Solving engineering optimization problems by constrained differential evolution with nearest neighbor comparison

Pham Hoang Anh¹ and Nguyen Xuan Thanh¹

¹ National University of Civil Engineering, 55 Giai Phong Road, Hanoi, Vietnam
Email: anhpham.nuce@gmail.com

Abstract

This contribution introduces a differential evolution (DE) algorithm to solve constrained engineering optimization problems. The proposed DE optimization strategy combines the ε constrained method to handle constraints with a relatively simple mechanism to reduce the number of function evaluations, the nearest neighbor comparison method. The algorithm is validated using four benchmark engineering design problems and the results indicate that the proposed scheme is efficient to solve this kind of optimization problems because it obtains good results with a small number of objective functions evaluations.

Keywords: Engineering optimization problem, differential evolution, constraint handling techniques, nearest neighbor comparison.

1. Introduction

Engineering optimization problems arising from modern engineering design process often involve inequality and/or equality constraints. Most of these constrained optimization problems (COPs) are complex and difficult to solve by traditional optimization techniques (Ravindran et al., 2006). Evolutionary algorithms (EAs) for the COPs have received considerable attention and have been successfully applied in many real applications (Michalewicz and Schoenauer, 1996; Coello, 2002; Mezura-Montes and Coello, 2011).

Among different EAs, differential evolution (DE) (Storn and Price, 1997) is considered as one of the most efficient algorithm and suitable for various engineering problems. The advantage of DE is that it has simple structure, requires few control parameters and highly supports parallel computation (Das and Suganthan, 2011). Together with the constraint-handling techniques, DE has been applied to the COPs (Storn, 1999; Takahama and Sakai, 2006; Mezura-Montes et al., 2007; Wang and Cai, 2011; Wang and Cai, 2012; Gong et al., 2014; Silva et al., 2011).

However, one of the main issues in applying DE is its expensive computation requirement. It is from the fact that evolutionary algorithm (EA) often needs to evaluate objective function as well as constraints thousand times to get a well acceptable solutions. A simple method, the nearest neighbor comparison, has been proposed to reduce the number of function evaluations effectively (Pham, 2015). This method uses a nearest neighbor in the search population to judge a new point whether it is worth evaluating, i.e. the function evaluation of a solution is omitted when the fitness of its nearest point in the search population is worse than that of the compared point. The nearest neighbor comparison (NNC) method has been proposed for unconstrained optimization (Pham, 2015) and fuzzy structural analysis (Pham et al., 2014).
In this study, the NNC method is proposed to constrained optimization. In order to use the nature of NNC, the \( \varepsilon \) constrained method (Takahama and Sakai, 2005) is applied to handle constraints. The \( \varepsilon \) constrained method can transform algorithms for unconstrained problems into algorithms for constrained problems using the \( \varepsilon \) level comparison that compares search points based on their pair of fitness value and their constraint violation. It has been shown that, the application of \( \varepsilon \) constrained method to DE (\( \varepsilon \) constrained DE) could solve constrained problems successfully and stably (Takahama and Sakai, 2006, 2009, 2010a,b), including engineering optimization problems (Takahama and Sakai, 2006). The proposed constrained DE in this paper is defined by applying the NNC method to the \( \varepsilon \) level comparison. Thus, it is expected that both the number of fitness evaluations and the number of constraint evaluations can be reduced. The effectiveness of the proposed constrained DE is shown by solving four well-known benchmark engineering design problems and comparing the results with those of \( \varepsilon \) constrained DE (\( \varepsilon \)DE).

In section 2, the \( \varepsilon \) constrained method for constrained optimization is briefly reviewed. The new constrained DE with the NNC method, denoted as \( \varepsilon \)DE-NNC, is described in section 3. In section 4, numerical results on the four engineering design problems are shown and the results of the \( \varepsilon \)DE-NNC are compared with those of the \( \varepsilon \)DE. Conclusions are given in section 5.

2. Constrained optimization and the \( \varepsilon \) constrained method

2.1. Constrained optimization problems

In this work, we consider the following optimization problem with equality constraints, inequality constraints and boundary constraints:

\[
\begin{align*}
  \text{minimize} & \quad f(x) \\
  \text{subject to} & \quad g_j(x) \leq 0, \quad j = 1, \ldots, q \\
  & \quad h_j(x) = 0, \quad j = 1 + q, \ldots, m \\
  & \quad l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n
\end{align*}
\]

where \( x \) is a \( n \) dimension vector, \( f(x) \) is the \( i \)-th decision variable of \( x \), \( f(x) \) is an objective function, \( g_j(x) \leq 0 \) and \( h_j(x) = 0 \) are \( q \) inequality constraints and \( m-q \) equality constraints, respectively. The functions \( f, g \), and \( h \) are real-valued functions, can be linear or nonlinear. Values \( l \) and \( u \) are the lower bound and upper bound of \( x \), respectively.

