Constrained Optimization by the ε Constrained Differential Evolution with an Archive and Gradient-Based Mutation

Tetsuyuki Takahama, *Member, IEEE*, and Setsuko Sakai, *Member, IEEE*

***Abstract*— The ε constrained method is an algorithm transformation method, which can convert algorithms for unconstrained problems to algorithms for constrained problems using the ε level comparison, which compares search points based on the pair of objective value and constraint violation of them. We have proposed the ε constrained differential evolution (εDE), which is the combination of the ε constrained method and differential evolution (DE). It has been shown that the εDE can run very fast and can find very high quality solutions. Also, we proposed the εDE with gradient-based mutation (εDEg), which utilized gradients of constraints in order to solve problems with difficult constraints. In this study, we propose the ε constrained DE with an archive and gradient-based mutation (εDEag). The εDEag utilizes an archive to maintain the diversity of individuals and adopts a new way of selecting the ε level control parameter in the εDEg. The 18 problems, which are given in special session on "Single Objective Constrained Real-Parameter Optimization" in CEC2010, are solved by the εDEag and the results are shown in this paper.**

I. Introduction

Constrained optimization problems, especially nonlinear optimization problems, where objective functions are minimized under given constraints, are very important and frequently appear in the real world. There exist many studies on solving constrained optimization problems using evolutionary algorithms (EAs) [1]–[3]. EAs basically lack a mechanism to incorporate the constraints of a given problem in the fitness value of individuals. Thus, many studies have been dedicated to handle the constraints in EAs. In most successful constraint-handling techniques, the objective function value and the sum of constraint violations, or the constraint violation, are separately handled and an optimal solution is searched with balancing the optimization of the function value and the optimization of the constraint violation.

The ε constrained differential evolution (εDE) adopted one of such techniques called the *ε constrained method* and also adopted differential evolution (DE) as an optimization engine. The εDE can solve constrained problems successfully and stably [4]–[6], including engineering design problems [7]. Also, we proposed the εDE with gradient-based mutation (εDEg) [8], [9], which utilized gradients of constraints in order to solve problems with difficult constraints. The ε constrained method [5] is an algorithm transformation method, which can convert algorithms for unconstrained problems to algorithms for constrained problems using the ε level comparison, which compares search points based on the pair of objective value and constraint violation of them.

In this study, we propose the ε constrained DE with an archive and gradient-based mutation (εDEag) in order to improve the stability, usability, and efficiency of the εDEg as follows:

(1) In order to improve the stability, it is effective that all individuals approach to an optimal solution at a similar speed while maintaining the diversity of individuals. To attain this speed, a simple modification to DE is introduced: When a parent generates a child and the child is not better than the parent, the parent can generate another child. This modification will average the approaching speed of individuals. In addition, large population size is effective to the diversity, although the efficiency of evolution becomes low. In order to keep the diversity and efficiency, we utilized an archive which has many individuals initially and is updated using defeated individuals in survivor selection of DE.

(2) In the εDE, users need to decide whether enabling or disabling the control of the ε level which keeps a balance between the optimization of the function value and the optimization of the constraint violation. Also, users need to decide a parameter value for controlling the ε level. In this study, we propose a simple scheme for setting the parameter value automatically based on the state of an initial archive. By introducing this scheme, users can always enable the control of the ε level and need not to specify the parameter value. Thus, the usability is greatly improved.

(3) In order to improve the efficiency, the gradient-based mutation and the control of the ε level are slightly modified. In the gradient-based mutation, skip-move and long-move are introduced only when the number of violated constraints is one. In the final time for controlling the ε level, the ε level is increased in order to enlarge searching region and find individuals with better objective values.

In Section II, constrained optimization methods are briefly reviewed. The ε constrained method is defined in Section III. The εDEag is explained in Section IV. In Section V, experimental results on 18 constrained problems are shown. Finally, conclusions are described in Section VI.

II. Previous works

EAs for constrained optimization can be classified into several categories according to the way the constraints are treated as follows:

(1) Constraints are only used to see whether a search point is feasible or not. Approaches in this category are usually called death penalty methods. In this category, generating initial

T. Takahama is with the Department of Intelligent Systems, Hiroshima City University, Asaminami-ku, Hiroshima, 731-3194 Japan (e-mail: takahama@info.hiroshima-cu.ac.jp).

S. Sakai is with the Faculty of Commercial Sciences, Hiroshima Shudo University, Asaminami-ku, Hiroshima, 731-3195 Japan (e-mail: setuko@shudo-u.ac.jp).

feasible points is difficult and computationally demanding when the feasible region is very small.
(2) The constraint violation, which is the sum of the violation of all constraint functions, is combined with the objective function. The penalty function method is in this category [10]–[13]. The main difficulty of the method is the selection of an appropriate value for the penalty coefficient that adjusts the strength of the penalty. In order to solve the difficulty, some methods, where a kind of the penalty coefficient is adaptively controlled [14], [15], are proposed.
(3) The constraint violation and the objective function are used separately. In this category, both the constraint violation and the objective function are optimized by a lexicographic order in which the constraint violation precedes the objective function. Deb [16] proposed a method that adopted the extended objective function, which realized the lexicographic ordering. Takahama and Sakai proposed the α constrained method [17], and ε constrained method [18] that adopted a lexicographic ordering with relaxation of the constraints. Runarsson and Yao [19] proposed the stochastic ranking method that adopted the stochastic lexicographic order, which ignored the constraint violation with some probability. Mezura-Montes and Coello [20] proposed a comparison mechanism that was equivalent to the lexicographic ordering. Venkatraman and Yen [21] proposed a two-step optimization method, which first optimized constraint violation and then objective function. These methods were successfully applied to various problems.
(4) The constraints and the objective function are optimized by multiobjective optimization methods. In this category, the constrained optimization problems are solved as the multiobjective optimization problems in which the objective function and the constraint functions are objectives to be optimized [22]–[28]. But in many cases, solving multiobjective optimization problems is a more difficult and expensive task than solving single objective optimization problems.
(5) Hybridization methods. In this category, constrained problems are solved by combining some of above mentioned methods. Mallipeddi and Suganthan [29] proposed a hybridization of the methods in category (2), (3) and (4).

It has been shown that the methods in the category (3) have better performance than methods in other categories in many benchmark problems. Especially, the ε constrained method is quite unique approach to constrained optimization. The method can convert an algorithm for unconstrained optimization into an algorithm for constrained optimization by replacing the ordinal comparisons with the ε level comparisons in direct search methods. The method has been applied various algorithms such as PSO and GA, and the εPSO [18] and the hybrid algorithm of the εPSO and εGA [30] are proposed. It has been shown that the method has general-purpose properties.

III. The ε constrained method

A. Constrained Optimization Problems

In this study, the following optimization problem (P) with inequality constraints, equality constraints, upper bound constraints and lower bound constraints will be discussed.

$$\begin{array}{lll} \text{(P)} & \text{minimize} & f(\boldsymbol{x}) \\ & \text{subject to} & g_j(\boldsymbol{x}) \le 0,\ j=1,\ldots,q \\ & & h_j(\boldsymbol{x}) = 0,\ j=q+1,\ldots,m \\ & & l_i \le x_i \le u_i,\ i=1,\ldots,n, \end{array} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is an n dimensional vector, $f(\boldsymbol{x})$ is an objective function, $g_j(\boldsymbol{x}) \le 0$ and $h_j(\boldsymbol{x}) = 0$ are q inequality constraints and $m - q$ equality constraints, respectively. Functions f, g_j and h_j are linear or nonlinear real-valued functions. Values u_i and l_i are the upper bound and the lower bound of x_i, respectively. Also, let the feasible space in which every point satisfies all constraints be denoted by $\mathcal{F}$ and the search space in which every point satisfies the upper and lower bound constraints be denoted by $\mathcal{S}\ (\supset \mathcal{F})$.

