Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 5, No. 2, pp. 261–282, May 2009

FAST AND STABLE CONSTRAINED OPTIMIZATION BY THE ε CONSTRAINED DIFFERENTIAL EVOLUTION

Tetsuyuki Takahama and Setsuko Sakai

Abstract: While research on constrained optimization using evolutionary algorithms has been actively pursued, it has had to face the problems that the ability to solve multi-modal problems, which have many local solutions within a feasible region, is insufficient, that the ability to solve problems with equality constraints is inadequate, and that the stability and efficiency of searches is low. In this study, we propose a new constrained optimization algorithm εDE, defined by applying the ε constrained method to a differential evolution (DE). DE is a simple, fast and stable population based search algorithm that is robust to multi-modal problems. Also, a new and simple way of controlling relaxation of constraints is proposed for the ε constrained method to solve problems with equality constraints. The εDE realizes stable and efficient searches that can solve multi-modal problems and those with equality constraints. The advantage of the εDE is shown by applying it to thirteen constrained problems of various types and comparing the results to those obtained by other methods.

Key words: *evolutionary algorithms, constrained optimization, nonlinear optimization, ε constrained method, differential evolution*

Mathematics Subject Classification: *65K10, 68W25*

1 Introduction

An evolutionary algorithm (EA) is the term commonly used for algorithms based on principles of evolution, and includes genetic algorithms (GAs), evolution strategies (ESs), and evolutionary and genetic programming. EAs are direct search methods that use only the value of the objective function and do not require extensive knowledge of the search space such as functional derivatives; essentially solving unconstrained optimization problems. However optimization problems in the real world are often constrained optimization problems where objective functions are optimized under given constraints.

There are many studies on solving constrained optimization problems using EAs [3, 5, 18, 23]. However there have been some difficulties in these studies:

(1) The ability to solve multi-modal problems is insufficient. EAs for constrained optimization can locate a feasible region and find feasible solutions in uni-modal problems. But when multi-modal problems that have many local solutions in a feasible region are solved, even if the EAs can locate the feasible region, they are sometimes trapped to a local solution and cannot search for an optimal solution. Thus a method that is robust to multi-modal problems is required.

(2) The feasibility of solutions obtained by EAs in problems with equality constraints is inadequate. Many EAs for constrained optimization cannot directly solve problems with

equality constraints. To overcome such problems, the definition of the problems needs to be changed by converting equality constraints into relaxed inequality constraints. As a result, the feasibility of the obtained solutions is inadequate. A method that can dynamically relax equality constraints and find better solutions with minimum constraint violation is required.

(3) The stability and efficiency of searches is low. Even when solving the same problem, EAs sometimes find good solutions but sometimes find only very bad solutions. The initial search points are usually generated randomly and the search process includes many stochastic operations in EAs. Sometimes EAs cannot overcome the effect of randomness in the search process of some problems. Thus, the stability of the search becomes low. Also, many EAs need rank-based selection or replacement, stochastic selection and mutations based on Gaussian or Cauchy distributions that incur high computational costs. Thus, the efficiency of search also becomes low.

To overcome these problems, we propose a new constrained optimization algorithm εDE, defined by applying the ε constrained method [41] to a differential evolution (DE) [31,32]. DE is an evolutionary algorithm for unconstrained global optimization that incorporates both crossover and mutation into a simple operation realized by selecting a random parent and adding a scaled difference between two new random parents to the parent. By incorporating DE, problems (1) and (3) can be solved; DE is a simple, fast and stable search algorithm that is robust to multi-modal problems. The εDE is stable because it uses a simple and stable selection and replacement mechanism excluding stochastic operations. The εDE is also efficient because it uses a simple arithmetic operation and does not use any rank-based operations or mutations based on Gaussian and Cauchy distributions. The problem (2) is solved by providing a new and simple way of controlling the relaxation of equality constraints for the ε constrained method to directly solve problems with equality constraints.

The εDE realizes stable and efficient searches that can solve multi-modal problems and those with equality constraints. The advantage of the εDE is shown by applying it to thirteen constrained problems of various types and comparing the results to those obtained by other methods.

The rest of this paper is organized as follows: Section II describes previous works. Section III briefly describes the ε constrained method. Section IV describes DE and the εDE with a new way of controlling the relaxation of equality constraints. Section V presents experimental results of various benchmark problems. Comparisons with other methods are included in this section. Section VI has discussion on parameter settings. Finally, Section VII concludes with a brief summary of the paper and some remarks.

2 Previous Works

EAs for constrained optimization can be classified into several categories by the way the constraints are treated:

(1) Constraints are only used to see whether a search point is feasible or not [6, 21]. In this category, the search process begins with one or multiple feasible points and continues to search for new points within the feasible region. When a new search point is generated and the point is not feasible, the point is repaired or discarded.

(2) The constraint violation, which is the sum of the violation of all constraint functions, is combined with the objective function. Approaches in this category are called penalty function methods. An extended objective function is defined by adding the constraint violation to the objective function as a penalty. The optimization of the objective function and the constraint violation is realized by the optimization of the extended objective function.

(3) The constraint violation and the objective function are used separately. In this category, both the constraint violation and the objective function are optimized separately.

Takahama and Sakai proposed the α constrained method [33–35] and the ε constrained method [41]. These methods adopt a lexicographic order, in which the constraint violation precedes the objective function, with relaxation of the constraints. The methods can effectively optimize problems with equality constraints through the relaxation of the constraints. Deb [4] proposed a method using an extended objective function that realizes the lexicographic ordering. Runarsson and Yao [28] proposed a stochastic ranking method based on ES and using a stochastic lexicographic order that ignores constraint violations with some probability. Mezura-Montes and Coello [17] proposed a comparison mechanism that is equivalent to the lexicographic ordering based on ES. Venkatraman and Yen [43] proposed a two-step optimization method based on GA, which first optimizes constraint violation and then objective function. These methods have been successfully applied to various problems.

(4) The constraints and the objective function are optimized by multiobjective optimization methods. In this category, constrained optimization problems are solved as multiobjective optimization problems in which the objective and the constraint functions are objectives to be optimized [1, 2, 26, 29, 30].

In category (1), generating initial feasible points is difficult and computationally demanding when the feasible region is very small. Especially if the problem has equality constraints, it is almost impossible to find initial feasible points. In category (2), it is difficult to select an appropriate value for the penalty coefficient that adjusts the strength of the penalty. Several methods that dynamically control the penalty coefficient have been proposed [12,13,20]. However, ideal control of the coefficient is problem dependent [27] and it is difficult to determine a general control scheme. Farmani and Wright [7] proposed a self-adaptive fitness formulation, but the solutions found for some problems are not sufficient. In category (4), the difficulty is that solving multiobjective optimization problems is a more difficult and expensive task than solving single objective optimization problems.

In this paper, we investigate the ε constrained method in the promising category (3). In the ε constrained method, a *constraint violation* for the constraints is introduced to indicate how much a search point violates the constraints, and the ε *level comparison* is defined as an order relation that gives priority to the constraint violation over the value of the objective function. The α and the ε constrained methods are new types of transformation methods that convert an algorithm for unconstrained optimization into an algorithm for constrained optimization by replacing the ordinal comparisons with the α and the ε level comparisons in direct search methods. We call the methods *algorithm transformation methods*. The α constrained method was applied to Powell's method [25], the nonlinear simplex method by Nelder and Mead [24], a genetic algorithm [9] and particle swarm optimization (PSO) [14,15]; and the α constrained Powell's method [33–35], the α constrained simplex method [36,38,42], the αGA [37, 39] and the αPSO [40] were proposed. The ε constrained method was applied to PSO, and the ε constrained PSO [41] was proposed.

