LSHADE44 with an Improved ϵ Constraint-handling Method for Solving Constrained Single-objective Optimization Problems

Zhun Fan
Department of Electronic Engineering
Shantou University
Guangdong, Shantou 515063

Yi Fang
Department of Electronic Engineering
Shantou University
Guangdong, Shantou 515063

Wenji Li
Department of Electronic Engineering
Shantou University
Guangdong, Shantou 515063

Yutong Yuan
Department of Electronic Engineering
Shantou University
Guangdong, Shantou 515063

Zhaojun Wang
Department of Electronic Engineering
Shantou University
Guangdong, Shantou 515063

Xinchao Bian
Department of Electronic Engineering
Shantou University
Guangdong, Shantou 515063

***Abstract*—This paper proposes an improved ϵ constrained handling method (IEpsilon) for solving constrained single-objective optimization problems (CSOPs). The IEpsilon method adaptively adjusts the value of ϵ according to the proportion of feasible solutions in the current population, which has an ability to balance the search between feasible regions and infeasible regions during the evolutionary process. The proposed constrained handling method is embedded to the differential evolutionary algorithm LSHADE44 to solve CSOPs. Furthermore, a new mutation operator DE/randr1*/1 is proposed in the LSHADE44-IEpsilon. In this paper, twenty-eight CSOPs given by "Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization" are tested by the LSHADE44-IEpsilon and four other differential evolution algorithms CAL-SHADE, LSHADE44+IDE, LSHADE44 and UDE. The experimental results show that the LSHADE44-IEpsilon outperforms these compared algorithms, which indicates that the IEpsilon is an effective constraint-handling method to solve the CEC2017 benchmarks.**

***Index Terms*—Constraint-handling Technique, Constrained Single-objective Optimization, Differential Evolution**

I. Introduction

Many real-parameter optimization problems in the real world are subject to constraints [1]. Without loss of generality, a CSOP can be defined as follows:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}),\ \mathbf{x} = (x_1, ..., x_D) \in S \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, \ldots, q \\ & h_j(x) = 0, j = 1, \ldots, p \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is the objective function. $\mathbf{x}$ is the decision vector and x_i is the i-th component of $\mathbf{x}$. $S = \prod_{i=1}^{D} [L_i,\ U_i]$ is the decision space, where D is the dimension of $\mathbf{x}$. L_i and U_i are the lower bound and the upper bound of x_i, respectively. $g_i(\mathbf{x}) \leq 0$ is the i-th inequality constraint, and $h_j(\mathbf{x}) = 0$ is the j-th equality constraint. In order to evaluate the violation of the constraint functions, the overall constraint violation method is adopted. It summaries all constraints into a scalar value. The overall constraint violation of the solution $\mathbf{x}$ can be defined as follows:

$$\phi(\mathbf{x}) = \sum_{i=1}^{q} \max(g_i(\mathbf{x}), 0) + \sum_{j=1}^{p} \max(|h_j(\mathbf{x})| - \sigma, 0) \tag{2}$$

In this paper, σ is set as 0.0001 according to [2]. If $\phi(\mathbf{x})$ is equal to 0, the solution $\mathbf{x}$ is feasible, otherwise it is infeasible.

To tackle complex numerical optimization problems, differential evolution (DE) is proposed [3], which has been proved to be an efficient algorithm in the past two decades [4]. In a general framework of DE, the scaling factor F, the crossover rate CR and the population size N are three important factors, which need to be preset. A series of variants of DE are proposed to adjust these factors.

For example, FADE [5] regards the information of the population in two generations as the input of the fuzzy logic control and generate the values of F and CR. jDE [6] updates F and CR with different probabilities in each generation. JADE [7] generates a pair of F and CR for each individual, according to Cauchy and normal distribution with an adaptive mean μ_F and μ_{CR} respectively. μ_F and μ_{CR} are updated according to the pairs of CR and F, which have generated successful trial vectors in the previous generation. CoDE [8] applies several groups of F and CR in DE. SHADE [9] proposes a new factor adaption setting technique according to a memory of successful CR and F. LSHADE [10] is a variant of SHADE, which reduces the population size N linearly with the increasing of objective function evaluations (FEs).

In DE, a suitable evolutionary strategy (such as the mutation operators) should be selected. The current-to-$p$best/1 [7] is a kind of greedy strategy used in LSHADE [10]. To enhance

the robust of LSHADE for different types of CSOPs, an improved version of LSHADE (named LSHADE44) using 4 kinds of strategies adaptively during the evolutionary process, is proposed in [11].

To solve CSOPs, an efficient constraint-handling method is also an important component for heuristic algorithms. According to [12], constraint-handling methods can be generally classified into (1) Penalty approaches; (2) Repair approaches; (3) Separatist approaches and (4) Hybrid approaches.

As a representative separatist approach, constrained dominance principle (CDP) is widely used [13]. It is defined as follows (for minimization CSOPs):

(1) When two feasible solutions are compared, the one with the smaller objective value is better than the other.
(2) A feasible solution is always better than an infeasible solution.
(3) When two infeasible solutions are compared, the one with the lower overall constraint violation is better than the other.

As a modified version of CDP, an ϵ constraint-handling method [14] is defined as follow (for minimization CSOPs):

(1) When the overall constraint violations of two solutions are both lower than or equal to ϵ, the one with the smaller objective value is better than the other.
(2) When the overall constraint violations of two solutions are equal, the one with the smaller objective value is better than the other.
(3) When at least one of the constraint violations of two solutions are larger than ϵ, the one with the lower constraint violation is better than the other.

The ϵ constraint-handling method can help the working population fully explore the infeasible regions, which prevents the population trapping into local optimum. When $\epsilon = 0$, the ϵ constraint-handling method is the same as CDP.

In the original framework of the ϵ constraint-handling method [14], the value of ϵ is decreasing with the increasing of FEs. When the proportion of feasible solutions in the current generation (PFS) is small, the ϵ should be set small in order to achieve more feasible solutions. When the PFS is large, the ϵ should be also set large in order to get some infeasible solutions, which can help the working population get across infeasible regions. In this paper, we propose an improved version of ϵ constrained method (IEpsilon) to adaptively adjust the value of ϵ according to the PFS.