B. Constraint violation and ε level comparisons

In the ε constrained method, constraint violation $\phi(\boldsymbol{x})$ is defined. The constraint violation can be given by the maximum of all constraints or the sum of all constraints.

$$\phi(\boldsymbol{x}) = \max\{\max_j\{0, g_j(\boldsymbol{x})\}, \max_j |h_j(\boldsymbol{x})|\} \tag{2}$$

$$\phi(\boldsymbol{x}) = \sum_j || \max\{0, g_j(\boldsymbol{x})\}||^p + \sum_j ||h_j(\boldsymbol{x})||^p \tag{3}$$

where p is a positive number.

The ε *level comparisons* are defined as an order relation on a pair of objective function value and constraint violation $(f(\boldsymbol{x}), \phi(\boldsymbol{x}))$. If the constraint violation of a point is greater than 0, the point is not feasible and its worth is low. The ε level comparisons are defined basically as a lexicographic order in which $\phi(\boldsymbol{x})$ precedes $f(\boldsymbol{x})$, because the feasibility of $\boldsymbol{x}$ is more important than the minimization of $f(\boldsymbol{x})$. This precedence can be adjusted by the parameter ε.

Let $f_1\,(f_2)$ and $\phi_1\,(\phi_2)$ be the function values and the constraint violation at a point $\boldsymbol{x}_1\,(\boldsymbol{x}_2)$, respectively. Then, for any ε satisfying $\varepsilon \ge 0$, ε level comparisons $<_\varepsilon$ and $\le_\varepsilon$ between (f_1, ϕ_1) and (f_2, ϕ_2) are defined as follows:

$$(f_1, \phi_1) <_\varepsilon (f_2, \phi_2) \Leftrightarrow \begin{cases} f_1 < f_2, & \text{if } \phi_1, \phi_2 \le \varepsilon \\ f_1 < f_2, & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \tag{4}$$

$$(f_1, \phi_1) \le_\varepsilon (f_2, \phi_2) \Leftrightarrow \begin{cases} f_1 \le f_2, & \text{if } \phi_1, \phi_2 \le \varepsilon \\ f_1 \le f_2, & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \tag{5}$$

In case of ε=∞, the ε level comparisons $<_\infty$ and $\le_\infty$ are equivalent to the ordinary comparisons $<$ and $\le$ between function values. Also, in case of $\varepsilon = 0$, $<_0$ and $\le_0$ are equivalent to the lexicographic orders in which the constraint violation $\phi(\boldsymbol{x})$ precedes the function value $f(\boldsymbol{x})$.

C. The properties of the ε constrained method

The ε constrained method converts a constrained optimization problem into an unconstrained one by replacing the order relation in direct search methods with the ε level comparison. An optimization problem solved by the ε constrained method, that is, a problem in which the ordinary comparison is replaced with the ε level comparison, ($\mathrm{P}_{\leq_\varepsilon}$), is defined as follows:

$$(\mathrm{P}_{\leq_\varepsilon}) \quad \text{minimize}_{\leq_\varepsilon} \quad f(\boldsymbol{x}), \tag{6}$$

where minimize$_{\leq_\varepsilon}$ denotes the minimization based on the ε level comparison $\leq_\varepsilon$. Also, a problem (P^ε) is defined such that the constraints of (P), that is, $\phi(\boldsymbol{x}) = 0$, is relaxed and replaced with $\phi(\boldsymbol{x}) \leq \varepsilon$:

$$\begin{array}{lll} (\mathrm{P}^\varepsilon) & \text{minimize} & f(\boldsymbol{x}) \\ & \text{subject to} & \phi(\boldsymbol{x}) \leq \varepsilon \end{array} \tag{7}$$

It is obvious that (P^0) is equivalent to (P). It is shown that ($\mathrm{P}_{\leq_\varepsilon}$) is equivalent to ($\mathrm{P}^\varepsilon$) under proper conditions and an optimal solution of (P^0) can be given by converging ε to 0 as well as by increasing the penalty coefficient to infinity in the penalty method [18].

IV. The εDEag

In this section, differential evolution is described first and then the εDEag is defined.

A. Differential Evolution

Differential evolution (DE) is proposed by Storn and Price [31]. DE is a stochastic direct search method using population or multiple search points. DE has been successfully applied to the optimization problems including non-linear, non-differentiable, non-convex and multi-modal functions. It has been shown that DE is fast and robust to these functions.

There are some variants of DE that have been proposed, such as DE/rand/1/exp. The variants are classified using the notation DE/*base*/*num*/*cross*. "*base*" indicates the method of selecting an individual that will form the base vector. For example, DE/rand selects an individual for the base vector at random from the population. DE/best selects the best individual in the population.

In DE/rand/1, for each individual $\boldsymbol{x}^i$, three individuals $\boldsymbol{x}^{p1}$, $\boldsymbol{x}^{p2}$ and $\boldsymbol{x}^{p3}$ are chosen from the population without overlapping $\boldsymbol{x}^i$ and each other. A new vector, or a mutant vector $\boldsymbol{x}'$ is generated by the base vector $\boldsymbol{x}^{p1}$ and the difference vector $\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}$, where F is a scaling factor.

$$\boldsymbol{x}' = \boldsymbol{x}^{p1} + F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}) \tag{8}$$

"*num*" indicates the number of difference vectors used to perturb the base vector. "*cross*" indicates the crossover mechanism used to create a child. For example, 'bin' shows that the crossover is controlled by binomial crossover using constant crossover rate, and 'exp' shows that the crossover is controlled by a kind of two-point crossover using exponentially decreasing the crossover rate.

Fig. 1 shows exponential crossover. A new child $\boldsymbol{x}^{\text{child}}$ is generated from the parent $\boldsymbol{x}^i$ and the mutant vector $\boldsymbol{x}'$, where CR is a crossover rate.

```
k=1; j=randint(1,n);
do {
    $x_j^{\text{child}}$=$x'_j$;
    k=k+1; j=$(j+1)\%n$;
} while($k \leq n$ && $u(0,1) < CR$);
while($k \leq n$) {
    $x_j^{\text{child}}$=$x_j^i$;
    k=k+1; j=$(j+1)\%n$;
}
```

Fig. 1. Exponential crossover operation, where randint(1,n) generates an integer randomly from $[1,n]$.

B. The algorithm of the εDEag

The following ideas are introduced in the εDEag: (1) An archive is introduced to maintain the diversity of individuals. The archive is similar to the archive proposed in [32]. (2) A way of offspring generation is adopted to average the approaching speed to an optimal solution: When a parent generates a child and the child is not better than the parent, the parent can generate another child. These two ideas will improve the stability and the efficiency of DE. (3) The ε constrained method with automatic selection of the ε level control parameter is adopted in order to improve the usability of the εDE. (4) Gradient-based mutation is adopted to solve problems with difficult constraints.

The algorithm of the εDEag is as follows:

Step1 Initialization of an archive. Initial M individuals $A = \{\boldsymbol{x}^i, i = 1, 2, \cdots, M\}$ are generated randomly in search space $\mathcal{S}$ and form an archive.