Of these methods, the α constrained simplex method with mutations [42] is the best as it can stably and efficiently search solutions. In the nonlinear simplex method, a simplex is spanned by multiple search points and the simplex shows the region including an optimal solution. The simplex is gradually reduced while searching for better points. When the simplex has sufficiently converged, an optimal solution with a high precision is obtained. The nonlinear simplex method is a simple and fast mathematical optimization algorithm and is very effective for uni-modal problems. However, the simplex is often trapped by local solutions in multi-modal problems and an optimal solution cannot be found. To avoid such situations, boundary mutation, which searches the boundary of a feasible region, is

introduced in the α constrained simplex method. But boundary mutation incurs a high computational cost and also reduces search efficiency when an optimal solution exists inside the feasible region because the mutation searches for solutions at the boundary. Contrarily, the εDE can stably solve multi-modal problems.

3 The ε Constrained Method

In this section, the ε constrained method [41] is described.

3.1 Constrained Optimization Problems

The general constrained optimization problem (P) with inequality, equality, upper bound and lower bound constraints is defined as follows:

$$\begin{array}{lll} \text{(P)} \ \text{minimize} & f(\boldsymbol{x}), & \\ \text{subject to} & g_j(\boldsymbol{x}) \leq 0,\ j = 1, \ldots, q, & \\ & h_j(\boldsymbol{x}) = 0,\ j = q+1, \ldots, m, & \\ & l_i \leq x_i \leq u_i,\ i = 1, \ldots, n, & \end{array} \tag{3.1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is an n dimensional vector of decision variables, $f(\boldsymbol{x})$ is an objective function, $g_j(\boldsymbol{x}) \leq 0$ are q inequality constraints and $h_j(\boldsymbol{x}) = 0$ are $m - q$ equality constraints. The functions f, g_j and h_j are linear or nonlinear real-valued functions. The values u_i and l_i are the upper and lower bounds of x_i, respectively. The upper and lower bounds define the *search space* $\mathcal{S}$. Inequality and equality constraints define the *feasible region* $\mathcal{F}$. Feasible solutions exist in $\mathcal{F} \subseteq \mathcal{S}$.

3.2 The Constraint Violation

We introduce the constraint violation $\phi(\boldsymbol{x})$ to indicate by how much a search point $\boldsymbol{x}$ violates the constraints. The constraint violation $\phi(\boldsymbol{x})$ is the following function:

$$\begin{cases} \phi(\boldsymbol{x}) = 0\ (\boldsymbol{x} \in \mathcal{F}) \\ \phi(\boldsymbol{x}) > 0\ (\boldsymbol{x} \notin \mathcal{F}) \end{cases} \tag{3.2}$$

Some types of constraint violations, which are adopted as a penalty in penalty function methods, can be defined as follows:

$$\phi(\boldsymbol{x}) = \max\{\max_j\{0, g_j(\boldsymbol{x})\}, \max_j |h_j(\boldsymbol{x})|\} \tag{3.3}$$

$$\phi(\boldsymbol{x}) = \sum_j max\{0, g_j(\boldsymbol{x})\}^p + \sum_j |h_j(\boldsymbol{x})|^p \tag{3.4}$$

where p is a positive number.

3.3 The ε Level Comparison

The ε *level comparison* is defined as an order relation on the set of $(f(\boldsymbol{x}), \phi(\boldsymbol{x}))$. 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 by 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 $f(\boldsymbol{x})$.

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

$$(f_1, \phi_1) <_\varepsilon (f_2, \phi_2) \quad \Leftrightarrow \quad \begin{cases} f_1 < f_2, \text{ if } \phi_1, \phi_2 \leq \varepsilon \\ f_1 < f_2, \text{ if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, \text{otherwise} \end{cases} \tag{3.5}$$

$$(f_1, \phi_1) \leq_\varepsilon (f_2, \phi_2) \quad \Leftrightarrow \quad \begin{cases} f_1 \leq f_2, \text{ if } \phi_1, \phi_2 \leq \varepsilon \\ f_1 \leq f_2, \text{ if } \phi_1 = \phi_2 \\ \phi_1 \leq \phi_2, \text{otherwise} \end{cases} \tag{3.6}$$

In the case of $\varepsilon = 0$, $<_0$ and $\leq_0$ are equivalent to the lexicographic order in which the constraint violation $\phi(\boldsymbol{x})$ precedes the function value $f(\boldsymbol{x})$. Also, in the case of $\varepsilon = \infty$, the ε level comparisons $<_\infty$ and $\leq_\infty$ are equivalent to the ordinal comparisons $<$ and $\leq$ between function values.

3.4 The Properties of the ε Constrained Method

An optimization problem solved by the ε constrained method, that is, a problem in which ordinary comparisons are replaced with ε level comparisons, $(\mathrm{P}_{\leq_\varepsilon})$, is defined as follows:

$$(\mathrm{P}_{\leq_\varepsilon}) \quad \text{minimize}_{\leq_\varepsilon} \quad f(\boldsymbol{x}), \tag{3.7}$$

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

$$\begin{array}{lll} (\mathrm{P}^\varepsilon) & \text{minimize} & f(\boldsymbol{x}), \\ & \text{subject to} & \phi(\boldsymbol{x}) \leq \varepsilon. \end{array} \tag{3.8}$$

The problem $(\mathrm{P}_{\leq_\varepsilon})$ means searching for the minimum point ordered by the order relation ε level comparison $\leq_\varepsilon$. The problem (P^ε) means searching for the minimum point ordered by the ordinary comparison $\leq$ under the relaxed condition $\phi(\boldsymbol{x}) \leq \varepsilon$.

For the three types of problems, (P^ε), $(\mathrm{P}_{\leq_\varepsilon})$ and (P), the following theorems are given.

Theorem 3.1. *If an optimal solution of* (P^0) *exists, any optimal solution of* $(\mathrm{P}_{\leq_\varepsilon})$ *is an optimal solution of* (P^ε).

Proof. Let $\boldsymbol{x}^*$ be an optimal solution of (P^0). As $\boldsymbol{x}^*$ satisfies the constraints of (P^0), $\phi(\boldsymbol{x}^*) = 0$. Then, letting $\hat{\boldsymbol{x}}$ be an optimal solution of problem $(\mathrm{P}_{\leq_\varepsilon})$, there does not exist a point $\boldsymbol{x}$ which is smaller than $\hat{\boldsymbol{x}}$ in the meaning of the ε level comparison. That is, $(f(\hat{\boldsymbol{x}}), \phi(\hat{\boldsymbol{x}})) \leq_\varepsilon (f(\boldsymbol{x}^*), \phi(\boldsymbol{x}^*))$. From the definition of $\leq_\varepsilon$ and $\phi(\boldsymbol{x}^*) \leq \varepsilon$, $\phi(\hat{\boldsymbol{x}}) \leq \varepsilon$. Therefore, $\hat{\boldsymbol{x}}$ is a feasible solution of problem (P^ε).

For any point $\boldsymbol{x} \in X^\varepsilon = \{\boldsymbol{x} | \phi(\boldsymbol{x}) \leq \varepsilon\}$, $(f(\hat{\boldsymbol{x}}), \phi(\hat{\boldsymbol{x}})) \leq_\varepsilon (f(\boldsymbol{x}), \phi(\boldsymbol{x}))$. $f(\hat{\boldsymbol{x}}) \leq f(\boldsymbol{x})$ from the definition of $\leq_\varepsilon$, $\phi(\hat{\boldsymbol{x}}) \leq \varepsilon$ and $\phi(\boldsymbol{x}) \leq \varepsilon$. Therefore, $\hat{\boldsymbol{x}}$ is an optimal solution of problem (P^ε).

Theorem 3.2. *If an optimal solution of* (P) *exists, any optimal solution of* $(\mathrm{P}_{\leq_0})$ *is an optimal solution of* (P).

Proof. Since problem (P) is identical with problem (P^0), letting $\varepsilon = 0$, this theorem is proved by Theorem3.1.