The proposed IEpsilon is applied to the framework of LSHADE44, and a new strategy named DE/randr1*/1 is proposed to the original framework. DE/randr1*/1 uses the target individual to replace a random parent individual, which can reduce the difference between the generated trial vector and the target individual to enhance the local search for the target individual.

28 problems [2] with 4 kinds of dimensions (D = 10, 30, 50, 100) are tested by the proposed LSHADE44-IEpsilon and other four DE algorithms CAL-SHADE [15], LSHADE44+IDE [16], LSHADE44 [11] and UDE [17] in 25 independent runs.

The rest of this paper is organized as follows. Section II introduces differential evolution (DE). Section III introduces the IEpsilon method and the proposed LSHADE44-IEpsilon. Section IV gives the experimental results of LSHADE44-IEpsilon and other four DE algorithms on the test problems, and Section V concludes the paper.

II. Differential Evolution

The DE initially generates a population P with a size of N in the D-dimension decision space S. In each generation, mutation and crossover operators with parameters F_i and CR_i are applied to each individual $\mathbf{x}_i$ ($i = 1, ...N$). Then a trial vector $\mathbf{y}_i$ is generated. If $f(\mathbf{y}_i)$ is lower than $f(\mathbf{x}_i)$, $\mathbf{x}_i$ will be replaced by $\mathbf{y}_i$. The algorithm continues to evolve the population until the stopping criteria are met, and then it outputs the best solution of population. The pseudo-code of DE is shown in Algorithm 1.

Algorithm 1: Differential Evolution

Input:

1) a SOP and a stopping criterion.
2) N: the population size.

Output: A best solution $\mathbf{x}_{best}$;

Step 1: Initialization: Randomly generate a population $P = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ in S and evaluate each solution in P.

Step 2: Population update

For $i = 1, \ldots, N$, do

a) Perform a mutation operator on $\mathbf{x}_i$ to generate $\mathbf{v}_i$.
b) Perform a crossover operator on $\mathbf{v}_i$ and $\mathbf{x}_i$ to generate a trial vector $\mathbf{y}_i$.
c) **Update of Solutions**: If $f(\mathbf{y}_i) < f(\mathbf{x}_i)$, replace $\mathbf{x}_i$ with $\mathbf{y}_i$.

Step 3: Termination If stopping criteria are satisfied, output the best solution $\mathbf{x}_{best}$ in P. Otherwise, go to **Step 2**.

Some classical mutation operators are introduced as follows:

(1) DE/rand/1:
$\mathbf{v}_{i,G} = \mathbf{x}_{r1,G} + F_i \cdot (\mathbf{x}_{r2,G} - \mathbf{x}_{r3,G})$
(2) DE/randr1/1 [18]:
$\mathbf{v}_{i,G} = \mathbf{x}_{r1^*,G} + F_i \cdot (\mathbf{x}_{r2^*,G} - \mathbf{x}_{r3^*,G})$
(3) DE/current-to-best/1:
$\mathbf{v}_{i,G} = \mathbf{x}_{i,G} + F_i \cdot (\mathbf{x}_{best,G} - \mathbf{x}_{i,G}) + F_i \cdot (\mathbf{x}_{r1,G} - \mathbf{x}_{r2,G})$
(4) DE/current-to-$p$best/1:
$\mathbf{v}_{i,G} = \mathbf{x}_{i,G} + F_i \cdot (\mathbf{x}_{pbest,G} - \mathbf{x}_{i,G}) + F_i \cdot (\mathbf{x}_{r1,G} - \mathbf{x}_{r2,G})$

where $\mathbf{x}_{i,G}$ ($i = 1, 2, ..., N$) is the target solution in the generation G. $\mathbf{v}_{i,G}$ is a vector generated by the mutation operator. $r1$, $r2$ and $r3$ are different integer labels uniformly chosen from the set $\{1, 2, ..., N\} \setminus \{i\}$. $r1^* \in \{r1, r2, r3\}$ is the label which makes $\mathbf{x}_{r1^*,G}$ be the best solution, and then $r2^*$, $r3^*$ are the two rest labels. $\mathbf{x}_{best,G}$ is the best solution in the current population. p is a random value falling in $[1/N, 0.2]$. $\mathbf{x}_{pbest,G}$ is the solution which is randomly chosen from the $100p\%$ best solutions in the current population. There are two versions of DE/current-to-$p$best/1 operators [7], and

the version of the operator without external archive is chosen in this paper.

After performing a mutation operator, a crossover operator will be performed for $\mathbf{x}_{i,G} = (x_{i,1,G}, ..., x_{i,D,G})$ and $\mathbf{v}_{i,G} = (v_{i,1,G}, ..., v_{i,D,G})$. There are mainly two crossover operators in DE, which are listed as follows::

(1) Binomial crossover (Bin):

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if rand}_j(0,1) \leq CR_i \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{otherwise} \end{cases}$$

where j_{rand} is an integer randomly chosen from $[1, D]$ and $\text{rand}_j(0,1)$ is a random number falling in $[0, 1]$.

(2) Exponential crossover (Exp) [19]:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{for } j = \langle l \rangle_D, \langle l+1 \rangle_D, ..., \langle l+L-1 \rangle_D \\ x_{i,j,G} & \text{otherwise} \end{cases}$$

where $\langle\rangle_D$ denotes a modulo function with modulus D, l is the starting integer number randomly chosen from $[0, D-1]$. The integer L is drawn from $[0, D-1]$ with the probability $P_r(L \geq v) = CR_i^{v-1} (v > 0)$.