Step2 Initialization of the ε level. An initial ε level is given by the ε level control function $\varepsilon(0)$.

Step3 Initialization of a population. Top N individuals are selected from the archive A and form a population $P = \{\boldsymbol{x}^i\}$. The rank of individuals are defined by the ε level comparison.

Step4 Termination condition. If the number of function evaluations exceeds the maximum number of evaluation $FE_{\max}$, the algorithm is terminated.

Step5 DE operations (at most twice). Each individual $\boldsymbol{x}^i$ is selected as a parent. If all individuals are selected, go to Step7. DE/rand/1/exp operation is applied and a new child $\boldsymbol{x}^{\text{child}}$ is generated. A fixed scaling factor F_0 and a fixed crossover rate CR_0 are used with probability 0.95, and the randomly generated scaling factor F and the crossover rate CR_0 are used with the probability 0.05. The third vector $\boldsymbol{x}^{p3}$ is selected from the archive A with probability 0.95 and selected from the population P with probability 0.05. If the new one is better than the parent based on the ε level comparison, the parent $\boldsymbol{x}_i$ is immediately replaced by the trial vector $\boldsymbol{x}^{\text{child}}$ and go to Step 6. In the εDEag, not discrete generation model but continuous generation model is adopted. Otherwise, the same operation is applied to the parent only once again.

Step6 Gradient-based mutation. If $\boldsymbol{x}^{\text{child}}$ is not feasible, or $\phi(\boldsymbol{x}^{\text{child}}) > 0$, the child is changed by the gradient-based mutation with probability P_g until the number of the changes becomes R_g or $\boldsymbol{x}^{\text{child}}$ becomes feasible. Go back to Step5 and the next individual is selected as a parent.

Step7 Control of the ε level. The ε level is updated by the ε level control function $\varepsilon(t)$.

Step7 Go back to Step4.

Fig. 2 shows the algorithm of the εDEag.

```
$\varepsilon$DEag()
{
 $F$=$F_0$; $CR$=$CR_0$;
// Initialize an archive
 $A$=$M$ individuals generated randomly in $\mathcal{S}$;
 $FE$=$M$;
// Initialize the $\varepsilon$ level
 $\varepsilon$=$\varepsilon(0)$;
// Initialize a population
 $P$=Top $N$ individuals $\{\boldsymbol{x}^i\}$ from $A$ using $<_\varepsilon$;
 $A$=$A$-$P$;
 for($t$=1; $FE \le FE_{max}$; $t$++) {
  $F$=$F_0$;
  if($t > 0.95T_C$ && $t < T_C$)
    Modify the $\varepsilon$ level and $F$ using Eq.(12),(13);
  if($u(0,1)$<0.05)
    $F$=1+$|rand_G(0,0.05)|$, $F$ is truncated to 1.1;
  for($i$=1; $i \le N$; $i$++) {
// DE operation
   for($k$=1; $k \le 2$; $k$++) {
     $\boldsymbol{x}^{p1}$=Randomly selected from $P$;
     $\boldsymbol{x}^{p2}$=Randomly selected from $P$;
     if($u(0,1)$<0.05)
       $\boldsymbol{x}^{p3}$=Randomly selected from $P$;
     else
       $\boldsymbol{x}^{p3}$=Randomly selected from $P \cup A$;
     $\boldsymbol{x}'$=$\boldsymbol{x}^{p1}$+$(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3})$;
     $\boldsymbol{x}^{\text{child}}$=trial vector is generated
          from $\boldsymbol{x}^i$ and $\boldsymbol{x}'$ by exponential crossover;
// Gradient-based mutation
     if($t$%$n$==0 && $u(0,1) < P_g$)
       for($h$=1; $h \le R_g$ && $\phi(\boldsymbol{x}^{\text{child}})$>0; $h$++) {
         Apply gradient-based mutation to $\boldsymbol{x}^{\text{child}}$;
         $FE$=$FE$+$n$+1;
       }
     else $FE$=$FE$+1;
     if$\big((f(\boldsymbol{x}^{\text{child}}), \phi(\boldsymbol{x}^{\text{child}})) <_\varepsilon (f(\boldsymbol{x}^i), \phi(\boldsymbol{x}^i))\big)$ {
       $\boldsymbol{x}^i$=$\boldsymbol{x}^{\text{child}}$;
       break;
     } else {
       $\boldsymbol{x}^A$=Randomly selected from $A$;
       $A$=$A$+$\{\boldsymbol{x}^{\text{child}}\}$-$\{\boldsymbol{x}^A\}$;
     }
   }
  }
// Control the $\varepsilon$ level
  $\varepsilon$=$\varepsilon(t)$;
 }
}
```

Fig. 2. The algorithm of the εDEag, where $\varepsilon(t)$ is the ε level control function, FE is the number of function evaluations, $u(l, r)$ is a uniform random number generator in $[l, r]$, $rand_G(\mu, \sigma)$ is a random number generator obeying Gaussian distribution with the mean μ and the standard deviation σ, and P_g and R_g are parameters for gradient-based mutation.

C. Controlling the ε level

Usually, the ε level is controlled according to the equation (9). The initial ε level $\varepsilon(0)$ is the constraint violation of the top θ-th individual in the initial search points. The ε level is updated until the number of iterations t becomes the control generation T_c. After the number of iterations exceeds T_c, the ε level is set to 0 to obtain solutions with minimum constraint violation.

$$\begin{aligned}\varepsilon(0) &= \phi(\boldsymbol{x}_\theta) \\ \varepsilon(t) &= \begin{cases} \varepsilon(0)(1-\frac{t}{T_c})^{cp}, & 0 < t < T_c, \\ 0, & t \ge T_c \end{cases}\end{aligned} \tag{9}$$

where $\boldsymbol{x}_\theta$ is the top θ-th individual. If $\theta > 1$, $\phi(\boldsymbol{x}_\theta) = \theta \times \max_{\boldsymbol{x}} \phi(\boldsymbol{x})$.

In this study, a simple scheme of setting the parameter value *cp* is proposed: The ε level is adjusted to be a small value $\varepsilon_\lambda = 10^{-5}$ at the generation $T_\lambda = 0.95T_c$.

$$\varepsilon(T_\lambda) = \varepsilon(0)(1 - T_\lambda/T_c)^{cp} = \varepsilon_\lambda \tag{10}$$

$$\begin{aligned} cp &= (\log \varepsilon_\lambda - \log \varepsilon(0))/\log(1 - T_\lambda/T_c) \quad (11) \\ &= (-5 - \log \varepsilon(0))/\log 0.05 \end{aligned}$$

