieved when FES= 5×10^3, FES= 5×10^4 and FES= 5×10^5 for function G16-G19 and G21

		G16	G17	G18	G19	G21
5×10^3	Best	3.9394E-03	8.8160E+03	-2.6751E-01	5.3038E+01	4.0133E+02
	Median	9.6029E-03	8.9614E+03	5.1285E-01	1.0054E+02	6.7771E+02
	Worst	2.4616E-02	9.2010E+03	8.0098E-01	1.3151E+02	9.6477E+02
	c	0,0,0	0,4,0	0,0,0	0,0,0	0,3,1
	v	0	9.5126E-02	0	0	1.8268E-03
	Mean	1.1051E-02	8.9672E+03	4.9005E-01	9.6983E+01	6.7307E+02
	Std	5.3806E-03	1.0383E+02	1.8203E-01	2.4519E+01	1.5715E+02
5×10^4	Best	3.7748E-15	7.7749E+00	3.3295E-06	1.2885E-02	2.7546E-05
	Median	3.2196E-14	8.3318E+01	1.8149E-05	3.2261E-02	4.5160E+00
	Worst	6.3349E-13	3.4429E+02	1.2721E-04	1.151E-01	1.3099E+02
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
	v	0	0	0	0	0
	Mean	1.0176E-13	7.5867E+01	2.8616E-05	3.6846E-02	5.0162E+01
	Std	1.5267E-13	6.6433E+01	2.9068E-05	2.1706E-02	5.7877E+01

Table 5. (*Continued*)						
5×10^5	Best	3.7748E-15	-5.8000E-03	1.7130E-11	1.5960E-08	0.0000E+00
	Median	3.7748E-15	-5.8000E-03	1.8764E-10	5.4233E-08	2.2989E-299(6)
	Worst	3.7748E-15	2.9749E+02	7.7298E-10	1.6040E-07	2.7393E-259(6)
	c	0,0,0	0,0,0	0,0,0	0,0,0	0,6,0
	v	0	0	0	0	2.4686E-01
	Mean	3.7748E-15	3.9079E+01	1.9762E-10	6.4420E-08	1.1283E-260
	Std	0.0000E+00	6.5092E+01	1.8283E-10	3.2980E-08	0.0000E-00

Table 6. Function error values achieved when FES= 5×10^3, FES= 5×10^4 and FES= 5×10^5 for function G23-G24

		G23	G24
5×10^3	Best	-4.9639E+02	3.5212E-09
	Median	-1.3372E+02	6.5883E-08
	Worst	2.3647E+02	4.0692E-07
	c	0,4,0	0,0,0
	v	1.1303E-01	0
	Mean	-9.5647E+01	1.1387E-07
	Std	1.6961E+02	1.1752E-07
5×10^4	Best	2.3201E-01	3.2863E-14
	Median	3.7585E+01	3.2863E-14
	Worst	3.1068E+02	3.2863E-14
	c	0,0,0	0,0,0
	v	0	0
	Mean	4.9361E+01	3.2863E-14
	Std	6.1611E+01	0.0000E+00
5×10^5	Best	7.9471E-08	3.2863E-14
	Median	7.9992E-06	3.2863E-14
	Worst	4.6202E-02	3.2863E-14
	c	0,0,0	0,0,0
	v	0	0
	Mean	1.8593E-03	3.2863E-14
	Std	9.2381E-03	0.000E+00

As shown in Table 2-6, in spite of the test functions G05, G13, G14, G15, G17, G21, G23, for other 15 test benchmark functions, the proposed algorithm can obtain feasible solutions within 5×10^3 FES. All the test benchmark functions can obtain feasible solutions within 5×10^4 FES. Especially, to function G11 and G12, the best known solutions are obtained within 5×10^4 FES. In 5×10^5 FES, 9 out of 22 test benchmark functions (i.e. G03, G04, G05, G06, G07, G09, G10, G13, G15) can obtain a more precisely solutions than the best known solutions. In conclusion, the

solutions obtained by the proposed εDE-LS algorithm (except G21) are close to the best known solutions within 5×10^5 FES.

In Table 7, we present the number of FES needed in each run for each test benchmark function when satisfying the success condition: $f(\vec{x})-f(\vec{x}^*)\le 1.0E-04$ and $\vec{x}$ is feasible solution. The best, median, worst, mean and std denote the least, median, most, mean and standard deviation FES when meets the success condition during the 25 independent runs. The feasible rate is the ratio between the feasible solutions and 25 achieved solutions within 5×10^5 FES. The success rate is the ratio between the number of success runs and 25 runs within 5×10^5 FES. The success performance is the mean number of FES for successful runs multiplied by the total runs and divided by the number of successful runs.

In Table 8, a comparison with respect to other state-of-the-art algorithms in terms of the success performance. The related success performance of other state-of-the-art algorithms is can be referred to Wang et al.[3].