To solve the above optimization problem using EAs, the constraints can be treated as follows: (1) Constraints are used to see if a search point is feasible (the death penalty method); (2) The sum of the violation of all constraint functions is combined with the objective function to form an extended objective function (the penalty function method); (3) The constraints and the objective function are optimized by multi-objective optimization methods; (4) The constraint violation and the objective function are treated separately.

It is seen that the methods in the last category show better performance than methods in the other categories in many benchmark problems. Belonging to this category, the \( \varepsilon \) constrained method (Takahama and Sakai, 2005) are the recently developed approach, which can be applied to various unconstrained direct search algorithm to obtain constrained optimization algorithms. The \( \varepsilon \) constrained method is described briefly in the following.

2.2. The \( \varepsilon \) constrained method

In the \( \varepsilon \) constrained method, the constraint violation is defined by the maximum of all constraints or the sum of all constraints:
Solving engineering optimization problems by constrained differential evolution
with nearest neighbor comparison

\[ \phi(x) = \max \left\{ \max_j \{0, g_j(x)\}, \max_j h_j(x) \right\} \]  

\[ \phi(x) = \sum_j \left\| \max_j \{0, g_j(x)\} \right\|^p + \sum_j h_j(x) \]  

where \( p \) is a positive number.

The \( \varepsilon \) constrained method uses the \( \varepsilon \) level comparison that is defined as an order relation on a pair of objective function value and constraint violation \((f(x), \phi(x))\). Let \( f_1 \) (\( f_2 \)) and \( \phi_1 \) (\( \phi_2 \)) be the function values and the constraint violation at a point \( x_1 \) (\( x_2 \)), respectively. Then, for any \( \varepsilon \geq 0 \), \( \varepsilon \) level comparisons \( <_{\varepsilon} \) and \( \leq_{\varepsilon} \) between \((f_1, \phi_1)\) and \((f_2, \phi_2)\) are defined as follows:

\[ (f_1, \phi_1) <_{\varepsilon} (f_2, \phi_2) \iff \begin{cases} f_1 < f_2, & \text{if } \phi_1, \phi_2 < \varepsilon \text{ or } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \]  

\[ (f_1, \phi_1) \leq_{\varepsilon} (f_2, \phi_2) \iff \begin{cases} f_1 \leq f_2, & \text{if } \phi_1, \phi_2 \leq \varepsilon \text{ or } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \]  

When \( \varepsilon = \infty \), the \( \varepsilon \) level comparisons \( <_{\varepsilon} \) and \( \leq_{\varepsilon} \) become the ordinary comparisons \( < \) and \( \leq \) between function values. When \( \varepsilon = 0 \), \( <_{\varepsilon} \) and \( \leq_{\varepsilon} \) are equivalent to the lexicographic orders in which the constraint violation \( \phi(x) \) precedes the function value \( f(x) \).

The \( \varepsilon \) constrained method converts a constrained optimization problem into an unconstrained one by replacing the ordinary comparison in direct search methods with the \( \varepsilon \) level comparison (Takahama et al., 2006).

3. The \( \varepsilon \) constrained DE with the NNC method

3.1. Differential evolution

Differential Evolution (DE), which is originated by Storn and Price (1995), is a population-based optimizer. DE creates a trial individual using differences within the search population. The population is then restructured by survival individuals evolutionally. Basic of DE (based on DE/rand/1/bin) is given in the following.

We want to search for the global optima of an objective function \( f(x) \) over a continuous space: \( x = \{x_i\}, x_i \in [x_{i,\text{min}}, x_{i,\text{max}}], i = 1,2,...,n \). For each generation \( G \), a population \( P \) of \( NP \) points \( x_k, k = 1,2,...,NP \), is utilized. The initial population is generated as

\[ x_{k,i} = x_{i,\text{min}} + \text{rand}[0,1].(x_{i,\text{max}} - x_{i,\text{min}}), \]
\[ i = 1,2,...,n \]  

where \( \text{rand}[0,1] \) is a uniformly distributed random value in the range [0,1]. For each target point \( x_k, k = 1,2,...,NP \), a perturbed point \( y \) is generated according to

\[ y = x_k + F(x_{i,j} - x_{i,j}) \]

\[ \phi(x) = \max \left\{ \max_j \{0, g_j(x)\}, \max_j h_j(x) \right\} \]  

\[ \phi(x) = \sum_j \left\| \max_j \{0, g_j(x)\} \right\|^p + \sum_j h_j(x) \]
with $r_1, r_2, r_3$ are randomly chosen integers and $1 \leq r_1 \neq r_2 \neq r_3 \neq k \leq NP$; $F$ is a real and constant factor usually chosen in the interval $[0,1]$, which controls the amplification of the differential variation $(x_k - x_i)$.