To avoid too small value of *cp*, if *cp* is less than $cp_{\min}$, *cp* is to be $cp_{\min}$ where $cp_{\min} = 3$. Also, after T_λ, in order to increase the ε level, to enlarge ε-feasible region and to search better objective values, *cp* is decreased and F is increased as follows:

$$cp = 0.3cp + 0.7cp_{\min} \tag{12}$$

$$F = 0.3F_0 + 0.7 \tag{13}$$

D. Gradient-based Mutation

The gradient-based mutation [8] is an operation similar to the gradient-based repair method proposed by Chootinan and Chen [33]. The vector of constraint functions $C(\boldsymbol{x})$, the vector of constraint violations $\Delta C(\boldsymbol{x})$ and the increment of a point $\boldsymbol{x}$ to satisfy constraints $\Delta\boldsymbol{x}$ are defined as follows:

$$C(\boldsymbol{x})=(g_1(\boldsymbol{x}) \cdots g_q(\boldsymbol{x})\, h_{q+1}(\boldsymbol{x}) \cdots h_m(\boldsymbol{x}))^T \tag{14}$$

$$\Delta C(\boldsymbol{x})=(\Delta g_1(\boldsymbol{x}) \cdots \Delta g_q(\boldsymbol{x})\, h_{q+1}(\boldsymbol{x}) \cdots h_m(\boldsymbol{x}))^T \tag{15}$$

$$\nabla C(\boldsymbol{x})\Delta\boldsymbol{x}=-\Delta C(\boldsymbol{x}) \tag{16}$$

$$\Delta\boldsymbol{x}=-\nabla C(\boldsymbol{x})^{-1}\Delta C(\boldsymbol{x}) \tag{17}$$

where $\Delta g_j(\boldsymbol{x}) = \max\{0, g_j(\boldsymbol{x})\}$. And $\nabla C(\boldsymbol{x})$ is the gradient matrix of $C(\boldsymbol{x})$.

$$\nabla C(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial g_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial g_1(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial g_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_q(\boldsymbol{x})}{\partial x_1} & \frac{\partial g_q(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial g_q(\boldsymbol{x})}{\partial x_n} \\ \frac{\partial h_{q+1}(\boldsymbol{x})}{\partial x_1} & \frac{\partial h_{q+1}(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial h_{q+1}(\boldsymbol{x})}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial h_m(\boldsymbol{x})}{\partial x_1} & \frac{\partial h_m(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial h_m(\boldsymbol{x})}{\partial x_n} \end{pmatrix} \tag{18}$$

When some constraints are not differentiable, the gradient matrix $\nabla C(\boldsymbol{x})$ can be obtained numerically by calculating

$C(\boldsymbol{x})$ repeatedly with changing the value of each decision variable x_i to $x_i+\eta$:

$$\nabla C(\boldsymbol{x}) = \frac{1}{\eta}\left(C(\boldsymbol{x}+\eta\,\boldsymbol{e}_1) - C(\boldsymbol{x}), \cdots, C(\boldsymbol{x}+\eta\,\boldsymbol{e}_n) - C(\boldsymbol{x})\right) \tag{19}$$

where $\boldsymbol{e}_i$ is a unit vector of which i-th element is 1 and other elements are 0 and η is a very small amount. Although the $\nabla C(\boldsymbol{x})$ is not invertible in general, the Moore-Penrose inverse or pseudoinverse $\nabla C(\boldsymbol{x})^+$ [34], which gives an approximate or best (least squares) solution to a system of linear equations, can be used instead in Eq. (17).

After $\Delta\boldsymbol{x}$ is obtained, a mutated vector can be obtained as follows:

$$\boldsymbol{x}^{\text{new}} = \boldsymbol{x} + \Delta\boldsymbol{x} \tag{20}$$

This mutation or repair operation is executed with gradient-based mutation probability P_g. In [33], only non-zero elements of $\Delta C(\boldsymbol{x})$ are repaired and the repair operation is repeated with some probability while amount of repair is not small. In the εDE, however, non-zero inequality constraints and all equality constraints are considered to keep the feasibility of equality constraints. The mutation operation is repeated fixed times R_g while the point is not feasible.

When the number of violated constraints is one, it is very difficult to decide the proper value of $\Delta\boldsymbol{x}$ because there are too many values that can satisfy Eq. (16). In this case, skip-move, where the gradient-based mutation is not applied with probability 0.5, is introduced.

V. Solving nonlinear optimization problems

In this paper, eighteen benchmark problems for the CEC2010 competition and special session on "Single Objective Constrained Real-Parameter Optimization" [29] are solved.

A. Test problems and the experimental conditions

In 18 problems from C01 to C18, 6 problems C06, C08, C09, C11, C14 and C18 have inequality constraints only and other 12 problems have equality constraints. All problems are solved in two cases of the number of decision variables (n) being 10 (10D) and 30 (30D).

In problems with equality constraints, the equality constraints are relaxed and converted to inequality constraints according to the equation, which is adopted in many methods: $|h_j(\boldsymbol{x})| \le \delta,\ \delta > 0$, where $\delta = 10^{-4}$.

B. Experimental Conditions

Independent 25 runs are performed for 18 problems. The maximum number of function evaluations is 200,000 for 10D and 600,000 for 30D in each run. In the ε constrained method including the εDEag, the objective function and the constraint violation are treated separately, and the evaluation of the objective function can often be omitted when the ε-level comparison can be decided only by constraint violation. Also, objective values are not used in the gradient-based mutation. However, in the experiments the number of evaluations for constraint violation is included in the number of function evaluations. If the algorithm is continued until the number of evaluations for the objective function reaches the specified maximum number of function evaluations, the results will become better. Also, the gradient matrix $\nabla C(\boldsymbol{x})$ is calculated using Eq. (19) in order to satisfy the condition of the competition although the constraints are differentiable. Thus, obtaining the gradient matrix costs $n+1$ function evaluations.

Experimental conditions are shown in the specified format as follows:

PC Configure:

System: Cygwin on Windows7 64bit	CPU: Intel Core2 Duo P8800 (2.67GHz)
RAM: 4GB	Language: C (gcc 4.3.4)
Algorithm: εDEag	

Parameters Setting:

a) All parameters to be adjusted.

Parameters for DE are population size (N), a scaling factor (F_0) and a crossover rate (CR_0). Parameters for the ε constrained method are control generations (T_c) and a parameter for initial ε level (θ). The others are archive size (M), gradient-based mutation rate (P_g) and the number of repeating the mutation (R_g).

b) Corresponding dynamic ranges.

We recommend the following ranges: $N \in [2n, 20n]$, $F_0 \in [0.6, 0.8]$, $CR_0 \in [0.6, 0.95]$, $T_c \in [500, 2000]$, $\theta \in [0.1, 2]$, $M \in [20n, 200n]$, $P_g \in [0.05, 0.5]$, and $R_g \in [1, 5]$.

c) Guidelines on how to adjust the parameters.

Higher N, F_0, CR_0, T_c, θ, M, P_g and R_g make search process more robust, but less fast.

d) Estimated cost of parameter tuning in terms of number of FEs.

In this paper, we studied a few values of $F_0 \in \{0.7, 0.5\}$ and $\theta \in \{0.2, 0.9\}$. However, the cost of parameter tuning is not so high.

e) Actual parameter values used.

$N = 4n$, $F_0 = 0.5$, $CR_0 = 0.9$, $M = 100n$, $T_c = 1000$, θ=0.9, $P_g = 0.2$ and $R_g = 3$.

C. Experimental results

Tables I-III and Tables IV-VI summarize the experimental results on problems of 10D and 30D, respectively. In the tables, the best, median, worst, average function values and standard deviation are shown when FES is 2×10^4, 1×10^5 and 2×10^5 in 10D cases or 6×10^4, 3×10^5 and 6×10^5 in 30D cases. c is the number of violated constraints at the median solution: the sequence of three numbers indicate the number of violations by more than 1.0, more than 0.01 and more than 0.0001 respectively. $\bar{v}$ is the mean value of the violations of all constraints at the median solution. The numbers in the parenthesis after the fitness value of the best, median, worst solution are the number of constraints which cannot satisfy feasible condition at the best, median and worst solutions respectively. Also, the rate of feasible runs, where the best individual is feasible, over 25 runs, the rate of actual number of evaluations for objective values and the rate of number of

TABLE I

FUNCTION VALUES ACHIEVED WHEN FES = 2×10^4, FES = 1×10^5, FES = 2×10^5 FOR 10D PROBLEMS C01-C06.