Theorem 3.3. *Let $\{\varepsilon_n\}$ be a strictly decreasing non-negative sequence converging to 0. Let $f(\boldsymbol{x})$ and $\phi(\boldsymbol{x})$ be continuous functions of $\boldsymbol{x}$. Assume that an optimal solution $\boldsymbol{x}^*$ of (P^0) exists and an optimal solution $\hat{\boldsymbol{x}}_n$ of $(\mathrm{P}_{\leq_{\varepsilon_n}})$ exists for any ε_n. Then, any accumulation point of the sequence $\{\hat{\boldsymbol{x}}_n\}$ is an optimal solution of (P^0).*

Proof. From Theorem3.1, $\hat{\boldsymbol{x}}_n$ is an optimal solution of problem $(\mathrm{P}^{\varepsilon_n})$ and $f(\hat{\boldsymbol{x}}_n) \leq f(\boldsymbol{x}^*)$. Let $\bar{\boldsymbol{x}}$ be an accumulation point of the sequence $\{\hat{\boldsymbol{x}}_n\}$ and let $\{\hat{\boldsymbol{x}}_{n_k}\}$ be a subsequence of $\{\hat{\boldsymbol{x}}_n\}$ converging to $\bar{\boldsymbol{x}}$. That is, $\lim_{k\to\infty} \hat{\boldsymbol{x}}_{n_k} = \bar{\boldsymbol{x}}$. Thus, $f(\boldsymbol{x}^*) \geq \lim_{k\to\infty} f(\hat{\boldsymbol{x}}_{n_k}) = f(\bar{\boldsymbol{x}})$. On the other hand, from the continuity of ϕ and $\lim_{k\to\infty} \varepsilon_{n_k} = 0$, $\phi(\bar{\boldsymbol{x}}) = \lim_{k\to\infty} \phi(\hat{\boldsymbol{x}}_{n_k}) \leq \lim_{k\to\infty} \varepsilon_{n_k} = 0$. Then, $\bar{\boldsymbol{x}}$ is a feasible solution of problem (P^0) and $f(\boldsymbol{x}^*) \leq f(\bar{\boldsymbol{x}})$. Therefore, $f(\bar{\boldsymbol{x}}) = f(\boldsymbol{x}^*)$. That is, $\bar{\boldsymbol{x}}$ is an optimal solution of problem (P^0). □

Theorems 3.1 and 3.2 show that a constrained optimization problem can be transformed into an equivalent unconstrained optimization problem using the ε level comparisons. So, if the ε level comparisons are incorporated into an existing unconstrained optimization method, constrained optimization problems can be solved. It is thought that the ε constrained method converts an algorithm for unconstrained optimization into an algorithm for constrained optimization by replacing ordinary comparisons with the ε level comparisons. Theorem 3.3 shows that an optimal solution of (P^0) is asymptotically obtained by converging the ε level to 0, if a perfect optimization algorithm, which can find optimal solution perfectly, is used. This is the same as the penalty method can solve constrained problems by increasing the penalty coefficient to ∞.

4 The εDE

In this section, we first describe differential evolution. Then, we describe the εDE, which is the integration of the ε constrained method and DE. A control function of the relaxation of equality constraints is also defined.

4.1 Differential Evolution

Differential evolution is a variant of ES proposed by Storn and Price [31, 32]. DE is a stochastic direct search method using population or multiple search points. DE has been successfully applied to 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.

The main feature of DE is that DE uses simple arithmetic operations to avoid the control of Gaussian mutation in ES. In general, the mutation process must be controlled to adjust the step size of the Gaussian mutation, because the ideal step size depends on the locus or the position of the gene that is mutated and also depends on the state of the evolution process. DE adopts the sum of a base vector and the scaled difference vectors as the mutation operation instead of Gaussian mutation. The base vector is an individual selected from the population. The difference vectors are formed by the differences between a pair of individuals randomly selected from the population. As the search space by the population contracts and expands over generations, the step size in each dimension, which is given by the difference vectors, adapts automatically.

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

DE/best/*num*/*cross* selects the best individual in the population. "*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, DE/*base*/*num*/bin shows that crossover is controlled by a binomial crossover using constant crossover rate. DE/*base*/*num*/exp shows that crossover is controlled by a binomial crossover using exponentially decreasing the crossover rate.

In DE, initial individuals are randomly generated within the search space and form an initial population. Each individual contains n genes as decision variables or a decision vector. At each generation or iteration, all individuals are selected as parents. Each parent is processed as follows: The mutation process begins by choosing $1+2$ *num* individuals from all individuals except for the parent in the processing. The first individual is a base vector. All subsequent individuals are paired to create *num* difference vectors. The difference vectors are scaled by the scaling factor F and added to the base vector. The resulting vector is then recombined or crossovered with the parent. The probability of recombination at an element is controlled by the crossover rate CR. This crossover process produces a trial vector. Finally, for survivor selection, the trial vector is accepted for the next generation if the trial vector is better than the parent.

4.2 The Algorithm of the εDE

The algorithm of the εDE based on DE/rand/1/exp variant, which is used in this study, is as follows:

Step0 Initialization. Initial N individuals $\boldsymbol{x}^i$ are generated as the initial search points, where there is an initial population $P(0) = \{\boldsymbol{x}^i, i = 1, 2, \cdots, N\}$. An initial ε level is given by the ε level control function $\varepsilon(0)$.

Step1 Termination condition. If the number of generations (iterations) exceeds the maximum generation $T_{\max}$, the algorithm is terminated.

Step2 Mutation. 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 $\boldsymbol{x}'$ is generated by the base vector $\boldsymbol{x}^{p1}$ and the difference vector $\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}$ as follows:

$$\boldsymbol{x}' = \boldsymbol{x}^{p1} + F(\boldsymbol{x}^{p2} - \boldsymbol{x}^{p3}) \tag{4.1}$$

where F is a scaling factor.

Step3 Crossover. The vector $\boldsymbol{x}'$ is crossovered with the parent $\boldsymbol{x}^i$. A crossover point j is chosen randomly from all dimensions $[1, n]$. The element at the j-th dimension of the trial vector $\boldsymbol{x}^{\text{new}}$ is inherited from the j-th element of the vector $\boldsymbol{x}'$. The elements of subsequent dimensions are inherited from $\boldsymbol{x}'$ with exponentially decreasing probability defined by a crossover rate CR. Otherwise, the elements are inherited from the parent $\boldsymbol{x}^i$. In real processing, Step2 and Step3 are integrated as one operation.

Step4 Survivor selection. The trial vector $\boldsymbol{x}^{\text{new}}$ is accepted for the next generation if the trial vector is better than the parent $\boldsymbol{x}^i$.

Step5 Controlling the ε level. The ε level is updated by the ε level control function $\varepsilon(t)$.

Step6 Go back to Step1.

Fig. 1 shows the algorithm of the εDE.

```
εDE/rand/1/exp()
{
  P=Generate N individuals {x^i} randomly;
  ε=ε(0);
  for(t=1; t ≤ T_max; t++) {
    for(i=1; i ≤ N; i++) {
      (p1, p2, p3)=select randomly from [1, N]\{i} s.t. p1 ≠ p2, p2 ≠ p3, p3 ≠ p1;
      x^new=x^i ∈ P;
      j=select randomly from [1, n];
      k=1;
      do {
        x_j^new=x_j^p1+F(x_j^p2 − x_j^p3);
        j=(j + 1)%n;
        k++;
      } while(k ≤ n && u(0, 1) < CR);
      if((f(x^new), φ(x^new)) <_ε (f(x^i), φ(x^i))) x^i=x^new;
    }
    ε=ε(t);
  }
}
```

Figure 1: The algorithm of the ε constrained differential evolution with control of the ε level, where $\varepsilon(t)$ is the ε level control function, F is a scaling factor, CR is a crossover rate, and $u(0,1)$ is a uniform random number generator in $[0,1]$.

4.3 Controlling the ε Level

Usually, the ε level does not need to be controlled. Many constrained problems can be solved based on the lexicographic order where the ε level is constantly 0. However, when a constrained problem has severe small feasible region, the searching process is often trapped at a local solution. From theorem 3.3, the ε constrained method can avoid this situation by enlarging feasible region using the ε level and converging the enlarged feasible region to the original feasible region. Although DE is not a perfect optimization algorithm, DE is a very robust algorithm. Thus, it is expected that εDE can solve constrained optimization problems with severe feasible region if the ε level is controlled properly.