III. LSHADE44-IEpsilon

A. LSHADE44

1) Successful History based Parameter Settings: As we have discussed, each individual $\mathbf{x}^i$ is associated with its parameter pair $\{F_i, CR_i\}$. In LSHADE [10], F and CR are generated by successful history memories. There are a pair of memories M_F and M_{CR} with H cells, which store the adaptive values m_F and m_{CR}.

k is a parameter, which is initialized as 0. At the beginning of each generation, two sets S_F and S_{CR} are both initialized as $\emptyset$, which are used to store successful parameters. For each individual $\mathbf{x}_i$ $(i = 1, 2, ..., N)$, when its trial vector $\mathbf{y}_i$ is better than $\mathbf{x}_i$, F_i and CR_i will be inserted to the sets S_F and S_{CR}. Then at the end of each generation, if S_F and S_{CR} are not empty, the value of k will be increased by 1. When the value of k is larger than H, the value of k will be reset as 1. Then, the adaptive values m_F and m_{CR} are calculated by Eq (3)-(8), which will be stored into the k-th cell of memories M_F and M_{CR}.

$$m_F = mean_{WL}(S_F) \quad \text{if } S_F \neq \emptyset \tag{3}$$

$$m_{CR} = mean_{WA}(S_{CR}) \quad \text{if } S_{CR} \neq \emptyset \tag{4}$$

$$mean_{WL}(S_F) = \frac{\sum_{t_1=1}^{|S_F|} \omega_{t_1} F_{t_1}^2}{\sum_{t_2=1}^{|S_F|} \omega_{t_2} F_{t_2}} \tag{5}$$

$$mean_{WA}(S_{CR}) = \sum_{t=1}^{|S_{CR}|} \omega_t CR_t \tag{6}$$

$$\omega_t = \frac{\Delta func_t}{\sum_{u=1}^{|S_{CR}|} \Delta func_u} \tag{7}$$

$$\Delta func_t = |func(\mathbf{x}_t) - func(\mathbf{y}_t)| \tag{8}$$

where $func(\cdot)$ is the objective function $f(\cdot)$ when $\phi(\mathbf{x}) = \phi(\mathbf{y})$ but $f(\mathbf{x}) > f(\mathbf{y})$, according to Eq (1), (2). $func(\cdot)$ is the constraint violation function $\phi(\cdot)$ when $\phi(\mathbf{x}) > \phi(\mathbf{y})$.

When the M_F and M_{CR} are both empty, the means μ_F and μ_{CR} are both set as 0.5. Otherwise, a random integer r_i is selected from $[1, H]$, and then the r_i-th pair of values in M_F and M_{CR} will be selected as the value of μ_F and μ_{CR}.

Before generating a trial vector $\mathbf{y}_i$, the two parameters F_i and CR_i are generated by using means μ_F and μ_{CR} as follows:

$$F_i = \text{randc}_i(\mu_F, 0.1) \tag{9}$$

$$CR_i = \text{randn}_i(\mu_{CR}, 0.1) \tag{10}$$

where $\text{randc}_i(\mu, \sigma)$ denotes a value generated by a Cauchy distribution of mean μ and standard deviation σ. $\text{randn}_i(\mu, \sigma)$ denotes a value generated by a normal distribution of mean μ and standard deviation σ. F_i and CR_i should both fall in the interval $[0, 1]$. When their values are beyond $[0, 1]$, Eq (9) or (10) will be executed repeatedly till $F_i, CR_i \in [0, 1]$.

2) Linear Population Size Reduction: To improve the performance of DE, a population size reduction method is adopted in the original LSHADE. The population size N dynamically decreases with the increasing of FEs according to Eq (11).

$$N = round[N_{max} - \frac{FEs}{MaxFEs}(N_{max} - N_{min})] \tag{11}$$

where N_{max} is the initial population size, and N_{min} is the minimal population size. $MaxFEs$ is the allowed number of function evaluations.

3) DE/randr1/1:* Here we propose a modified version of DE/randr1/1 mutation operator, which is defined as follows: DE/randr1*/1

$$\mathbf{v}_{i,G} = \mathbf{x}_{r1^*,G} + F_i \cdot (\mathbf{x}_{r2^*,G} - \mathbf{x}_{r3^*,G}) \tag{12}$$

where $\mathbf{v}_{i,G}$ is the mutation vector of the target individual $\mathbf{x}_{i,G}, (i \in 1, ..., N)$ at the generation G. $r1$ and $r2$ are different integer labels uniformly chosen from the set $\{1, 2, ..., N\} \backslash \{i\}$. $r1^* \in \{r1, r2, i\}$ is the label which makes $\mathbf{x}_{r1^*,G}$ be the best solution, and then $r2^*$, $r3^*$ are the two rest labels $\{r1, r2, i\} \setminus r1^*$.

In the DE/randr1/1, all mutated individuals are not the target individual. It may cause that the mutation vector differs much from the target individual. The DE/randr1*/1 can reduce the difference between the mutation vector and the target individual, which can accelerate the local search for each individual in the population.

4) Strategies Competing Selection: For different kinds of problems, different evolutionary strategies should be adopted. However, in real world, many optimization problems are black-box problems. Without prior knowledge of problems, various strategies applying to the DE simultaneously can enhance the robust of the algorithm for different problems. In this paper, four groups of strategies are adopted in the LSHADE44: (1) DE/current-to-*p*best/1 and Bin; (2) DE/current-to-*p*best/1

and Exp; (3) DE/randr1*/1 and Bin; (4) DE/randr1*/1 and Exp.

A mechanism of strategies competition [11] is applied to the LSHADE44 to adaptively select strategies during the process. The selection probabilities are given to each strategies. At the beginning of the evolution, the same probability ($q_l = 1/K$ for $l = 1, 2, ..., K$) is related to each strategy. When a strategy is successful, the probability of each strategy is updated as follows:

$$q_l = \frac{n_l + n_0}{\sum_{k=1}^{K}(n_k) + n_0} \tag{13}$$

where n_l is the successful count of the l-th strategy, and $n_0 > 0$ is a constant, which weakens the influence of a random success of l-th strategy for q_l. To avoid degeneration of the algorithm, if any probability q_l decreases below a given parameter δ during the evolutionary process, each q_l and n_l are reset as $1/K$ and 0.