FEs		C01	C02	C03	C04	C05	C06
	Best	-7.470547e-01(0)	-2.079862e+00(0)	2.504222e+01(1)	4.322817e+01(4)	-4.786748e+02(0)	-5.114200e+02(1)
	Median	-7.465616e-01(0)	-1.948129e+00(0)	5.496418e+01(1)	2.022213e+01(4)	-4.669074e+02(0)	-5.055504e+02(2)
	Worst	-7.380606e-01(0)	-1.762696e+00(2)	5.884982e+01(1)	3.822121e+01(4)	-3.735183e+02(2)	-5.110654e+02(2)
2.0e+04	c	0, 0, 0	0, 0, 0	0, 1, 0	4, 0, 0	0, 0, 0	0, 0, 1
	$\bar{v}$	0.000000e+00	0.000000e+00	1.823799e-01	2.485030e+03	0.000000e+00	1.097707e-03
	Mean	-7.461990e-01	-1.827413e+00	8.859793e+01	2.417511e+01	-4.492354e+02	-5.040519e+02
	std	1.710721e-03	4.086036e-01	6.543636e+01	1.465286e+01	4.601499e+01	7.979349e+01
	Best	-7.473104e-01(0)	-2.277535e+00(0)	4.426954e-16(0)	3.189818e-03(0)	-4.836106e+02(0)	-5.786570e+02(0)
	Median	-7.473103e-01(0)	-2.266686e+00(0)	7.764341e-15(0)	4.574955e-03(0)	-4.836105e+02(0)	-5.786520e+02(0)
	Worst	-7.405572e-01(0)	-2.174499e+00(0)	1.684037e-13(0)	7.528466e-03(0)	-4.836104e+02(0)	-5.786397e+02(0)
1.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-7.470402e-01	-2.258064e+00	3.291394e-14	4.644191e-03	-4.836105e+02	-5.786514e+02
	std	1.323345e-03	2.372208e-02	4.973518e-14	9.666501e-04	3.901442e-05	4.410354e-03
	Best	-7.473104e-01(0)	-2.277702e+00(0)	0.000000e+00(0)	-9.992345e-06(0)	-4.836106e+02(0)	-5.786581e+02(0)
	Median	-7.473104e-01(0)	-2.269502e+00(0)	0.000000e+00(0)	-9.977276e-06(0)	-4.836106e+02(0)	-5.786533e+02(0)
	Worst	-7.405572e-01(0)	-2.174499e+00(0)	0.000000e+00(0)	-9.282295e-06(0)	-4.836106e+02(0)	-5.786448e+02(0)
2.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-7.470402e-01	-2.258870e+00	0.000000e+00	-9.918452e-06	-4.836106e+02	-5.786528e+02
	std	1.323339e-03	2.389779e-02	0.000000e+00	1.546730e-07	3.890350e-13	3.627169e-03
Feasible Rate(%)		100.0	100.0	100.0	100.0	100.0	100.0
Eval/Grad Rate(%)		50.43/2.52	13.79/16.45	97.66/2.08	36.08/17.22	59.37/10.46	22.33/20.86

TABLE II

FUNCTION VALUES ACHIEVED WHEN FES = 2×10^4, FES = 1×10^5, FES = 2×10^5 FOR 10D PROBLEMS C07-C12.

FEs		C07	C08	C09	C10	C11	C12
	Best	4.803959e+00(0)	1.164528e+01(0)	1.059736e+02(0)	5.712509e+01(0)	3.086598e+00(1)	-7.905214e+02(2)
	Median	6.573802e+00(0)	3.365420e+01(0)	5.146480e+02(0)	1.921460e+02(0)	-5.821922e+00(1)	-9.857587e+02(2)
	Worst	1.350772e+01(0)	1.422115e+02(0)	3.950100e+03(0)	1.978797e+04(0)	-2.980720e+00(1)	-1.094713e+03(2)
2.0e+04	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	2, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.837402e+07	8.929373e+08
	Mean	6.973119e+00	3.939858e+01	8.322056e+02	1.473156e+03	6.274059e-02	-7.769031e+02
	std	1.692819e+00	2.789563e+01	8.726522e+02	3.996628e+03	5.402863e+00	3.515025e+02
	Best	4.380900e-18(0)	1.468910e-18(0)	6.171378e-18(0)	2.721106e-15(0)	6.986599e-03(1)	-8.807972e+02(2)
	Median	8.660471e-17(0)	1.094156e+01(0)	3.427626e-16(0)	1.937221e-14(0)	-1.082266e-01(1)	-8.870475e+02(2)
	Worst	7.158432e-16(0)	1.537535e+01(0)	4.842253e-14(0)	2.251440e-12(0)	-1.728858e-01(1)	-8.349845e+02(2)
1.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	2, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.679925e+00	1.222527e+02
	Mean	1.322563e-16	6.727555e+00	5.625742e-15	1.675481e-13	-1.117674e-01	-8.796310e+02
	std	1.561722e-16	5.560668e+00	1.069657e-14	4.847480e-13	5.659352e-02	2.032367e+01
	Best	0.000000e+00(0)	0.000000e+00(0)	0.000000e+00(0)	0.000000e+00(0)	-1.522713e-03(0)	-5.700899e+02(0)
	Median	0.000000e+00(0)	1.094154e+01(0)	0.000000e+00(0)	0.000000e+00(0)	-1.522713e-03(0)	-4.231332e+02(0)
	Worst	0.000000e+00(0)	1.537535e+01(0)	0.000000e+00(0)	0.000000e+00(0)	-1.522713e-03(0)	-1.989129e-01(0)
2.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	0.000000e+00	6.727528e+00	0.000000e+00	0.000000e+00	-1.522713e-03	-3.367349e+02
	std	0.000000e+00	5.560648e+00	0.000000e+00	0.000000e+00	6.341035e-11	1.782166e+02
Feasible Rate(%)		100.0	100.0	100.0	100.0	100.0	100.0
Eval/Grad Rate(%)		99.79/0.03	71.86/1.88	97.70/2.03	97.68/2.04	30.82/6.66	21.71/14.97

TABLE III

FUNCTION VALUES ACHIEVED WHEN FES = 2×10^4, FES = 1×10^5, FES = 2×10^5 FOR 10D PROBLEMS C13-C18.