In this study, a new and simple way of controlling the ε level is defined according to the equation (4.2). In the later, it is shown that εDE can solve constrained problems with equality constraint, which are the problems with the most severe feasible region, using the ε level control with the equation (4.2). 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 $T_{\rm c}$. After the number of iterations exceeds $T_{\rm c}$, the ε level is set to 0 to obtain solutions with minimum constraint violation.

$$\begin{aligned} \varepsilon(0) &= \phi(\boldsymbol{x}_\theta) \\ \varepsilon(t) &= \begin{cases} \varepsilon(0)(1-\frac{t}{T_c})^{cp}, & 0 < t < T_c, \\ 0, & t \geq T_c \end{cases} \end{aligned} \tag{4.2}$$

where $\boldsymbol{x}_\theta$ is the top θ-th individual and $\theta = 0.2N$. $T_{\rm c}$ is 80% of the maximum generations, or $T_{\rm c} = 0.8T_{\rm max}$.

5 Solving Constrained Nonlinear Programming Problems

In this paper, thirteen benchmark problems that are mentioned in some studies [17, 28, 42] are optimized, and the results by the εDE are compared with those results.

5.1 Test Problems and the Experimental Conditions

In the thirteen benchmark problems, `g02`, `g03`, `g08` and `g12` are maximization problems, while the others are minimization problems. Problems `g03`, `g05`, `g11` and `g13` contain equality constraints. Problem `g12` has disjointed feasible regions. The harder version studied in [16], where the feasible region consists of 9^3 disjointed spheres, each with a radius of 0.25, is solved. Table 1 shows the outline of the thirteen problems [7, 17]. The table contains the number of variables n, the form of the objective function, the number of linear inequality constraints (LI), nonlinear inequality constraints (NI), linear equality constraints (LE), nonlinear equality constraints (NE) and the number of constraints active at the optimal solution. The problems marked by an upward arrow are maximization problems.

Table 1: Summary of test problems

f	n	Form of f	LI	NI	LE	NE	active
g01	13	quadratic	9	0	0	0	6
↑g02	20	nonlinear	1	1	0	0	1
↑g03	10	polynomial	0	0	0	1	1
g04	5	quadratic	0	6	0	0	2
g05	4	cubic	2	0	0	3	3
g06	2	cubic	0	2	0	0	2
g07	10	quadratic	3	5	0	0	6
↑g08	2	nonlinear	0	2	0	0	0
g09	7	polynomial	0	4	0	0	2
g10	8	linear	3	3	0	0	6
g11	2	quadratic	0	0	0	1	1
↑g12	3	quadratic	0	9^3	0	0	0
g13	5	nonlinear	0	0	1	2	3

The parameters for the ε constrained method are as follows: Every constraint violation is defined as a simple sum of constraints, or $p = 1$ in the equation (3.4). The ε level is controlled using the equation (4.2) for problems with equality constraints using $cp = 5$ and for the other problems the ε level is fixed to 0. The parameters for DE are: The number of search points $N = 40$, the maximum generation $T_{\max} = 4,999$, the scaling factor $F = 0.7$ and the crossover rate $CR = 0.9$ are common settings. In `g12`, $T_{\max} = 499$ is used. Thus, the maximum number of function evaluations is 200,000 except `g12` (20,000 evaluations). In this paper, 30 independent runs are performed. The effect of parameters will be discussed later.

5.2 Experimental Results

Table 2 summarizes the experimental results. The table shows the known "optimal" solution for each problem and the statistics from the 30 independent runs. These include the best,

median, mean, and worst values and the standard deviation of the objective values found. Also, the average constraint violation of the best solution, the average number of evaluations of the objective function and the constraints to find the best solution in each run are shown in the columns labeled violation, #func and #const respectively. The average execution time (seconds) in each run using a 1.3GHz Mobile Pentium III notebook PC is shown in the column labeled time(s). The table also shows the results when the equality constraints are relaxed and converted to inequality constraints according to the equation (5.1), which is adopted in many methods:

$$|h_j(\boldsymbol{x})| \leq \delta,\ \delta > 0, \tag{5.1}$$

where $\delta = 10^{-4}$.

Table 2: Experimental results on 13 benchmark problems using standard settings; 30 independent runs were performed

f	optimal	best	median violation	mean #func	worst #const	st. dev. time(s)
g01	-15.000	-15.000000	-15.000000	-15.000000	-15.000000	0
			0	52859.0	169331.2	0.31
↑g02	0.803619	0.803618	0.803614	0.803613	0.803588	5.587e-06
			0	82559.0	198707.4	0.62
↑g03	1.000	0.999999	0.999994	0.999991	0.999953	1.009e-05
			5.534e-11	64953.5	178246.8	0.29
	($\delta = 10^{-4}$)	1.000500	1.000500	1.000500	1.000500	6.457e-09
			1.000e-04	89817.4	199667.1	0.28
g04	-30665.539	-30665.538670	-30665.538670	-30665.538670	-30665.538670	0
			0	35612.8	89555.8	0.22
g05	5126.498	5126.498110	5126.498113	5126.498119	5126.498164	1.289e-05
			5.301e-15	58119.5	182886.2	0.28
	($\delta = 10^{-4}$)	5126.496714	5126.496714	5126.496714	5126.496714	1.819e-12
			1.000e-04	65846.2	175659.5	0.27
g06	-6961.814	-6961.813876	-6961.813876	-6961.813876	-6961.813876	0
			0	6948.4	12727.7	0.14
g07	24.306	24.306209	24.306209	24.306209	24.306209	4.269e-09
			0	56809.3	199759.1	0.30
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825	0
			0	4234.4	4896.8	0.22
g09	680.630	680.630057	680.630057	680.630057	680.630057	0
			0	33177.7	72071.8	0.24
g10	7049.248	7049.248021	7049.248021	7049.248021	7049.248021	0
			0	33754.5	198651.9	0.25
g11	0.750	0.750000	0.750000	0.750000	0.750000	0
			9.035e-21	115694.2	171171.9	0.13
	($\delta = 10^{-4}$)	0.749900	0.749900	0.749900	0.749900	0
			1.000e-04	118115.0	161004.4	0.14
↑g12	1.000000	1.000000	1.000000	1.000000	1.000000	0
			0	4674.9	9278.9	0.29
g13	0.053950	0.053950	0.053950	0.053950	0.053950	8.453e-09
			5.864e-17	63913.8	196419.8	0.21
	($\delta = 10^{-4}$)	0.053942	0.053942	0.053942	0.053942	0
			1.000e-04	77860.8	188870.6	0.22

For every problem, the best, median, average and worst values are almost equivalent to the optimal solutions. For problems g01, g04, g06, g07, g08, g09, g10, g11, g12 and g13, the optimal solutions are found consistently in all 30 runs. For other three problems g02, g03 and g05, the optimal or near-optimal solutions are found in all 30 runs. These results show that the εDE is a very stable algorithm. The problem g02 is a multi-modal problem that has many local optima (maxima) with high peaks near the global optimum within the feasible region. Many other methods cannot constantly obtain high quality solutions, but the εDE in all 30 runs consistently found near-optimal solutions. Thus, it is thought that the εDE has a high ability to solve multi-modal problems.

For problems g03, g05, g11 and g13, which contain equality constraints, the constrained violation, given by the maximum of the constraint functions defined by the equation (3.3) is about from 10^{-20} to 10^{-10}, which is very small and indeed negligible. These results show that the εDE can directly solve problems with equality constraints without converting the equality constraints to relaxed inequality constraints and so can obtain very strict and feasible solutions. When equality constraints are relaxed using $\delta = 10^{-4}$, the εDE found better objective than optimal values because the constraint violation was a fairly large value 10^{-4}.

It is thought that the good performance of εDE is caused by good balance between the diversity and the intensity of the search process. In earlier generations, the diversity of the search is very large. The search points gradually converge to the global optimum or near optimum. In later generations, the intensity of the search becomes large and the diversity is almost lost.