5) Bound Constraint Handling: In this paper, when a generated solution is evaluated, a bound constraint handling technique [20] is executed. For $\mathbf{x} = (x_1, ..., x_i..., x_D)$, if $x_i < L_i$, we set $x_i = L_i$, and if $x_i > U_i$, we set $x_i = U_i$.

B. Improved ϵ Level Comparison

The original DE introduced above is suitable for the unconstrained SOPs. To provide comparison rules for the DE on CSOPs during the evolutionary process, the ϵ level comparison $<_\epsilon$ [21] is applied. Given an ϵ, a solution $\mathbf{x}_a$ is considered better than another solution $\mathbf{x}_b$ according to Eq (14)

$$\mathbf{x}_a <_\epsilon \mathbf{x}_b \Leftrightarrow \begin{cases} f(\mathbf{x}_a) < f(\mathbf{x}_b) & \text{if } \phi(\mathbf{x}_a), \phi(\mathbf{x}_b) \leq \epsilon \\ f(\mathbf{x}_a) < f(\mathbf{x}_b) & \text{if } \phi(\mathbf{x}_a) = \phi(\mathbf{x}_b) \\ \phi(\mathbf{x}_a) < \phi(\mathbf{x}_b) & \text{otherwise} \end{cases} \tag{14}$$

where $\phi(\cdot)$ is the overall constraint violation function and $f(\cdot)$ is the objective function.

To balance the evolutionary search of the population between feasible and infeasible regions, an improved ϵ setting approach is suggested as follows:

$$\epsilon(k) = \begin{cases} \phi_\theta & \text{if } k = 0 \\ \epsilon(k-1)(1 - \frac{FEs}{T_c})^{cp} & \text{if } r_k < \alpha \text{ and } FEs < T_c \\ (1+\tau)\phi_{max} & \text{if } r_k \geq \alpha \text{ and } FEs < T_c \\ 0 & \text{if } FEs \geq T_c \end{cases} \tag{15}$$

where $\epsilon(k)$ is the value of ϵ and r_k is the proportion of feasible solutions (PFS) in the generation k. where ϕ_θ is the top θ-th overall constraint violation of all individuals in the initial population. T_c is the termination FEs and when FEs reaches T_c the value of ϵ will be set as 0. cp and τ are defined by users. α is a threshold to control the change of ϵ.

Algorithm 2: LSHADE44-IEpsilon

Input:

1) a SOP and a stopping criterion.
2) N_{max}, N_{min}: the maximal and the minimal population sizes.
3) cp, θ, T_c, α, τ: parameters of IEpsilon.
4) H: the length of the historic memories.
5) n_0, δ: parameters of strategy competition.

Output: A best solution $\mathbf{x}_{best}$.

Step 1: Initialization:

a) Set probabilities $q_l = 1/4$ for $l = 1, 2, 3, 4$.
b) Set counts $n_l = 0$ for $l = 1, 2, 3, 4$.
c) Generate a population $P = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, $N = N_{init}$.
d) Evaluate $f(\mathbf{x}_i)$ and $\phi(\mathbf{x}_i)$, $i = 1, ..., N$. Get the top and the top θ constraint violations ϕ_{max} and ϕ_θ.
e) Set $\epsilon = \phi_\theta$.

Step 2: Population update

For $i = 1, \ldots, N$, do

a) Choose the l-th strategy according to the $q_l, l = 1, 2, 3, 4$.
b) Generate F_i and CR_i according to the memories M_F and M_{CR}.
c) Generate a trial vector $\mathbf{y}_i$, evaluate $f(\mathbf{y}_i)$ and $\phi(\mathbf{y}_i)$, update $\mathbf{x}_{best}$ and ϕ_{max}.
d) **If** $\max(\phi(\mathbf{y}_i) - \epsilon, 0) < \max(\phi(\mathbf{x}_i) - \epsilon, 0)$ **then**
   1) replace $\mathbf{x}_i$ with $\mathbf{y}_i$,
   2) $n_l = n_l + 1$, update $q_s, s = 1, 2, 3, 4$ according to Eq (13),
   3) store $|\phi(\mathbf{x}_i) - \phi(\mathbf{y}_i)|$, store F_i and CR_i.
d) **ElseIf** $\max(\phi(\mathbf{y}_i) - \epsilon, 0) = \max(\phi(\mathbf{x}_i) - \epsilon, 0) \wedge f(\mathbf{y}_i) < f(\mathbf{x}_i)$ **then**
   1) replace $\mathbf{x}_i$ with $\mathbf{y}_i$,
   2) $n_l = n_l + 1$, update $q_s, s = 1, 2, 3, 4$ according to Eq (13),
   3) store $|f(\mathbf{x}_i) - f(\mathbf{y}_i)|$, store F_i and CR_i.

Step 4: Population size update

Update N according to Eq (11).

Step 5: Memories update

Update M_F and M_{CR} for each strategy according to Eq (3) - (8).

Step 6: Epsilon update

a) Get the proportion of feasible solutions r_k in the current generation k.
b) **If** $r_k < \alpha \ \wedge \ FEs < T_c$ **then** $\epsilon = \epsilon(1 - FEs/T_c)^{cp}$,
c) **ElseIf** $r_k \geq \alpha \ \wedge \ FEs < T_c$ **then** $\epsilon = (1 + \tau)\phi_{max}$,
d) **ElseIf** $FEs \geq T_c$ **then** $\epsilon = 0$.

Step 7: Termination If stopping criteria are satisfied, output the best solution $\mathbf{x}_{best}$. Otherwise, go to **Step 2**.

When the PFS is larger than α, the population is considered to have lots of feasible solutions, and the ϵ gradually increases with the generation increasing, which makes the population search in the infeasible regions. Conversely, when the PFS is lower than α, the population is considered to have few feasible solutions, and the ϵ decreases with the generation increasing, which makes the population tend to search for feasible solutions.