FEs		C13	C14	C15	C16	C17	C18
	Best	-2.744446e+01(0)	2.929111e+01(0)	9.864584e+01(0)	9.508562e-01(0)	1.568122e-01(0)	3.554314e-02(0)
	Median	8.018183e-01(0)	1.838913e+02(0)	6.272825e+02(0)	9.149691e-01(2)	1.959052e+00(0)	5.720749e-01(0)
	Worst	-1.362981e+01(1)	7.083548e+02(0)	4.792146e+03(0)	4.728808e-01(3)	1.901360e+02(0)	1.204689e+02(0)
2.0e+04	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 2, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	7.026079e-03	0.000000e+00	0.000000e+00
	Mean	-6.354859e+00	2.208989e+02	9.351854e+02	1.059883e+00	1.955948e+01	1.148526e+01
	std	1.223757e+01	2.016908e+02	1.198279e+03	2.384153e-01	4.220278e+01	3.002662e+01
	Best	-6.842850e+01(0)	9.522514e-16(0)	6.145255e-15(0)	5.175322e-06(0)	3.661408e-05(0)	5.451715e-13(0)
	Median	-6.791585e+01(0)	1.936708e-14(0)	5.664446e-14(0)	7.534254e-01(0)	5.653326e-03(0)	1.656687e-12(0)
	Worst	-6.462275e+01(0)	5.312856e-13(0)	4.498220e+00(0)	1.046122e+00(0)	2.832923e+00(0)	6.131050e-12(0)
1.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-6.747553e+01	6.144868e-14	1.799288e-01	5.123838e-01	2.390668e-01	1.997418e-12
	std	1.090999e+00	1.270958e-13	8.814676e-01	4.250333e-01	5.927959e-01	1.253200e-12
	Best	-6.842937e+01(0)	0.000000e+00(0)	0.000000e+00(0)	0.000000e+00(0)	1.463180e-17(0)	3.731439e-20(0)
	Median	-6.842936e+01(0)	0.000000e+00(0)	0.000000e+00(0)	2.819841e-01(0)	5.653326e-03(0)	4.097909e-19(0)
	Worst	-6.842936e+01(0)	0.000000e+00(0)	4.497445e+00(0)	1.018265e+00(0)	7.301765e-01(0)	9.227027e-18(0)
2.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-6.842936e+01	0.000000e+00	1.798978e-01	3.702054e-01	1.249561e-01	9.678765e-19
	std	1.025960e-06	0.000000e+00	8.813156e-01	3.710479e-01	1.937197e-01	1.811234e-18
Feasible Rate(%)		100.0	100.0	100.0	100.0	100.0	100.0
Eval/Grad Rate(%)		43.86/6.25	98.90/0.94	96.20/2.25	32.09/21.64	35.19/16.84	91.02/4.64

TABLE IV

FUNCTION VALUES ACHIEVED WHEN FES = 6×10^4, FES = 3×10^5, FES = 6×10^5 FOR 30D PROBLEMS C01-C06.

FEs		C01	C02	C03	C04	C05	C06
	Best	-7.333921e-01(0)	-1.994110e+00(0)	7.032977e+05(1)	2.485433e+01(4)	-4.092602e+02(0)	-4.593746e+02(2)
	Median	-7.035866e-01(0)	-1.777555e+00(0)	1.163191e+07(1)	1.492430e+01(4)	-3.883296e+02(0)	-4.317375e+02(2)
	Worst	-6.734933e-01(0)	-1.225212e+00(0)	1.336649e+06(1)	1.109962e+01(4)	-2.803086e+02(0)	-4.367668e+02(2)
6.0e+04	c	0, 0, 0	0, 0, 0	1, 0, 0	2, 2, 0	0, 0, 0	0, 0, 2
	$\bar{v}$	0.000000e+00	0.000000e+00	2.929462e+02	6.886533e+04	0.000000e+00	3.627696e-03
	Mean	-7.049626e-01	-1.746020e+00	5.672480e+07	1.897917e+01	-3.798507e+02	-3.024492e+02
	std	1.545449e-02	1.874331e-01	1.391002e+08	1.173982e+01	2.668034e+01	1.995699e+02
	Best	-8.197135e-01(0)	-2.167921e+00(0)	2.874428e+01(0)	1.883304e-01(3)	-4.527445e+02(0)	-5.284646e+02(0)
	Median	-8.178069e-01(0)	-2.151026e+00(0)	2.886133e+01(1)	4.597759e-01(3)	-4.486845e+02(0)	-5.276549e+02(0)
	Worst	-8.136913e-01(0)	-2.117096e+00(0)	4.418723e+01(1)	5.645744e-01(3)	-4.381987e+02(0)	-5.263893e+02(0)
3.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 2, 1	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	2.059402e-05	3.024051e-02	0.000000e+00	0.000000e+00
	Mean	-8.173765e-01	-2.148776e+00	3.009609e+01	3.620548e-01	-4.483747e+02	-5.276050e+02
	std	1.699387e-03	1.243480e-02	3.037083e+00	1.719255e-01	3.390638e+00	6.036711e-01
	Best	-8.218255e-01(0)	-2.169248e+00(0)	2.867347e+01(0)	4.698111e-03(0)	-4.531307e+02(0)	-5.285750e+02(0)
	Median	-8.206172e-01(0)	-2.152145e+00(0)	2.867347e+01(0)	6.947614e-03(0)	-4.500404e+02(0)	-5.280407e+02(0)
	Worst	-8.195466e-01(0)	-2.117096e+00(0)	3.278014e+01(0)	1.777889e-02(0)	-4.421590e+02(0)	-5.264539e+02(0)
6.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-8.208687e-01	-2.151424e+00	2.883785e+01	8.162973e-03	-4.495460e+02	-5.279068e+02
	std	7.103893e-04	1.197582e-02	8.047159e-01	3.067785e-03	2.899105e+00	4.748378e-01
Feasible Rate(%)		100.0	100.0	100.0	100.0	100.0	100.0
Eval/Grad Rate(%)		53.70/2.26	14.17/16.51	51.40/5.95	20.75/21.91	8.68/22.37	14.25/22.66

TABLE V

FUNCTION VALUES ACHIEVED WHEN FES = 6×10^4, FES = 3×10^5, FES = 6×10^5 FOR 30D PROBLEMS C07-C12.

FEs		C07	C08	C09	C10	C11	C12
	Best	2.078496e+03(0)	5.258766e+04(0)	1.513880e+06(0)	1.172075e+06(0)	-1.816687e+00(1)	7.010353e+02(1)
	Median	4.102869e+03(0)	1.307418e+05(0)	6.289817e+06(0)	3.762993e+06(0)	1.127853e+00(1)	-7.596637e+02(2)
	Worst	6.296039e+03(0)	2.438215e+05(0)	4.845045e+07(0)	1.067332e+07(0)	-2.300643e+00(1)	9.836261e+02(2)
6.0e+04	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	2, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.219559e+08	4.173898e+09
	Mean	3.886217e+03	1.352206e+05	8.140545e+06	4.639142e+06	2.047444e-01	-3.743380e+02
	std	1.076481e+03	4.529672e+04	9.470601e+06	2.638561e+06	1.800180e+00	7.377465e+02
	Best	1.706180e+00(0)	5.101791e+00(0)	1.247163e-02(0)	3.252002e+01(0)	-2.494509e-02(1)	-8.957723e+02(1)
	Median	2.277172e+00(0)	6.140925e+00(0)	6.293352e-02(0)	3.328903e+01(0)	2.825208e-04(1)	-4.907037e+02(1)
	Worst	3.253668e+00(0)	7.458995e+00(0)	1.052759e+02(0)	3.463263e+01(0)	1.602419e-02(1)	6.293700e+01(1)
3.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 0, 0	1, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.713867e+01	1.852738e+00
	Mean	2.367238e+00	6.160371e+00	1.076762e+01	3.326176e+01	1.311506e-03	-4.618196e+02
	std	4.215514e-01	5.676969e-01	2.820171e+01	4.545818e-01	1.542525e-02	4.301514e+02
	Best	1.147112e-15(0)	2.518693e-14(0)	2.770665e-16(0)	3.252002e+01(0)	-3.268462e-04(0)	-1.991453e-01(0)
	Median	2.114429e-15(0)	6.511508e-14(0)	1.124608e-08(0)	3.328903e+01(0)	-2.843296e-04(0)	5.337125e+02(1)
	Worst	5.481915e-15(0)	2.578112e-13(0)	1.052759e+02(0)	3.463243e+01(0)	-2.236338e-04(0)	5.461723e+02(1)
6.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 1, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.240709e-01
	Mean	2.603632e-15	7.831464e-14	1.072140e+01	3.326175e+01	-2.863882e-04	3.562330e+02
	std	1.233430e-15	4.855177e-14	2.821923e+01	4.545577e-01	2.707605e-05	2.889253e+02
Feasible Rate(%)		100.0	100.0	100.0	100.0	100.0	12.0
Eval/Grad Rate(%)		99.68/0.08	93.40/0.70	50.86/5.02	28.53/5.71	12.33/7.95	13.16/11.11