The εDE is a very fast algorithm. The execution times ranged from 0.13 seconds to 0.62 seconds using a notebook PC. The execution times are less than 1/3 seconds in all problems, except for g02. The number of evaluations of the constraints to find the best solution ranged from about 5,000 to 200,000. The number of evaluations of the objective function ranged between about 4,000 and 118,000. These results show that the εDE is very efficient algorithm.

In the εDE, the objective function and the constraints are treated separately. So, when the order relation of the search points can be decided only by the constraint violation of the constraints, the objective function is not evaluated. Thus, the number of evaluations of the objective function is less than the number of evaluations of the constraints. This nature of the εDE contributes to the efficiency of the algorithm especially when the objective function is computationally demanding.

5.3 Comparison with Other Methods

There are some methods that solved the same thirteen problems. In the methods, for comparative studies we chose the stochastic ranking method (SR) proposed Runarsson and Yao [28], the simple multimembered evolution strategy (SMES) proposed by Mezura-Montes and Coello [17] and α constrained simplex method with mutations (αSimplex) as proposed by Takahama and Sakai [42], because the results of these methods are better than the results of the other methods, and they reported good quality statistical information.

In SR, 30 independent runs were performed, the maximum number of evaluations in each run was $1750 \times 200 = 350,000$ except for g12 (175×200=35,000) and all equality constraints were relaxed by the equation (5.1), where $\delta = 10^{-4}$. Thus, it is thought that the constraint violation in problems with equality constraints was about 10^{-4}. In SMES, 30 independent runs were performed, the maximum number of evaluations in each run was $800 \times 300 = 240,000$. For problems with equality constraints, allowable tolerance, which

is similar to the ε level, is controlled. They used different control schemes to solve the problems. In g03, g05 and g11, the allowable tolerance is controlled from 0.001 to 0.0004 and is controlled from 3.0 to 0.00003 in g13. Thus, it is thought that the constraint violation in g03, g05 and g11 was about 4×10^{-4} and in g13 it was about 3×10^{-5}. Contrarily, the εDE used the same way of controlling the ε level for all problems with equality constraints. In αSimplex, 30 independent runs were performed, the maximum number of evaluations in each run was about 30,000 for problem g12 and about from 290,000 to 330,000 for the other problems. All equality constraints were relaxed using $\delta = 10^{-4}$.

Table 3, 4, 5, 6 and 7 show comparisons of the best, median, average, worst values and standard deviation for the four methods. The better cases are highlighted using boldface. The results of the εDE were taken from Table 2, where equality constraints are relaxed using $\delta = 10^{-4}$, and the maximum number of evaluations was 20,000 for problem g12 and 200,000 for the other problems.

Table 3: Comparison of the best values among proposed (indicated by εDE), Runarsson and Yao's (indicated by SR), Mezura-Montes and Coello's (indicated by SMES) and Takahama and Sakai's (indicated by αSimplex) algorithms

f	optimal	εDE	SR	SMES	αSimplex
g01	-15.000	-15.000	-15.000	-15.000	-15.000
↑g02	0.803619	0.803618	0.803515	0.803601	**0.803619**
↑g03	1.000	**1.001**	1.000	1.000	**1.001**
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.498	**5126.497**	**5126.497**	5126.599	**5126.497**
g06	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
g07	24.306	**24.306**	24.307	24.327	**24.306**
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825
g09	630.63	**680.630**	**680.630**	680.632	**630.630**
g10	7049.25	**7049.248**	7054.316	7051.903	**7049.248**
g11	0.75	0.750	0.750	0.75	0.750
↑g12	1.000	1.000000	1.000000	1.000	1.000000
g13	0.053950	**0.053942**	0.053957	0.053986	**0.053942**

All methods found optimal solutions for all 30 runs for g01, g04, g06, g08, g11 and g12. The εDE found better best values than SR and SMES for g02, g03, g07, g10 and g13, and better best values than SMES for g05 and g09. The αSimplex found better best value for g02 but the difference was very small, and the εDE found the same best values as the αSimplex for the other problems. For median values, the εDE found better values than all other methods for g02, better values than SR and SMES for g03, g05, g07, g09, g10 and g13. For average values, the εDE found better values than all other methods for g02 and g13, better values than SR and SMES for g03, g05, g07, g09 and g10. For worst values, the εDE found better values than all other methods for g02, g07 and g13, better values than SR and SMES for g03, g05, g09 and g10. For standard deviations that shows the stability of methods, the εDE is better than all other methods for g02, g05, g06, g07, g09, g10, g11 and g13. Also, the εDE found solutions more stable than αSimplex for g01 and g12, than SR and SMES for g03, than SR for g04 and g08.

The εDE found better solutions, or at least the same solutions, than other methods for all problems except for the best value found by the αSimplex for g02. These results show that the performance of the εDE is better than the performance of the other methods because

Table 4: Comparison of the median values among εDE, SR, SMES and αSimplex algorithms

f	optimal	εDE	SR	SMES	αSimplex
g01	-15.000	-15.000	-15.000	-15.000	-15.000
↑g02	0.803619	**0.803614**	0.785800	0.792549	0.785163
↑g03	1.000	**1.001**	1.000	1.000	**1.001**
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.498	**5126.497**	5127.372	5160.198	**5126.497**
g06	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
g07	24.306	**24.306**	24.357	24.426	**24.306**
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825
g09	630.630	**680.630**	680.641	680.642	**680.630**
g10	7049.248	**7049.248**	7372.613	7253.603	**7049.248**
g11	0.75	0.750	0.750	0.75	0.750
↑g12	1.000	1.000000	1.000000	1.000	1.000000
g13	0.053950	**0.053942**	0.057006	0.061873	**0.053942**

Table 5: Comparison of the average values among εDE, SR, SMES and αSimplex algorithms

f	optimal	εDE	SR	SMES	αSimplex
g01	-15.000	-15.000	-15.000	-15.000	-15.000
↑g02	0.803619	**0.803613**	0.781975	0.785238	0.784187
↑g03	1.000	**1.001**	1.000	1.000	**1.001**
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.498	**5126.497**	5128.881	5174.492	**5126.497**
g06	-6961.814	-6961.814	-6875.940	-6961.284	-6961.814
g07	24.306	**24.306**	24.374	24.475	**24.306**
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825
g09	630.630	**680.630**	680.656	680.643	**680.630**
g10	7049.248	**7049.248**	7559.192	7253.047	**7049.248**
g11	0.75	0.750	0.750	0.75	0.750
↑g12	1.000	1.000000	1.000000	1.000	1.000000
g13	0.053950	**0.053942**	0.067543	0.166385	0.066770

the εDE found better solutions or at least the same solutions on average in all problems using much fewer numbers of function evaluations. Also, the stability of the εDE is clearly better than the other methods.

6 Discussion

In this section, the effect of algorithm parameters such as scaling factor and crossover rate is discussed.

6.1 Effect of Scaling Factor

The scaling factor F is effective for controlling the trade-off between a convergence speed and robustness of search. The recommended value is $F = 0.8$, while a lower F makes convergence speed slower and the search more robust, but it takes a long time to converge to an optimal

Table 6: Comparison of the worst values among εDE, SR, SMES and αSimplex algorithms

f	optimal	εDE	SR	SMES	αSimplex
g01	-15.000	-15.000	-15.000	-15.000	-15.000
↑g02	0.803619	**0.803588**	0.726288	0.751322	0.784187
↑g03	1.000	**1.001**	1.000	1.000	**1.001**
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.498	**5126.497**	5142.472	5304.167	**5126.497**
g06	-6961.814	-6961.814	-6350.262	-6952.482	-6961.814
g07	24.306	**24.306**	24.642	24.843	24.307
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825
g09	630.630	**680.630**	680.763	680.719	**680.630**
g10	7049.248	**7049.248**	8835.655	7638.366	**7049.248**
g11	0.75	0.750	0.750	0.75	0.750
↑g12	1.000	1.000000	1.000000	1.000	1.000000
g13	0.053950	**0.053942**	0.216915	0.468294	0.066770

solution. Therefore, a suitable scaling factor needs to be selected. In this study, a lower scaling factor $F = 0.7$ is used for more robust searching of multi-modal problems.