Table 7. The success performance, feasible rate and success rate of the εDE-LS algorithm

Pro.b	Best	Median	Worst	Mean	Std	Feasible Rate	Success Rate	Success performance
G01	33360	37480	40680	37290	2012.9	100%	100%	37290
G02	204100	273100	294000	256662	54591.1	100%	100%	256662
G03	97800	102200	106300	101960	2288.4	100%	100%	101960
G04	12440	14600	16040	14565	977.2	100%	100%	14565
G05	16240	29360	43960	28365	8458.6	100%	100%	28365
G06	23160	25200	27800	25426	1077.2	100%	100%	25426
G07	202900	212700	222900	212300	5168.8	100%	100%	212300
G08	320	7240	10080	7162	1745.0	100%	100%	7162
G09	18840	19920	21520	20061	777.61	100%	100%	20061
G10	243600	286900	384300	306412	53413.7	100%	100%	306412
G11	5600	7200	7800	7102	504.4	100%	100%	7102
G12	1120	2960	5040	3038	957.4	100%	100%	3038
G13	82100	85400	87000	84812	1419.3	100%	100%	84812
G14	32360	39720	47720	39766	4893.0	100%	100%	39766
G15	9480	10400	11840	10435	653.7	100%	100%	10435
G16	13920	20800	25080	19832	3037.2	100%	100%	19832
G17	173800	326800	366500	257400	56434	100%	56%	459642
G18	35520	44560	49680	43797	3947.7	100%	100%	43797
G19	287900	314700	335300	314744	10950.7	100%	100%	314744
G21	NA	NA	NA	NA	NA	NA	NA	NA
G23	384200	431300	499800	433073	33406.8	100%	92%	451117
G24	2440	2960	3360	2963	221.5	100%	100%	2963

Table 8. εDE-LS with respect to MDE[9], MPDE[10], GDE[11], jDE-2[12], CMODE[3] in terms of success performance

Prob.	Success performance						
	εDE	MDE	MPDE	GDE	jDE-2	CMODE	εDE-LS
G01	5.9E+04	7.5E+04	4.3E+04	4.1E+04	5.0E+04	1.2E+05	**3.7E+04**
G02	1.5E+05	**6.0E+04**	3.0E+05	1.5E+05	1.5E+05	1.9E+05	2.6E+05
G03	8.9E+04	4.5E+04	**2.5E+04**	3.5E+06	NA	7.5E+04	1.0E+05
G04	2.6E+04	4.2E+04	2.1E+04	1.5E+04	4.1E+04	7.3E+04	**1.5E+04**
G05	9.7E+04	**2.1E+04**	2.2E+05	1.9E+05	4.5E+05	2.9E+04	2.8E+04
G06	7.4E+03	**5.2E+03**	1.1E+04	6.5E+03	2.9E+04	3.5E+04	2.5E+04
G07	7.4E+04	1.9E+05	5.7E+04	1.2E+05	1.3E+05	1.6E+05	**2.1E+04**
G08	1.1E+03	**9.2E+02**	1.5E+03	1.5E+03	3.2E+03	5.9E+03	7.2E+03
G09	2.3E+04	**1.6E+04**	2.1E+04	3.0E+04	5.5E+04	7.1E+04	2.0E+04
G10	1.1E+05	1.6E+05	**4.8E+04**	8.3E+04	1.5E+05	1.8E+05	3.1E+05
G11	1.6E+04	**3.0E+03**	2.3E+04	8.5E+03	5.4E+04	6.0E+03	7.1E+03
G12	4.1E+03	**1.3E+03**	4.2E+03	3.1E+03	6.4E+03	5.0E+03	3.0E+03
G13	3.5E+04	**2.2E+04**	7.4E+05	8.7E+05	NA	3.1E+04	8.5E+04
G14	1.1E+05	2.9E+05	4.3E+04	2.3E+05	9.8E+04	1.1E+05	**4.0E+04**
G15	8.4E+04	1.0E+04	2.0E+05	7.5E+04	2.4E+05	1.3E+04	**1.0E+04**
G16	1.3E+04	**8.7E+03**	1.3E+04	1.3E+04	3.2E+04	2.9E+04	2.0E+04
G17	9.9E+04	**2.6E+04**	7.3E+05	2.1E+06	1.1E+07	1.4E+05	4.6E+05
G18	5.9E+04	1.0E+05	4.4E+04	4.8E+05	1.0E+05	1.1E+05	**4.4E+04**
G19	3.5E+04	NA	**1.2E+05**	2.0E+05	2.0E+05	2.5E+05	3.2E+05
G21	1.4E+05	1.1E+05	2.1E+05	5.8E+05	1.3E+05	1.3E+05	NA
G23	**2.0E+05**	3.6E+05	2.1E+05	1.1E+06	3.6E+05	2.4E+05	4.5E+05
G24	3.0E+03	1.8E+03	4.3E+03	3.1E+03	1.0E+04	2.2E+04	**3.0E+03**