C. The IEpsilon Embedded in LSHADE44

In this paper, the IEpsilon is embedded in the LSHADE44 to solve CSOPs, the pseudo-code of LSHADE44-IEpsilon is shown in Algorithm 2. The best solution $\mathbf{x}_{best}$ updated by each evaluation is the optimal result.

IV. EXPERIMENTAL STUDY

All 28 test instances defined in the report [2] are optimized by the LSHADE44-IEpsilon in this paper. Each instance is a single-objective optimization problem with some inequality or equality constraints. 25 independent runs are carried out for each problem with each kind of dimension levels ($D = 10,\ 30,\ 50,\ 100$). The maximal FEs is set as $MaxFEs = 20000D$. The experimental results are shown in Table I - IV. As defined in [2], 'Best', 'Median', 'Worst', 'Mean' and 'std' in tables denote the best, the median, the worst, the mean value and the standard deviation on the 25 runs. c is the sequence of three numbers, which indicates violated constraints of the median solution in ranges $[1.0, +\infty]$, $[0.01, 1.0]$ and $[0.0001, 0.01]$, respectively. $\overline{v}$ is the mean of all the constraints violation of the median solution. 'SR' is the feasibility rate of the solutions obtained in 25 runs. $\overline{vio}$ is the mean constraint violation value of all the solutions in 25 runs.

A. Experimental Settings

The parameter settings are summarized as follows:

1) Population size: the maximal population size $N_{max} = 5D$, the minimal size $N_{min} = 5$.
2) The length of historic memory: $H = 10$.
3) Parameters of strategy competition: $K = 4$, $n_0 = 2$, $\delta = 1/(5K) = 0.05$.
4) DE/current-to-$p$best/1 parameter: $p = 0.2$.
5) Parameters of IEpsilon: $cp = 2$, $\theta = 20\%$, $T_c = 0.8MaxFEs$, $\alpha = 0.5$, $\tau = 0.1$.

B. Experimental conditions

Experimental Conditions are shown in Table V:

TABLE V
PC CONFIGURE

System	Windows7 64bit
CPU	Intel(R) i7-6500U CPU @ 2.50GHz 2.50GHz
RAM	8GB
Language	Matlab (Matlab 2017a)
Algorithm	LSHADE44-IEpsilon

C. Algorithm Complexity

The algorithm complexities of the LSHADE44-IEpsilon on 4 kinds of dimension D are shown in Table VI in seconds, where $T_1 = (\sum_{i=1}^{28} t_{1i})/28$ and $T_2 = (\sum_{i=1}^{28} t_{2i})/28$. t_{1i} is the computing time of 10000 evaluations for problem i. t_{2i} is the complete computing time for the algorithm with 10000 evaluations for problem i.

TABLE VI
ALGORITHM COMPLEXITY OF THE LSHADE44-IEPSILON

D	T_1	T_2	$(T_2 - T_1)/T_1$
10	0.246258721	0.714261721	1.9005
30	0.30420195	0.842962546	1.7711
50	0.378859571	1.022363696	1.6985
100	0.771647804	2.026898707	1.6267

D. Comparison Among LSHADE44-IEpsilon and Four DE Algorithms

We compare LSHADE44-IEpsilon with four algorithms on the 28 benchmarks of CEC2017 competition on constrained real parameter optimization with 10, 30, 50 and 100 dimensions. These compared algorithms includes CAL-SHADE [15], LSHADE44+IDE [16], LSHADE44 [11] and UDE [17]. All the experimental results of these four algorithms come from the official website of CEC2018.

Two approaches for ranking two algorithms on one problem with 25 runs are suggested in the technique report [2] as follows.

- Ranking algorithms based on mean values;
- Ranking the algorithms based on the median solutions.

The summarized results on all benchmarks based on mean values and median value are shown in Table VII and TableVIII. The signs '+', '-' and '=' indicate the numbers of problems, on which the correlative algorithm performs better than, worse than, and not better or worse than the LSHADE44-IEpsilon, respectively.

According to [2], the rank value of each algorithm, which consists of rank values based on mean values and median solutions is shown in Table IX. The best algorithm will obtain the lowest rank value.

The experimental results show that the LSHADE44-IEpsilon has the best performance on these twenty-eight benchmarks for all $D \in \{10, 30, 50, 100\}$.

V. CONCLUSIONS

The paper proposes a new constraint-handling technique named IEpsilon for solving CSOPs. It utilizes the information of the proportion of feasible solutions in the current population to adjust the ϵ level adaptively, which has an ability to balance the search of the population between feasible and infeasible regions. The proposed LSHADE44-IEpsilon and other four algorithms are tested on the CEC2017 benchmarks with 10, 30, 50 and 100 dimensions for 25 independent runs, the experimental results show that the LSHADE44-IEpsilon outperforms

TABLE I
VALUES ACHIEVED FOR D = 10, ALL RESULTS FOR C01 - C28.