Table 8 summarizes the mean of the objective values in the cases of the scaling factor F alone being changed to 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0 from standard settings using $\delta = 0$. The better cases are highlighted using boldface. In the case of $F = 0.5$ the results for problems `g02`, `g05`, `g06` and `g13` are a little worse than the results when F is greater than 0.5. Also, in the case of $F = 1.0$ the results of problems `g03`, `g07`, `g09` and `g10` are worse than the other results. These results show that the εDE is very stable for the selection of a scaling factor, and that a scaling factor between 0.6 and 0.9 is an appropriate setting for many problems.

6.2 Effect of a Crossover Rate

DE is much more sensitive to the choice of a scaling factor than it is to the choice of a crossover rate CR. CR is a parameter for fine tuning. High values of CR give faster convergence if convergence occurs. Sometimes, however, a very low value is chosen to make DE robust enough for a particular problem. The recommended value is $CR = 0.9$. In this study, the recommended value is used.

Table 9 summarizes the mean of the objective values in the cases of the crossover rate CR alone being changed to 0.5, 0.6, 0.7, 0.8, 0.9 and 0.95 from standard settings using $\delta = 0$. In the cases of $CR = 0.5, 0.6$, the results for `g02`, `g03`, `g05`, `g07`, `g09` and `g10` are worse than the other results. Also, in the case of $CR = 0.7$, the results for `g02`, `g03`, `g05`, `g07` and `g10` are a little worse than the results when CR is higher than 0.7. These results show that the εDE is fairly stable for selection of the crossover rate, and that a crossover rate between 0.7 and 0.95 is an appropriate setting for many problems.

6.3 Effect of the Parameter for Controlling the ε Level

In the εDE, a feasible region can be expanded by relaxing the ε level. The expanded feasible region can be reduced to the original feasible region by decreasing the ε level to 0. The parameter cp in the equation (4.2) adjusts the speed of decreasing the ε level and the speed of reducing the expanded feasible region. If cp is properly selected, the ε level approaches to 0 gradually and the risk that the search points converge to a local optimum becomes low.

Table 7: Comparison of the standard deviations among εDE, SR, SMES and αSimplex algorithms

f	εDE	SR	SMES	αSimplex
g01	**0**	**0**	**0**	6.4e-06
↑g02	**5.6e-06**	2.0e-02	1.67e-02	1.3e-02
↑g03	6.5e-09	1.9e-04	2.09e-04	**8.5e-14**
g04	**0**	2.0e-05	**0**	4.2e-11
g05	**0**	3.5	50.06	3.5e-11
g06	**0**	1.6e+02	1.85	1.3e-10
g07	**4.3e-09**	6.6e-02	1.32e-01	1.3e-04
↑g08	**0**	2.6e-17	**0**	3.8e-13
g09	**0**	3.4e-02	1.55e-02	2.9e-10
g10	**0**	5.3e+02	136.02	4.7e-06
g11	**0**	8.0e-05	1.52e-04	4.9e-16
↑g12	**0**	**0**	**0**	3.9e-10
g13	**0**	3.1e-02	1.77e-01	6.9e-02

Table 8: Experimental results on the 13 benchmark problems with varying F; 30 independent runs were performed

F	0.5	0.6	0.7	0.8	0.9	1.0
g01	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000
↑g02	0.799559	0.803251	**0.803613**	0.803401	0.802345	0.799869
↑g03	0.999997	**0.999998**	0.999991	0.999867	0.997852	0.991508
g04	-30665.538670	-30665.538670	-30665.538670	-30665.538670	-30665.538670	-30665.538670
g05	5126.736741	5126.498638	**5126.498119**	5126.498138	5126.498411	5126.499120
g06	-6591.521053	**-6961.813876**	**-6961.813876**	**-6961.813876**	**-6961.813876**	**-6961.813876**
g07	24.306674	24.306210	**24.306209**	**24.306209**	24.306215	24.344551
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630177	**680.630057**	**680.630057**	**680.630057**	**680.630057**	680.631202
g10	7049.392502	7049.248266	**7049.248021**	**7049.248021**	7049.248780	7087.910946
g11	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000
↑g12	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
g13	0.066780	**0.053950**	**0.053950**	**0.053950**	**0.053950**	**0.053950**

Table 10 summarizes the mean of the objective values in the cases of the control parameter cp alone being changed to 3, 4, 5, 6, 7 and 9 from standard settings for problems with equality constraints. In the case of $cp = 3$, the results for `g03`, `g05` and `g13` are a little worse than the results of higher cp, but the difference is small. Thus, the value of cp did not affect the results much. In this study, $cp = 5$ is selected to avoid too fast convergence to a feasible region, but from the results, a larger cp gives better results. These results show that a control parameter cp greater than 3 is an appropriate setting for many problems.

7 Conclusions

Differential evolution is a recently proposed variant of an evolutionary strategy. DE is known as a simple, efficient and robust search algorithm that can solve unconstrained optimization problems. In this study, we proposed the εDE by applying the ε constrained method to DE and showed that the εDE can solve constrained optimization problems. Also, to solve problems with equality constraints, which are very difficult problems for numerical opti-

Table 9: Experimental results on 13 benchmark problems with varying CR; 30 independent runs were performed

CR	0.5	0.6	0.7	0.8	0.9	0.95
g01	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000	-15.000000
↑g02	0.790441	0.795309	0.798266	0.801206	**0.803613**	0.803251
↑g03	0.918676	0.948491	0.976485	0.998252	0.999991	**0.999999**
g04	-30665.538670	-30665.538670	-30665.538670	-30665.538670	-30665.538670	-30665.538670
g05	5126.545569	5126.501069	5126.498416	5126.498139	5126.498119	**5126.498115**
g06	-6961.813876	-6961.813876	-6961.813876	-6961.813876	-6961.813876	-6961.813876
g07	24.520028	24.348820	24.309285	24.306219	**24.306209**	**24.306209**
↑g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.830787	680.630061	**680.630057**	**680.630057**	**680.630057**	**680.630057**
g10	7152.161161	7067.708219	7050.626313	7049.248526	**7049.248021**	**7049.248021**
g11	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000
↑g12	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
g13	0.054007	0.053956	**0.053950**	**0.053950**	**0.053950**	**0.053950**

Table 10: Experimental results on 4 benchmark problems which have equality constraints with varying cp; 30 independent runs were performed

cp	3	4	5	6	7	9
↑g03	0.999541	0.999955	0.999991	0.999998	0.999999	**1.000000**
g05	5126.503352	5126.498257	5126.498119	**5126.498110**	**5126.498110**	**5126.498110**
g11	0.750000	0.750000	0.750000	0.750000	0.750000	0.750000
g13	0.053951	**0.053950**	**0.053950**	**0.053950**	**0.053950**	**0.053950**

mization, we proposed a simple way of controlling the relaxation of the equality constraints without changing the equality constraints to relaxed inequality constraints. We showed that the εDE could solve thirteen standard benchmark problems very fast. Also, by comparing the εDE with a stochastic ranking method, a simple multimembered evolution strategy and the α constrained simplex method, which are high performance algorithms for constrained optimization, it was shown that the εDE was a more efficient and stable algorithm than the other methods. Experiments for parameter settings were performed and it was shown that the εDE could search for high quality solutions when the parameters were changed among appropriate ranges.

In the future, we will apply the εDE to various real world problems that have large numbers of decision variables and constraints.

Acknowledgment

This research is supported in part by a Grant-in-Aid for Scientific Research (c) (No.17510139, 20500138) of Japan Society for the Promotion of Science and Hiroshima City University Grant for Special Academic Research(General Studies) 7111.

Appendix

Here, we summarize 13 test problems used as benchmark test problems in this paper. The first 12 problems, g01 $\sim$ g12, are taken from Koziel and Michalewicz [16] and the 13th problem g13 from Michalewicz [19]. For the reference, the source paper of each problem is cited.