Crossover is introduced to increase the diversity, creating a trial point $z$ with its elements determined by:

$$z_i = \begin{cases} 
    y_i & \text{if } (\text{rand}[0,1] \leq Cr) \text{ or } (r = i) \\
    x_{k,(i)} & \text{if } (\text{rand}[0,1] > Cr) \text{ and } (r \neq i)
\end{cases}$$

(8)

Here, $r$ is randomly chosen integer in the interval $[1,n]$; $Cr$ is user-defined crossover constant in $[0,1]$. The new point $z$ is then compared with $x_k$. If $z$ is better than $x_k$ then $z$ becomes a member in $P$ of the next generation $(G+1)$; otherwise, the old value $x_k$ is retained.

3.2. Nearest neighbor comparison method

It is desirable that only trial points which might better than the target point should be evaluated. A concept of possibly useless trial point is defined. A trial point with high possibility of being worse than the compared point is called possibly useless trial point (PUT point).

To judge a trial point whether it is a PUT point, we use its nearest neighbor, $x_{nn}$, in the population to compare with the target point. This method is named as nearest neighbor comparison (NNC). The point $x_{nn}$ nearest to the trial point $z$ is searched in the current population using distance measure. For this task, the following normalized distance measure is adopted.

$$d(x,z) = \sqrt{\sum_{i=1}^{n} \left( \frac{x_i - z_i}{\max_{k} x_{k,i} - \min_{k} x_{k,i}} \right)^2}$$

(9)

where $d(x,z)$ is distance between two points $x$ and $z$. Thus, point $x_{nn}$ has smallest distance to $z$. Comparison is then made between $x_{nn}$ and $x_k$. If $x_{nn}$ is worse than $x_k$, the trial point $z$ is possibly not better than $x_k$, and it is judged as PUT vector and evaluations of its fitness and constraint violation are not carried out.

The NNC for constrained optimization using the $\varepsilon$ constrained method can be written as follows:

**If** $(f(x_{nn}), \phi(x_{nn})) \leq (f(x_k), \phi(x_k))$

**Then** Evaluate $z$;

**If** $(f(z), \phi(z)) \leq (f(x_k), \phi(x_k))$

**Then** $x_k = z$;

**End**

where the true values at the nearest neighbor point $(f(x_{nn}), \phi(x_{nn}))$ and the parent point $(f(x_k), \phi(x_k))$ are known. Thus, the NNC can reject PUT points and omit several function evaluations.
4. Solving engineering optimization problems

4.1. Test problems and experimental conditions

In this study, four benchmark engineering design problems are tested: welded beam design, tension/compression spring design, pressure vessel design and speed reducer design. The results of the proposed εDE-NNC are compared with those of εDE.

The parameter setting for the ε level comparison is as follows: the constraint violation \( \phi \) is given by the sum of all constraints \( (p=1) \) in eq. (3) and the ε level is assigned to 0. The binary crossover and random mutation with one pair of individuals (DE/rand/1/bin) is adopted as the base algorithm. The parameters for DE are: the \( NP = 20 \), \( F = 0.8 \), \( Cr = 0.9 \), except for the speed reducer design \( NP = 65 \). The stop condition for the optimization process is when the relative accuracy value, determined by the ratio between the standard derivative and the mean of objective function values in the population, is less than \( 1e^{-4} \). For each problem, 50 independent runs are performed.

4.2. Experimental results

4.2.1. The welded beam design problem

A welded beam is designed for minimum cost subjected to some constraints (Rao, 1996). The design variables are: weld thickness \( x_1 \), length of weld \( x_2 \), width of beam \( x_3 \), thickness of beam \( x_4 \). The problem has 7 inequality constraints on shear stress, bending stress in the beam, buckling load on the bar and end deflection of the beam.

Fig. 1a shows the plot of the best function values over the number of function evaluations. In the graphs, the solid line shows optimization process by εDE-NNC. The dashed line shows optimization process by εDE. It is clearly seen in the figure that εDE-NNC is faster than εDE.