Problem	C01	C02	C03	C04	C05	C06	C07
Best	0	0	0	1.357E+01	0	0	-2.179E+02
Median	0	0	7.573E+01	1.357E+01	0	0	-1.197E+02
c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	0	0	0	0	0
Mean	0	0	5.931E+01	1.357E+01	0	1.690E+01	-1.205E+02
Worst	0	0	1.713E+02	1.357E+01	0	1.016E+02	-1.234E+01
std	0	0	4.990E+01	6.160E-05	0	4.732E+01	4.403E+01
SR	100%	100%	100%	100%	100%	60%	100%
$\overline{vio}$	0	0	0	0	0	2.071E-03	0
Problem	C08	C09	C10	C11	C12	C13	C14
Best	-1.348E-03	-4.975E-03	-5.096E-04	-1.688E-01	3.988E+00	0	2.376E+00
Median	-1.348E-03	-4.975E-03	-5.096E-04	-1.688E-01	3.988E+00	0	2.376E+00
c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	0	0	0	0	0
Mean	-1.348E-03	-4.975E-03	-5.096E-04	-1.688E-01	3.988E+00	2.816E+00	2.386E+00
Worst	-1.348E-03	-4.975E-03	-5.096E-04	-1.688E-01	3.989E+00	6.243E+01	2.623E+00
std	4.426E-19	2.656E-18	2.213E-19	7.986E-06	1.582E-04	1.247E+01	4.937E-02
SR	100%	100%	100%	100%	100%	100%	100%
$\overline{vio}$	0	0	0	0	0	0	0
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.356E+00	0	8.722E-02	3.660E+01	0	4.132E-01	3.988E+00
Median	2.356E+00	0	4.575E-01	3.660E+01	0	6.046E-01	3.989E+00
c	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	4.500E+00	0	6.634E+03	0	0
Mean	2.859E+00	0	5.276E-01	3.666E+01	0	6.179E-01	3.990E+00
Worst	5.501E+00	0	9.858E-01	3.822E+01	0	7.786E-01	3.992E+00
std	1.485E+00	0	4.682E-01	3.239E-01	0	9.810E-02	7.877E-04
SR	80%	100%	0	100%	0	100%	100%
$\overline{vio}$	2.751E-04	0	4.500E+00	0	6.634E+03	0	0
Problem	C22	C23	C24	C25	C26	C27	C28
Best	3.462E-27	2.376E+00	2.356E+00	0	1.252E+00	3.660E+01	1.495E+01
Median	3.462E-27	2.376E+00	2.356E+00	0	4.752E-01	3.660E+01	2.509E+01
c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	1, 0, 0
$\overline{v}$	0	0	0	0	5.500E+00	0	6.654E+03
Mean	1.116E+00	2.395E+00	2.859E+00	1.005E+00	7.730E-01	3.660E+01	2.876E+01
Worst	3.987E+00	2.629E+00	2.357E+00	6.283E+00	5.357E-01	3.660E+01	3.841E+01
std	1.827E+00	6.393E-02	1.485E+00	1.749E+00	3.634E-01	1.787E-04	6.493E+00
SR	100%	100%	84%	100%	0	100%	0
$\overline{vio}$	0	0	9.987E-05	0	5.020E+00	0	6.654E+03

TABLE VII
COMPARISON OF LSHADE44-IEPSILON WITH FOUR DE ALGORITHMS BASED ON MEAN VALUES FOR 25 INDEPENDENT RUNS

vs. LSHADE44-IEpsilon	Sign	$D = 10$	$D = 30$	$D = 50$	$D = 100$
CAL-SHADE	+	4	3	7	8
	-	17	20	18	19
	=	7	5	3	1
LSHADE44+IDE	+	8	7	6	10
	-	13	19	21	17
	=	7	2	1	1
LSHADE44	+	6	9	10	13
	-	15	15	18	15
	=	7	4	0	0
UDE	+	6	8	11	9
	-	16	17	17	18
	=	6	3	0	1

TABLE VIII
COMPARISON OF LSHADE44-IEPSILON WITH FOUR DE ALGORITHMS BASED ON MEDIAN VALUES FOR 25 INDEPENDENT RUNS

vs. LSHADE44-IEpsilon	Sign	$D = 10$	$D = 30$	$D = 50$	$D = 100$
CAL-SHADE	+	4	5	6	11
	-	14	15	16	15
	=	10	8	6	2
LSHADE44+IDE	+	5	7	7	8
	-	12	13	17	16
	=	11	8	4	4
LSHADE44	+	3	7	9	10
	-	15	13	13	14
	=	10	8	6	4
UDE	+	2	10	9	8
	-	13	10	15	16
	=	13	8	4	4

TABLE IX
RANK VALUES OF LSHADE44-IEPSILON AND FOUR COMPARED DE ALGORITHMS FOR 25 INDEPENDENT RUNS

	$D = 10$	$D = 30$	$D = 50$	$D = 100$
LSHADE44-IEpsilon	103	117	127	134
CAL-SHADE	171	179	195	199
LSHADE44+IDE	132	173	182	156
LSHADE44	144	139	148	140
UDE	123	143	160	172

the compared algorithms, which manifests that LSHADE44-IEpsilon is a quite competitive algorithm for solving CSOPs of CEC2017 benchmarks.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant (61175073, 61300159, 61332002, 51375287), the Guangdong Key Laboratory of Digital Signal and Image Processing, the Science and Technology Planning Project of Guangdong Province (2013B011304002) and the Project of Educational Commission of Guangdong Province, China 2015KGJHZ014).

REFERENCES

[1] C. A. Floudas and P. M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Springer Berlin Heidelberg, 1990.

TABLE II
VALUES ACHIEVED FOR D = 30, ALL RESULTS FOR C01 - C28.