The problems g02, g03, g08 and g12 are maximization problems and the other problems are minimization problems. The problems g03, g05, g11 and g13 contain the equality constraints. The problem g12 is a difficult problem where feasible region consists of 9^3 disjointed spheres, each with the radius of 0.25.

g01 [8]: minimize $f(\boldsymbol{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$,

subject to
$$\begin{aligned}
&g_1(\boldsymbol{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0,\\
&g_2(\boldsymbol{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0,\\
&g_3(\boldsymbol{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0,\\
&g_4(\boldsymbol{x}) = -8x_1 + x_{10} \le 0,\\
&g_5(\boldsymbol{x}) = -8x_2 + x_{11} \le 0,\\
&g_6(\boldsymbol{x}) = -8x_3 + x_{12} \le 0,\\
&g_7(\boldsymbol{x}) = -2x_4 - x_5 + x_{10} \le 0,\\
&g_8(\boldsymbol{x}) = -2x_6 - x_7 + x_{11} \le 0,\\
&g_9(\boldsymbol{x}) = -2x_8 - x_9 + x_{12} \le 0,\\
&0 \le x_i \le 1\,(i = 1, \cdots, 9),\ 0 \le x_i \le 100\,(i = 10, 11, 12),\ 0 \le x_{13} \le 1
\end{aligned}$$

The optimal solution is $\boldsymbol{x}^* = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 3,\ 3,\ 3,\ 1)$ and the optimal value is $f(\boldsymbol{x}^*) = -15$.

g02 [16]: maximize $f(\boldsymbol{x}) = \left|\sum_{i=1}^{n} \cos^4 x_i - 2\prod_{i=1}^{n} \cos^2 x_i\right| \Big/ \sqrt{\sum_{i=1}^{n} i\, x_i^2}$,

subject to
$$\begin{aligned}
&g_1(\boldsymbol{x}) = 0.75 - \prod_{i=1}^{20} x_i \le 0,\quad g_2(\boldsymbol{x}) = \sum_{i=1}^{20} x_i - 7.5n \le 0,\\
&0 \le x_i \le 10\,(i = 1, \cdots, n),\ n = 20
\end{aligned}$$

The maximal value is unknown. The known best value is $f(\boldsymbol{x}) = 0.803619$ [28].

g03 [22]: maximize $f(\boldsymbol{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$,

subject to $h_1(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0,\quad 0 \le x_i \le 1\,(i = 1, \cdots, n),\quad n = 10$

The optimal solution $\boldsymbol{x}_i^* = \dfrac{1}{\sqrt{n}}\,(i = 1, \cdots, n)$ and the optimal value $f(\boldsymbol{x}^*) = 1$.

g04 [10]: minimize $f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$,
subject to
$$\begin{aligned}
g_1(\boldsymbol{x}) &= 85.334407 + 0.0056858x_2x_5\\
&\quad +0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0,\\
g_2(\boldsymbol{x}) &= -85.334407 - 0.0056858x_2x_5\\
&\quad -0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0,\\
g_3(\boldsymbol{x}) &= 80.51249 + 0.0071317x_2x_5\\
&\quad +0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0,\\
g_4(\boldsymbol{x}) &= -80.51249 - 0.0071317x_2x_5\\
&\quad -0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0,\\
g_5(\boldsymbol{x}) &= 9.300961 + 0.0047026x_3x_5\\
&\quad +0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0,\\
g_6(\boldsymbol{x}) &= -9.300961 - 0.0047026x_3x_5\\
&\quad -0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0,\\
&78 \le x_1 \le 102,\quad 33 \le x_2 \le 45,\quad 27 \le x_i \le 45\,(i = 3, 4, 5)
\end{aligned}$$

The optimal solution $\boldsymbol{x}^* = (78,\ 33,\ 29.995256025682,\ 45,\ 36.775812905788)$ and the optimal value $f(\boldsymbol{x}^*) = -30665.539$.

g05 [**11**]: minimize $f(\boldsymbol{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + \frac{0.000002}{3}x_2^3$,
subject to $g_1(\boldsymbol{x}) = x_3 - x_4 - 0.55 \leq 0$, $g_2(\boldsymbol{x}) = -x_3 + x_4 - 0.55 \leq 0$,
$h_3(\boldsymbol{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$,
$h_4(\boldsymbol{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$,
$h_5(\boldsymbol{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$,
$0 \leq x_i \leq 1200\,(i = 1, 2)$, $-0.55 \leq x_i \leq 0.55\,(i = 3, 4)$

The minimum value is unknown. The known best value is $f(\boldsymbol{x}) = 5126.4981$ [16].

g06 [**8**]: minimize $f(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$,
subject to $g_1(\boldsymbol{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$,
$g_2(\boldsymbol{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$,
$13 \leq x_1 \leq 100$, $0 \leq x_2 \leq 100$

The optimal solution $\boldsymbol{x}^* = (14.095,\ 0.84296)$ and the optimal value $f(\boldsymbol{x}^*) = -6961.81388$.

g07 [**11**]: minimize $f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2$
$+2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$,

subject to $g_1(\boldsymbol{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$,
$g_2(\boldsymbol{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$,
$g_3(\boldsymbol{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$,
$g_4(\boldsymbol{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$,
$g_5(\boldsymbol{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$,
$g_6(\boldsymbol{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$,
$g_7(\boldsymbol{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$,
$g_8(\boldsymbol{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$,
$-10 \leq x_i \leq 10\,(i = 1, \cdots, 10)$

The optimal solution is $\boldsymbol{x}^* = (2.171996,\ 2.63683,\ 8.773926,\ 5.095984,\ 0.9906548,\ 1.430574,\ 1.321644,\ 9.828726,\ 8.280092,\ 8.375927)$ and the optimal value $f(\boldsymbol{x}^*) = 24.306209$.

g08 [**16**]: maximize $f(\boldsymbol{x}) = \dfrac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$,

subject to $g_1(\boldsymbol{x}) = x_1^2 - x_2 + 1 \leq 0$, $g_2(\boldsymbol{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$, $0 \leq x_i \leq 10\,(i = 1, 2)$

The optimal solution $\boldsymbol{x}^* = (1.2279713,\ 4.2453733)$ and the optimal value $f(\boldsymbol{x}^*) = 0.095825$.

g09 [**16**]: minimize $f(\boldsymbol{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2$
$+x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$,
subject to $g_1(\boldsymbol{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$,
$g_2(\boldsymbol{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$,
$g_3(\boldsymbol{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$,
$g_4(\boldsymbol{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$,
$-10 \leq x_i \leq 10\,(i = 1, \cdots, 7)$

The optimal solution $\boldsymbol{x}^* = (2.330499,\ 1.951372,\ -0.4775414,\ 4.365726,\ -0.6244870,\ 1.038131,\ 1.594227)$ and the optimal value $f(\boldsymbol{x}^*) = 680.6300573$.

g10 **[11]:** minimize $f(\boldsymbol{x}) = x_1 + x_2 + x_3$,
subject to $g_1(\boldsymbol{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$, $g_2(\boldsymbol{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$,
$g_3(\boldsymbol{x}) = -1 + 0.01(x_8 - x_5) \leq 0$,
$g_4(\boldsymbol{x}) = -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0$,
$g_5(\boldsymbol{x}) = -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0$,
$g_6(\boldsymbol{x}) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0$,
$100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000\,(i = 2,\ 3)$,
$10 \leq x_i \leq 1000\,(i = 4, \cdots, 8)$

The maximal value is unknown. The known best value is $f(\boldsymbol{x}) = 7049.3307$.

g11 **[16]:** minimize $f(\boldsymbol{x}) = x_1^2 + (x_2 - 1)^2$,
subject to $h(\boldsymbol{x}) = x_2 - x_1^2 = 0$, $-1 \leq x_i \leq 1\,(i = 1, 2)$

The optimal solution is $\boldsymbol{x}^* = (\pm\frac{1}{\sqrt{2}}, \frac{1}{2})$ and the optimal value $f(\boldsymbol{x}^*) = 0.75$.

g12 **[16]:** maximize $f(\boldsymbol{x}) = \frac{1}{100}\{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2\}$
subject to $g(\boldsymbol{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$,
$0 \leq x_i \leq 10\,(i = 1, 2, 3)$, $p, q, r = 1, 2, \cdots, 9$

The optimal solution is $\boldsymbol{x}^* = (5,\ 5,\ 5)$ and the optimal value $f(\boldsymbol{x}^*) = 1$.

g13 **[11]:** minimize $f(\boldsymbol{x}) = e^{x_1 x_2 x_3 x_4 x_5}$,
subject to $h_1(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$,
$h_2(\boldsymbol{x}) = x_2 x_3 - 5 x_4 x_5 = 0$,
$h_3(\boldsymbol{x}) = x_1^3 + x_2^3 + 1 = 0$,
$-2.3 \leq x_i \leq 2.3\,(i = 1, 2)$, $-3.2 \leq x_i \leq 3.2\,(i = 3, 4, 5)$

The optimal solution is $\boldsymbol{x}^* = (-1.717143,\ 1.595709,\ 1.827247,\ -0.7636413,\ -0.763645)$ and the optimal value $f(\boldsymbol{x}^*) = 0.0539498$.