TABLE VI

FUNCTION VALUES ACHIEVED WHEN FES = 6×10^4, FES = 3×10^5, FES = 6×10^5 FOR 30D PROBLEMS C13-C18.

FEs		C13	C14	C15	C16	C17	C18
	Best	-6.551573e+00(0)	5.221324e+06(0)	3.006197e+07(0)	2.430586e-02(0)	1.874116e+01(0)	7.148353e+01(0)
	Median	1.040293e+01(1)	1.733364e+07(0)	7.774775e+07(0)	9.567051e-02(0)	5.480293e+01(0)	3.936284e+02(0)
	Worst	2.641541e+00(1)	2.832984e+07(0)	1.580314e+08(0)	1.668657e+00(1)	3.815730e+02(0)	1.417606e+03(0)
6.0e+04	c	0, 1, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	2.482959e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-2.077929e+00	1.748971e+07	8.242089e+07	2.251338e-01	8.682126e+01	5.342980e+02
	std	8.397922e+00	6.147696e+06	3.363218e+07	3.360001e-01	7.534782e+01	4.302516e+02
	Best	-5.929556e+01(0)	6.236798e+00(0)	2.161664e+01(0)	2.710505e-19(0)	2.165719e-01(0)	1.226054e+00(0)
	Median	-5.831213e+01(0)	7.115322e+00(0)	2.161895e+01(0)	7.589415e-19(0)	6.307720e+00(0)	2.679497e+01(0)
	Worst	-5.634109e+01(0)	1.041738e+01(0)	2.162287e+01(0)	1.345495e-15(0)	1.889064e+01(0)	7.375363e+02(0)
3.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-5.813198e+01	7.327857e+00	2.161927e+01	1.176815e-16	6.612098e+00	8.754569e+01
	std	8.748056e-01	8.835483e-01	1.733093e-03	3.287739e-16	5.059539e+00	1.664753e+02
	Best	-6.642473e+01(0)	5.015863e-14(0)	2.160345e+01(0)	0.000000e+00(0)	2.165719e-01(0)	1.226054e+00(0)
	Median	-6.531507e+01(0)	1.359306e-13(0)	2.160375e+01(0)	0.000000e+00(0)	5.315949e+00(0)	2.679497e+01(0)
	Worst	-6.429690e+01(0)	2.923513e-12(0)	2.160403e+01(0)	5.421011e-20(0)	1.889064e+01(0)	7.375363e+02(0)
6.0e+05	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	Mean	-6.535310e+01	3.089407e-13	2.160376e+01	2.168404e-21	6.326487e+00	8.754569e+01
	std	5.733005e-01	5.608409e-13	1.104834e-04	1.062297e-20	4.986691e+00	1.664753e+02
Feasible Rate(%)		100.0	100.0	100.0	100.0	100.0	100.0
Eval/Grad Rate(%)		30.94/7.42	98.68/1.15	65.61/4.63	64.44/7.22	35.55/16.92	11.23/14.35

evaluations for gradient matrix over the maximum number of function evaluations are shown in the table.

It is difficult to evaluate the results, because the known best values are not given. The results can be outlined as follows: In problems C03, C05, C07, C09, C10, C11, C13, C14 of 10D and C16 of 30D, the εDEag can find feasible solutions with similar values consistently in all 25 runs. In all problems except for C12 of 30D, the εDEag can find feasible solutions in all 25 runs. In problems C12 of 30D, feasible solutions are found on only 3 runs.

Algorithm Complexity:
Table VII shows the algorithm complexity of the εDEag.

TABLE VII
ALGORITHM COMPLEXITY OF THE εDEAG

n	T_1	T_2	$(T_2 - T_1)/T_1$
10	0.04766	0.05080	0.06588
30	0.16205	0.16676	0.02906

where T_1 is the averaged computing time of 10000 FEs for all problems, T_2 is the averaged complete computing time for the εDEag with 10000 FEs for all problems.

Convergence Map:
Figs 3 to 6 show the convergence graphs for the 10D and 30D problems of C09, C10, C14, C15, C17 and C18, where only feasible solutions of the best run out of the 25 runs are shown. The graphs show the logarithmic plots of function values over the number of function evaluations.

VI. CONCLUSIONS

Differential evolution is known as a simple, efficient and robust search algorithm that can solve unconstrained optimization problems. In this study, we proposed a simple idea of generating children and utilizing an archive in order to improve the stability and also efficiency of the εDEg. We proposed a new scheme of setting the control parameter of the ε level automatically. By combining these ideas, we proposed the εDEag. It is thought that the εDEag is a stable optimization algorithm from the results of εDEag on eighteen benchmark problems for the CEC2010 competition and special session on "Single Objective Constrained Real-Parameter Optimization".

Acknowledgments

This research is supported in part by Grant-in-Aid for Scientific Research (C) (No. 20500138,22510166) of Japan society for the promotion of science and Hiroshima City University Grant for Special Academic Research (General Studies) 7111.

REFERENCES

[1] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods," in *Proc. of the 4th Annual Conference on Evolutionary Programming*. Cambridge, Massachusetts: The MIT Press, 1995, pp. 135–155.

[2] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, Jan. 2002.