Problem	C01	C02	C03	C04	C05	C06	C07
Best	0	0	1.405E+03	1.357E+01	0	6.159E+02	-4.557E+02
Median	3.020E-30	0	4.873E+03	1.357E+01	5.237E-28	4.515E+02	-2.869E+02
c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 6	0, 0, 0
$\overline{v}$	0	0	0	0	0	1.450E-03	0
Mean	5.705E-30	6.063E-30	7.192E+03	1.357E+01	6.556E-27	4.923E+02	-2.772E+02
Worst	3.457E-29	3.704E-29	2.830E+04	1.357E+01	7.467E-26	5.083E+02	-1.715E+02
std	8.052E-30	8.640E-30	6.339E+03	3.626E-15	1.595E-26	1.083E+02	5.858E+01
SR	100%	100%	100%	100%	100%	0	100%
$\overline{vio}$	0	0	0	0	0	1.839E-03	0
Problem	C08	C09	C10	C11	C12	C13	C14
Best	-2.840E-04	-2.666E-03	-1.028E-04	-1.352E+02	3.983E+00	0	1.409E+00
Median	-2.840E-04	-2.666E-03	-1.028E-04	-5.920E+02	3.983E+00	8.060E+01	1.409E+00
c	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	0	2.082E+01	0	0	0
Mean	-2.840E-04	-2.666E-03	-1.028E-04	-7.690E+02	3.987E+00	5.163E+01	1.412E+00
Worst	-2.840E-04	-2.666E-03	-1.028E-04	-2.006E+03	3.996E+00	8.114E+01	1.495E+00
std	1.397E-09	8.852E-19	1.059E-10	5.721E+02	4.763E-03	3.952E+01	1.738E-02
SR	100%	100%	100%	0	100%	100%	100%
$\overline{vio}$	0	0	0	5.701E+01	0	0	0
Problem	C15	C16	C17	C18	C19	C20	C21
Best	2.356E+00	6.283E+00	9.155E-01	3.652E+01	0	1.644E+00	3.983E+00
Median	5.498E+00	6.283E+00	8.998E-01	3.652E+01	0	1.930E+00	9.776E+00
c	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	1.550E+01	0	2.137E+04	0	0
Mean	4.995E+00	6.283E+00	8.728E-01	3.652E+01	0	1.935E+00	1.062E+01
Worst	8.639E+00	6.283E+00	9.505E-01	3.653E+01	0	2.291E+00	2.834E+01
std	1.740E+00	5.859E-08	1.179E-01	4.573E-03	0	1.663E-01	8.462E+00
SR	100%	100%	0	100%	0	100%	100%
$\overline{vio}$	0	0	1.550E+01	0	2.137E+04	0	0
Problem	C22	C23	C24	C25	C26	C27	C28
Best	2.153E+01	1.409E+00	2.356E+00	1.414E+01	9.839E-01	3.652E+01	1.524E+02
Median	1.380E+03	1.409E+00	5.498E+00	2.513E+01	8.115E-01	3.652E+01	1.325E+02
c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	1, 0, 0
$\overline{v}$	0	0	0	0	1.550E+01	0	2.149E+04
Mean	3.731E+03	1.409E+00	6.377E+00	2.551E+01	9.552E-01	3.884E+01	1.560E+02
Worst	2.996E+04	1.409E+00	8.639E+00	5.027E+01	1.014E+00	5.113E+01	1.444E+02
std	6.529E+03	1.214E-04	1.701E+00	8.191E+00	7.080E-02	4.437E+00	2.212E+01
SR	100%	100%	100%	100%	0	100%	0
$\overline{vio}$	0	0	0	0	1.550E+01	0	2.149E+04

[2] G. Wu, R. Mallipeddi, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 competition on constrained real-parameter optimization," *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2016.

[3] R. Storn and K. Price, "Differential evolution c a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[4] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2011.

[5] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[6] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE transactions on evolutionary computation*, vol. 10, no. 6, pp. 646–657, 2006.

[7] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation*, vol. 13, no. 5, pp. 945–958, 2009.

[8] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[9] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 71–78.

[10] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1658–1665.

[11] R. Poláková, "L-shade with competing strategies applied to constrained optimization," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1683–1689.

[12] A. R. Jordehi, "A review on constraint handling strategies in particle swarm optimisation," *Neural Computing and Applications*, vol. 26, no. 6, pp. 1265–1275, 2015.

[13] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[14] T. Takahama, S. Sakai, and N. Iwane, "Solving nonlinear constrained optimization problems by the ε constrained differential evolution," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, vol. 3. IEEE, 2006, pp. 2322–2327.

[15] A. Zamuda, "Adaptive constraint handling and success history differential evolution for cec 2017 constrained real-parameter optimization," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 2443–2450.

[16] J. Tvrdik and R. Polakova, "A simple framework for constrained problems with application of l-shade44 and ide," in *IEEE Congress on Evolutionary Computation*, 2017, pp. 1436–1443.

[17] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," in *Evolutionary Computation*, 2017, pp. 1231–1238.

[18] J. Tvrdık, "Competitive differential evolution," in *MENDEL*, 2006, pp. 7–12.

[19] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 22–34, 1999.

[20] T. Liao, D. Molina, M. A. M. D. Oca, and T. Sttzle, "A note on bound constraints handling for the ieee cec05 benchmark function suite," *Evolutionary Computation*, vol. 22, no. 2, pp. 351–359, 2014.

[21] T. Takahama and S. Sakai, "Constrained optimization by the constrained differential evolution with an archive and gradient-based mutation," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–9.

TABLE III
VALUES ACHIEVED FOR D = 50, ALL RESULTS FOR C01 - C28.