References

[1] 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* 59 (2004) 1989–2017.

[2] 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.), University of Vaasa, Vaasa, Finland, 1997, pp. 49–62.

[3] 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* 191 (2002) 1245–1287.

[4] K. Deb, An Efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering* 186 (2000) 311–338.

[5] A.E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

[6] A.I. El-Gallad, M.E. El-Hawary and A.A. Sallam, Swarming of intelligent particles for solving the nonlinear constrained optimization problem, *Engineering Intelligent Systems for Electrical Engineering and Communications* 9 (2001) 155–163.

[7] R. Farmani and J.A. Wright, Self-adaptive fitness formulation for constrained optimization, *IEEE Transactions on Evolutionary Computation* 7 (2003) 445–455.

[8] C. Floundas and P. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Lecture Notes in Computer Science Vol.455, Springer-Verlag, 1990.

[9] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, 1989.

[10] D. Hmmelblau, *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.

[11] W. Hock and K. Schittkowski, *Test examples for nonlinear programming codes*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 1981.

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

[13] S. Kazarlis and V. Petridis, Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms, in *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V), Amsterdan, The Netherlands*, A.E. Eiben,T. Bäck, M. Schoenauer and H.-P. Schwefel (eds.), Lecture Notes in Computer Science Vol. 1498, Springer-Verlag, Heidelberg, 1998, pp. 211–220.

[14] J. Kennedy and R.C. Eberhart, Particle swarm optimization, in *Proceedings of IEEE International Conference on Neural Networks*, Vol. 1498 of *Lecture Notes in Computer Science*, vol.IV, IEEE Press, Perth, Australia 1995, pp. 1942–1948.

[15] J. Kennedy and R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, 2001.

[16] S. Koziel and Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, *Evolutionary Computation* 7 (1999) 19–44.

[17] E. Mezura-Montes and C.A.C. Coello, A simple multimembered evolution strategy to solve constrained optimization problems, *IEEE Trans. on Evolutionary Computation* 9 (2005) 1–17.

[18] Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, J.R. McDonnell, R.G. Reynolds and D.B. Fogel (eds.), The MIT Press, Cambridge, Massachusetts, 1995, pp. 135–155.

[19] Z. Michalewicz, Genetic algorithms, numerical optimization and constraints, in *Proceedings of the 6th International Conference on Genetic Algorithms*, Pittsburgh, 1995, pp. 151–158.

[20] Z. Michalewicz and N. Attia, Evolutionary optimization of constrained problems, in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, A. Sebald and L. Fogel (eds.), World Scientific Publishing, River Edge, NJ, 1994, pp. 98–108.

[21] Z. Michalewicz and G. Nazhiyath, Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints, in *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, D.B. Fogel (ed.), IEEE Press, Piscataway, New Jersey, 1995, pp. 647–651.

[22] Z. Michalewicz, G. Nazhiyath and M. Michalewicz, A note on usefullness of geometrical-crossover of numerical optimization problems, in *Proc. 5th Annual Conference on Evolutionary Programming*, L.J. Fogel, P.J. Angeline and T. Bäck (eds.), MIT Press, 1996, pp. 305–312.

[23] Z. Michalewicz and M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation* 4 (1996) 1–32.

[24] J.A. Nelder and R. Mead, A simplex method for function minimization, *Computer Journal* 7 (1965) 308–313.

[25] M. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, *Computer Journal* 7 (1964) 155–162.

[26] 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* 6 (2002) 38–44.

[27] C. Reeves,Genetic algorithms for the operations researcher, *INFORMS Journal on Computing* 9 (1997) 231–250.

[28] T.P. Runarsson and X. Yao, Stochastic ranking for constrained evolutionary optimization, *IEEE Transactions on Evolutionary Computation* 4 (2000) 284–294.

[29] T.P. Runarsson and X. Yao, Evolutionary search and constraint violations, in *Proceedings of the 2003 Congress on Evolutionary Computation*, Vol. 2, IEEE Service Center, Piscataway, New Jersey, 2003, pp. 1414–1419.

[30] P.D. Surry and N.J. Radcliffe, The COMOGA method: Constrained optimisation by multiobjective genetic algorithms, *Control and Cybernetics* 26 (1997) 391–412.

[31] R. Storn and K. Price, Minimizing the real functions of the ICEC'96 contest by differential evolution, in *Proc. of the International Conference on Evolutionary Computation*, 1996, pp. 842–844.

[32] R. Storn and K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization* 11 (1997) 341–359.

[33] T. Takahama and S. Sakai, Tuning fuzzy control rules by α constrained method which solves constrained nonlinear optimization problems, *The Transactions of the Institute of Electronics, Information and Communication Engineers*, J82-A, (1999) 658–668 (in Japanese).

[34] T. Takahama and S. Sakai, Learning fuzzy control rules by α constrained Powell's method, in *Proceedings of 1999 IEEE International Fuzzy Systems Conference*, Vol. 2, Seoul, Korea, 1999, pp. 650–655.

[35] 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*, 83 (2000) 1–12.

[36] T. Takahama and S. Sakai, Learning fuzzy control rules by α-constrained simplex method, *The Transactions of the Institute of Electronics, Information and Communication Engineers* J83-D-I (2000) 770–779 (in Japanese).

[37] T. Takahama and S. Sakai, Constrained optimization by α constrained genetic algorithm (αGA), *The Transactions of the Institute of Electronics, Information and Communication Engineers* J86-D-I (2003) 198–207 (in Japanese).

[38] T. Takahama and S. Sakai, Learning fuzzy control rules by α-constrained simplex method, *Systems and Computers in Japan* 34 (2003) 80–90.

[39] T. Takahama and S. Sakai, Constrained optimization by α constrained genetic algorithm (αGA), *Systems and Computers in Japan* 35 (2004) 11–22.

[40] T. Takahama and S. Sakai, Constrained optimization by the α constrained particle swarm optimizer, *Journal of Advanced Computational Intelligence and Intelligent Informatics* 9 (2005) 282–289.

[41] 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)*, Muroran, Japan, 2005, pp. 1019–1029.

[42] T. Takahama and S. Sakai, Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations, *IEEE Transactions on Evolutionary Computation* 9 (2005) 437–451.

[43] S. Venkatraman and G.G. Yen, A generic framework for constrained optimization using genetic algorithms, *IEEE Trans. on Evolutionary Computation* 9 (2005) 424–435.

Manuscript received 21 February 2008
revised 14 July 2008, 25 November 2008
accepted for publication 5 December 2008

Tetsuyuki Takahama
Department of Intelligent Systems, Hiroshima City University
Asaminami-ku, Hiroshima, 731-3194 Japan
E-mail address: takahama@info.hiroshima-cu.ac.jp

Setsuko Sakai
Faculty of Commercial Sciences, Hiroshima Shudo University
Asaminami-ku, Hiroshima, 731-3195 Japan
E-mail address: setuko@shudo-u.ac.jp