From Table 7-8, we can conclude that the performance of εDE-LS algorithm is highly competitive. 19 out of 22 test benchmark functions can achieve 100% success rate within 5×10^5 FES. εDE-LS algorithm achieves 100% feasible In terms of success performance, εDE-LS algorithm obtained the least FES in test benchmark function G01, G04, G07, G14, G15, G18, and G24 comparing with other six state-of-the-art algorithms. As success performance indicate that the proposed εDE-LS requires less than 1×10^4 FES for 4 test benchmark functions, less than 5×10^4 FES for 14 test benchmark functions, less than 5.0×10^5 FES for 21 test benchmark functions to obtain the require accuracy.

5 Conclusion

This paper proposed the εDE-LS algorithm, in which a novel local search operator designed for COPs are introduced. The feasible and infeasible individuals can interact

with each other by applying the proposed mutation operator. By utilizing the novel mutation operator as the local search engine, we can guide the population moving towards the feasible region more effective. The effectiveness of the proposed εDE-LS algorithm is demonstrated by 22 test benchmark functions collected from IEEE CEC2006 special session on constrained real parameter optimization. The experimental results suggest that εDE-LS algorithm is highly competitive in terms of accuracy and convergent speed. εDE-LS algorithm can successfully solve 21 test benchmark functions and can achieve 21 feasible optimal solutions consistently. The success performance of εDE-LS algorithm is highly competitive when compares with other state-of-the-art algorithms. As the effectiveness and efficiency of the proposed algorithm demonstrated above, we can conclude that the εDE-LS is highly competitive one in dealing with COPs and should gain attention from researchers in the future. Besides, the performance of the εDE-LS can be further studied through using other indicators. In the future, more real world applications can be tested by the proposed εDE-LS algorithm. Moreover, as a part of the future direction, the performance of εDE-LS may be further improved by discovering a more efficient mutation operator.

Acknowledgement. This research work is supported by the National Basic Research Program of China (973 Program) under Grant no. 2011CB706804, and the Natural Science Foundation of China (NSFC) under Grant no. 51005088 and 51121002.

References

1. Storn, R., Price, K.: Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
2. Huang, F.Z., Wang, L., He, Q.: An Effective Co-evolutionary Differential Evolution for Constrained Optimization. Applied Mathematics and Computation 286, 340–356 (2007)
3. Wang, Y., Cai, Z.X.: Combining Multiobjective Optimization with Differential Evolution to Solve Constrained Optimization Problems. IEEE Transactions on Evolutionary Computation 16, 117–134 (2012)
4. Gong, W., Cai, Z.: A Multiobjective Differential Evolution Algorithm for Constrained Optimization. In: 2008 Congress on Evolutionary Computation (CEC 2008), pp. 181–188 (2008)
5. Storn, R.: System Design by Constraint Adaptation and Differential Evolution. IEEE Transactions on Evolutionary Computation, 22–34 (1999)
6. Lampinen, J.: A Constraint Handling Approach for Differential Evolution Algorithm. In: Proceedings of the Congress on Evolutionary Computation (CEC 2002), pp. 1468–1473 (2002)
7. Takahama, T., Sakai, S.: Constrained Optimization by the ε-Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites. In: 2006 IEEE congress on Evolutionary Computation (CEC 2006), pp. 308–315 (2006)
8. Liang, J.J., Runarsson, T.P., Mezura-Montes, E., et al.: Problems Definitions and Evaluation Criteria for the CEC' 2006 Special Session on Constrained Real-parameter Optimization (2006), `http://www.ntu.edu.sg/home/EPNSugan/cec2006/technicalreport.pdf`

9. Mezura-Montes, E., Velázquez-Reyes, J., CoelloCoello, C.A.: Modified Differential Evolution for Constrained Optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC 2006), pp. 332–339 (2006)
10. Tasgetiren, M.F., Suganthan, P.N.: A Multi-populated DifferentialEvolution Algorithm for Solving Constrained Optimization Problem. In: Proceedings of the Congress on Evolutionary Computation (CEC 2006), pp. 33–40 (2006)
11. Kukkonen, S., Lampinen, J.: Constrained Real-parameter Optimizationwith Generalized Differential Evolution. In: Proceedings of the Congress on Evolutionary Computation (CEC 2006), pp. 207–214 (2006)
12. Brest, J., Zumer, V., Maucec, M.S.: Self-adaptive DifferentialEvolution Algorithm in Constrained Real-parameter Optimization. In: Proceedings of the Congress onEvolutionary Computation (CEC 2006), pp. 215–222 (2006)