Problem	C01	C02	C03	C04	C05	C06	C07
Best	1.825E-26	4.065E-26	9.038E+03	1.357E+01	2.241E-26	1.118E+03	-5.343E+02
Median	4.269E-25	3.729E-25	1.635E+04	1.357E+01	3.743E-23	8.252E+02	-3.072E+02
c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 5	0, 0, 0
$\overline{v}$	0	0	0	0	0	7.634E-04	0
Mean	1.620E-24	6.542E-25	2.434E+04	1.357E+01	3.189E-01	9.761E+02	-3.221E+02
Worst	9.429E-24	4.151E-24	6.060E+04	1.359E+01	3.987E+00	1.175E+03	-2.267E+02
std	2.766E-24	9.292E-25	1.719E+04	2.802E-03	1.104E+00	1.690E+02	7.019E+01
SR	100%	100%	100%	100%	100%	0	100%
$\overline{vio}$	0	0	0	0	0	9.380E-04	0
Problem	C08	C09	C10	C11	C12	C13	C14
Best	-1.321E-04	-2.037E-03	-4.824E-05	-2.371E+03	1.794E+01	1.325E-10	1.100E+00
Median	-1.141E-04	-2.037E-03	-4.793E-05	-2.782E+03	2.045E+01	3.961E-08	1.100E+00
c	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	0	1.007E+02	0	0	0
Mean	-1.099E-04	-1.735E-03	-4.786E-05	-2.852E+03	2.735E+01	1.192E+01	1.104E+00
Worst	-4.987E-05	1.164E-04	-4.712E-05	-3.686E+03	6.218E+01	9.939E+01	1.197E+00
std	1.881E-05	6.038E-04	2.873E-07	3.452E+02	1.312E+01	3.295E+01	1.938E-02
SR	100%	100%	100%	0	100%	100%	100%
$\overline{vio}$	0	0	0	1.085E+02	0	0	0
Problem	C15	C16	C17	C18	C19	C20	C21
Best	5.498E+00	1.257E+01	1.033E+00	3.647E+01	0	3.180E+00	3.981E+00
Median	8.639E+00	1.885E+01	1.034E+00	3.649E+01	0	3.487E+00	7.069E+00
c	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	2.550E+01	0	3.612E+04	0	0
Mean	8.765E+00	1.684E+01	1.018E+00	3.678E+01	0	3.503E+00	1.141E+01
Worst	1.492E+01	2.513E+01	1.008E+00	4.322E+01	0	3.813E+00	3.642E+01
std	1.920E+00	3.888E+00	1.706E-02	1.349E+00	0	1.763E-01	9.148E+00
SR	100%	100%	0	100%	0	100%	100%
$\overline{vio}$	0	0	2.550E+01	0	3.612E+04	0	0
Problem	C22	C23	C24	C25	C26	C27	C28
Best	1.384E+04	1.100E+00	8.639E+00	5.027E+01	1.047E+00	3.647E+01	2.818E+02
Median	1.475E+05	1.100E+00	8.639E+00	7.540E+01	1.039E+00	3.648E+01	2.719E+02
c	1, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	1, 0, 0
$\overline{v}$	1.509E+01	0	0	0	2.550E+01	0	3.632E+04
Mean	1.256E+05	1.100E+00	9.268E+00	7.892E+01	1.033E+00	4.032E+01	2.793E+02
Worst	3.672E+05	1.102E+00	1.178E+01	1.131E+02	1.040E+00	5.620E+01	3.146E+02
std	7.199E+04	4.200E-04	1.283E+00	1.955E+01	7.981E-03	6.536E+00	2.154E+01
SR	8%	100%	100%	100%	0	100%	0
$\overline{vio}$	1.811E+01	0	0	0	2.550E+01	0	3.632E+04

TABLE IV
VALUES ACHIEVED FOR D = 100, ALL RESULTS FOR C01 - C28.

Problem	C01	C02	C03	C04	C05	C06	C07
Best	1.899E-08	1.056E-08	6.069E+04	1.362E+01	7.721E-05	2.997E+03	-7.120E+02
Median	1.632E-07	5.820E-08	1.509E+05	1.415E+01	4.161E+00	2.661E+03	-4.698E+02
c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 6	0, 0, 0
$\overline{v}$	0	0	0	0	0	5.032E-04	0
Mean	1.896E-07	1.034E-07	1.664E+05	1.441E+01	4.401E+00	3.001E+03	-4.771E+02
Worst	4.506E-07	4.845E-07	4.178E+05	1.592E+01	9.909E+00	3.112E+03	-2.979E+02
std	1.135E-07	1.101E-07	7.712E+04	7.528E-01	3.363E+00	3.385E+02	9.823E+01
SR	100%	100%	100%	100%	100%	0	100%
$\overline{vio}$	0	0	0	0	0	5.753E-04	0
Problem	C08	C09	C10	C11	C12	C13	C14
Best	1.040E-03	0.000E+00	9.066E-05	-7.235E+03	9.997E+00	6.133E+01	7.842E-01
Median	1.693E-03	1.552E-03	1.497E-04	-7.309E+03	1.886E+01	6.861E+01	8.172E-01
c	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	0	9.755E+01	0	0	0
Mean	1.678E-03	4.181E-01	1.531E-04	-7.410E+03	1.884E+01	6.787E+01	8.359E-01
Worst	2.123E-03	2.556E+00	2.527E-04	-8.380E+03	3.159E+01	7.349E+01	9.500E-01
std	2.893E-04	7.813E-01	3.680E-05	2.454E+02	9.514E+00	3.140E+00	5.414E-02
SR	100%	100%	100%	0	100%	100%	100%
$\overline{vio}$	0	0	0	1.072E+02	0	0	0
Problem	C15	C16	C17	C18	C19	C20	C21
Best	1.492E+01	1.634E+02	1.095E+00	3.638E+01	0	8.021E+00	3.981E+00
Median	1.492E+01	1.838E+02	1.097E+00	3.722E+01	0	8.859E+00	9.997E+00
c	0, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	1, 0, 0	0, 0, 0	0, 0, 0
$\overline{v}$	0	0	5.050E+01	0	7.297E+04	0	0
Mean	1.618E+01	1.880E+02	1.097E+00	3.786E+01	0	8.830E+00	9.344E+00
Worst	1.806E+01	2.325E+02	1.097E+00	4.222E+01	0	9.605E+00	1.886E+01
std	1.571E+00	1.737E+01	1.642E-03	1.822E+00	0	4.482E-01	5.362E+00
SR	100%	100%	0	100%	0	100%	100%
$\overline{vio}$	0	0	5.050E+01	0	7.297E+04	0	0
Problem	C22	C23	C24	C25	C26	C27	C28
Best	4.146E+05	7.857E-01	1.806E+01	4.650E+02	1.094E+00	3.638E+01	5.796E+02
Median	8.302E+05	7.882E-01	1.806E+01	5.042E+02	1.097E+00	7.191E+02	6.135E+02
c	1, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	1, 0, 0	1, 0, 0
$\overline{v}$	1.293E+02	0	0	0	5.050E+01	4.053E+01	7.341E+04
Mean	7.874E+05	7.928E-01	1.806E+01	5.034E+02	1.097E+00	3.834E+02	6.152E+02
Worst	1.015E+06	8.188E-01	1.806E+01	5.419E+02	1.097E+00	1.271E+03	6.188E+02
std	2.614E+05	9.415E-03	1.400E-06	1.950E+01	2.092E-03	4.275E+02	2.696E+01
SR	0	100%	100%	100%	0	20%	0
$\overline{vio}$	1.326E+02	0	0	0	5.050E+01	6.497E+01	7.341E+04