Also, the average number of evaluations of the constraints and the objective function when stop condition is met is listed in the columns labeled #func and #const of Table 1, respectively. It is noted that, the number of objective function evaluations is less than the number of constraint evaluations. The reason for this result is that in the ε constrained method, 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, the objective function is not evaluated. It can be seen from the Table 1 that εDE-NNC can omit 48.24% constraint evaluations and 49.27% function evaluations, comparing with εDE.

It is important to point out that εDE has better performance than various methods on the same problem as shown in Takahama et al., (2006).

4.2.2. The tension/compression spring design problem

The weight of a tension/compression spring is minimized with the constraints on minimum deflection, shear stress, surge frequency and limit on outside diameter (Arora, 1989). The design variables are: wire diameter \( x_1 \), the mean coil diameter \( x_2 \), and the number of active coils \( x_3 \). The problem has 4 inequality constraints.

Fig. 1b shows the plot of the best function values over the number of function evaluations. The figure shows that εDE-NNC is faster than εDE.

The average number of evaluations of the constraints and the objective function are given in Table 1. It is shown that εDE-NNC can omit 46.60% constraint evaluations and 45.01% function evaluations, comparing with εDE.

4.2.3. Pressure vessel design problem
A cylindrical vessel, capped at both ends by hemispherical heads, are designed to minimized total cost including the cost of materials forming the welding (Sandgren, 1990). The design variables are: thickness of the shell \( x_1 \), thickness of the head \( x_2 \), inner radius \( x_3 \), and the length of the cylindrical section of the vessel \( x_4 \). The variables \( x_1 \) and \( x_2 \) are discrete values, which are integer multiples of 0.0625 inches. The problem has 4 inequality constraints.

Experimental results on the problem are shown in Fig. 1c and Table 1. Clearly, \( \varepsilon \text{DE-NNC} \) requires less evaluations of constraints and objective function than \( \varepsilon \text{DE} \). Comparing with \( \varepsilon \text{DE} \), \( \varepsilon \text{DE-NNC} \) can reduce 43.68% constraint evaluations and 42.78% function evaluations.

4.2.4. Speed reducer design problem

The weight of a speed reducer is minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shaft and stress in the shaft (Golinski, 1973). The design variables are: face width \( x_1 \), module of teeth \( x_2 \), number of teeth on pinion \( x_3 \), length of the first shaft between bearings \( x_4 \), length of the second shaft between bearings \( x_5 \), diameter of the first shaft \( x_6 \), and diameter of the second shaft \( x_7 \). The problem has 11 inequality constraints.

For this problem, \( \varepsilon \text{DE-NNC} \) is also faster and requires less evaluations of functions than \( \varepsilon \text{DE} \) (Fig. 1d). There are 24.62% constraint evaluations and 22.91% objective function evaluations reduced with \( \varepsilon \text{DE-NNC} \), comparing with \( \varepsilon \text{DE} \) (Table 1)

Figure 1. Optimization results for: a) welded beam problem; b) spring problem; c) pressure vessel problem; and d) speed reducer problem
Table 1. Average constraint evaluations and function evaluations over 50 random runs with the same stop condition (relative accuracy value < 1e-4)

<table>
<thead>
<tr>
<th>Problem</th>
<th>εDE</th>
<th>εDE-NNC</th>
<th>omit (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#const</td>
<td>#func</td>
<td>#const</td>
</tr>
<tr>
<td>Welded beam</td>
<td>4596.40</td>
<td>1785.32</td>
<td>2379.18</td>
</tr>
<tr>
<td>Spring</td>
<td>3230.40</td>
<td>1239.18</td>
<td>1725.04</td>
</tr>
<tr>
<td>Pressure vessel</td>
<td>3087.20</td>
<td>1558.82</td>
<td>1738.78</td>
</tr>
<tr>
<td>Speed reducer</td>
<td>9624.33</td>
<td>4206.10</td>
<td>7254.80</td>
</tr>
</tbody>
</table>

5. Conclusion

A constrained differential evolution algorithm, εDE-NNC, was introduced for constrained optimization problem. The proposed approach applies the nearest neighbor comparison method to the ε level comparison in order to avoid unnecessary function evaluations. The εDE-NNC was used to solve 4 benchmark engineering design problems. It was shown that the εDE-NNC can reduce the evaluations of the constraints and objective function about 23% to 49%, comparing to the εDE. Therefore, the εDE-NNC can solve constrained optimization problems very effectively, especially for the problems with expensive function evaluation.

References