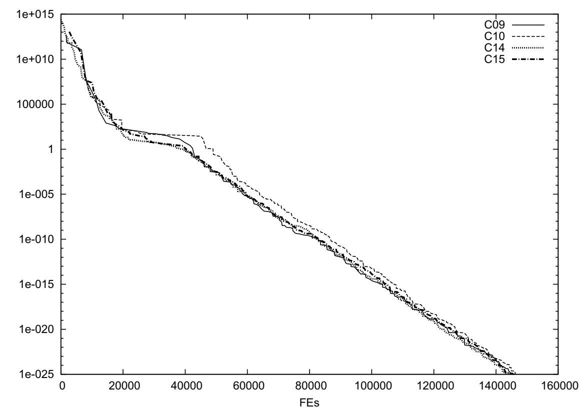

Fig. 3. Convergence plot for 10D problems C09, C10, C14 and C15

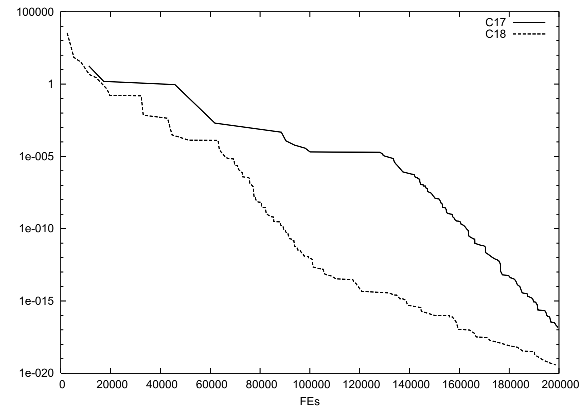

Fig. 4. Convergence plot for 10D problems C17 and C18

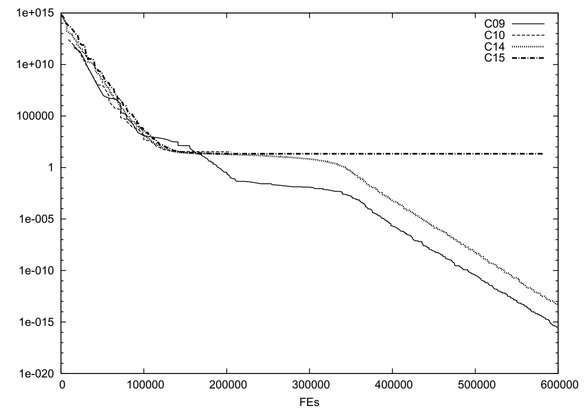

Fig. 5. Convergence plot for 30D problems C09, C10, C14 and C15

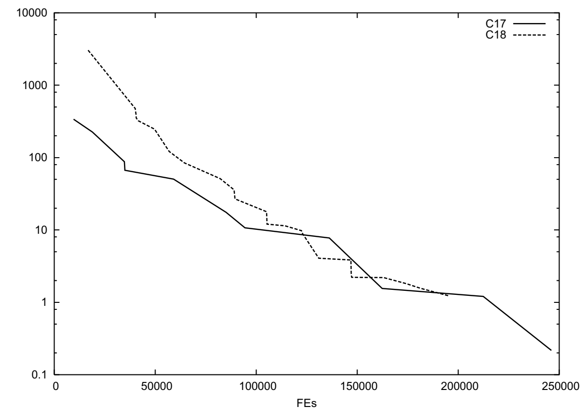

Fig. 6. Convergence plot for 30D problems C17 and C18

[3] T. Takahama and S. Sakai, "Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations," *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 5, pp. 437–451, Oct. 2005.

[4] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with dynamic ε-level control," in *Advances in Differential Evolution*, U. Chakraborty, Ed. Springer-Verlag, 2008, pp. 139–154.

[5] T. Takahama and S. Sakai, "Fast and stable constrained optimization by the ε constrained differential evolution," *Pacific Journal of Optimization*, vol. 5, no. 2, pp. 261–282, May 2009.

[6] T. Takahama and S. Sakai, "Efficient constrained optimization by the ε constrained adaptive differential evolution," in *Proc. of the 2010 IEEE Congress on Evolutionary Computation*, July 2010, to appear.

[7] T. Takahama, S. Sakai, and N. Iwane, "Solving nonlinear constrained optimization problems by the ε constrained differential evolution," in *Proc. of the 2006 IEEE Conference on Systems, Man, and Cybernetics*, Oct. 2006, pp. 2322–2327.

[8] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. of the 2006 IEEE Congress on Evolutionary Computation*, July 2006, pp. 308–315.

[9] T. Takahama and S. Sakai, "Solving difficult constrained optimization problems by the ε constrained differential evolution with gradient-based mutation," in *Constraint-Handling in Evolutionary Optimization*, E. Mezura-Montes, Ed. Springer-Verlag, 2009, pp. 51–72.

[10] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[11] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs," in *Proc. of the first IEEE Conference on Evolutionary Computation*, D. Fogel, Ed. Orlando, Florida: IEEE Press, 1994, pp. 579–584.

[12] Z. Michalewicz and N. Attia, "Evolutionary optimization of constrained problems," in *Proc. of the 3rd Annual Conference on Evolutionary Programming*, A. Sebald and L. Fogel, Eds. River Edge, NJ: World Scientific Publishing, 1994, pp. 98–108.

[13] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[14] B. Tessema and G. Yen, "A self adaptive penalty function based algorithm for constrained optimization," in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. C. Coello, and T. P. Runarsson, Eds. Vancouver, BC, Canada: IEEE Press, 16-21 July 2006, pp. 246–253.

[15] Y. Wang, Z. Cai, Y. Xhau, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary computation," *IEEE Trans. on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, 2008.

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

[17] T. Takahama and S. Sakai, "Tuning fuzzy control rules by the α constrained method which solves constrained nonlinear optimization problems," *Electronics and Communications in Japan, Part3: Fundamental Electronic Science*, vol. 83, no. 9, pp. 1–12, Sept. 2000.

[18] T. Takahama and S. Sakai, "Constrained optimization by ε constrained particle swarm optimizer with ε-level control," in *Proc. of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05)*, May 2005, pp. 1019–1029. [Online]. Available: http://www.ints.info.hiroshima-cu.ac.jp/˜takahama/eng/papers/ePSO05c.pdf

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

[20] E. Mezura-Montes and C. A. C. Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, Feb. 2005.

[21] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 4, pp. 424–435, Aug. 2005.

[22] E. Camponogara and S. N. Talukdar, "A genetic algorithm for constrained and multiobjective optimization," in *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, J. T. Alander, Ed. Vaasa, Finland: University of Vaasa, Aug. 1997, pp. 49–62.

[23] P. D. Surry and N. J. Radcliffe, "The COMOGA method: Constrained optimisation by multiobjective genetic algorithms," *Control and Cybernetics*, vol. 26, no. 3, pp. 391–412, 1997.

[24] C. A. C. Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil Engineering and Environmental Systems*, vol. 17, pp. 319–346, 2000.

[25] T. Ray, K. M. Liew, and P. Saini, "An intelligent information sharing strategy within a swarm for unconstrained and constrained optimization problems," *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, vol. 6, no. 1, pp. 38–44, Feb. 2002.

[26] T. P. Runarsson and X. Yao, "Evolutionary search and constraint violations," in *Proc. of the 2003 Congress on Evolutionary Computation*, vol. 2. Piscataway, New Jersey: IEEE Service Center, Dec. 2003, pp. 1414–1419.

[27] A. H. Aguirre, S. B. Rionda, C. A. C. Coello, G. L. Lizárraga, and E. M. Montes, "Handling constraints using multiobjective optimization concepts," *International Journal for Numerical Methods in Engineering*, vol. 59, no. 15, pp. 1989–2017, Apr. 2004.

[28] Y. Wang, Z. Cai, G. Cuo, and Z. Zhou, "Multiobjective optimization and hybrid evolutionary algorthm to solve constrained optimization problems," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 37, no. 3, pp. 560–575, 2007.

[29] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, 2010, to appear.

[30] T. Takahama, S. Sakai, and N. Iwane, "Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm," in *Proc. of the 18th Australian Joint Conference on Artificial Intelligence*, Dec. 2005, pp. 389–400, Lecture Notes in Computer Science 3809.

[31] R. Storn and K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[32] Z. Jingqiao and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[33] P. Chootinan and A. Chen, "Constraint handling in genetic algorithms using gradient-based repair method," *Computers & Operations Research*, vol. 33, no. 8, pp. 2263–2281, Aug. 2006.

[34] S.L.Campbell and C. J. Meyer, *Generalized Inverses of Linear Transformations*. Dover Publications, 1979.