Vietnam Journal of Mechanics, VAST, Vol. 38, No. 2 (2016), pp. 89 – 101
DOI:10.15625/0866-7136/38/2/6568

ENGINEERING OPTIMIZATION BY CONSTRAINED DIFFERENTIAL EVOLUTION WITH NEAREST NEIGHBOR COMPARISON

Pham Hoang Anh
National University of Civil Engineering, Hanoi, Vietnam
E-mail: anhpham.nuce@gmail.com
Received July 27, 2015

Abstract. It has been proposed to utilize nearest neighbor comparison to reduce the number of function evaluations in unconstrained optimization. The nearest neighbor comparison omits the function evaluation of a point when the comparison can be judged by its nearest point in the search population. In this paper, a constrained differential evolution (DE) algorithm is proposed by combining the ε constrained method to handle constraints with the nearest neighbor comparison method. The algorithm is tested using five benchmark engineering design problems and the results indicate that the proposed DE algorithm is able to find good results in a much smaller number of objective function evaluations than conventional DE and it is competitive to other state-of-the-art DE variants.

Keywords: Engineering optimization, differential evolution, ε constrained method, 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 [1]. Evolutionary algorithms (EAs) for the COPs have received considerable attention and have been successfully applied in many real applications [2–4].

Among different EAs, differential evolution (DE) [5] 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 [6]. Together with the constraint-handling techniques, DE has been applied to the COPs [7–12].

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 [13]. 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 [13] and fuzzy structural analysis [14].

In this study, the NNC method is proposed to constrained optimization. In order to use the nature of NNC, the ε constrained method [15] is applied to handle constraints. The ε constrained method can transform algorithms for unconstrained problems into algorithms for constrained problems using the ε 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 ε constrained method to DE (εDE) could solve constrained problems successfully and stably [16–19], including engineering optimization problems [16]. The proposed constrained DE in this paper is defined by applying the NNC method to the ε 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 five well-known benchmark engineering design problems and comparing the results with those of ε constrained DE and other state-of-the-art DE algorithms.

In section 2, the ε constrained method for constrained optimization is briefly reviewed. The new constrained DE with the NNC method, denoted as εDE-NNC, is described in section 3. In section 4, numerical results on the five engineering design problems are shown. Conclusions are given in section 5.

2. THE ε CONSTRAINED METHOD

2.1. Constrained optimization problems

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

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

where $\mathbf{x}$ is a n dimension vector, x_i is the i-th decision variable of $\mathbf{x}$, $f(\mathbf{x})$ is an objective function, $g_j(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$ are q inequality constraints and $m - q$ equality constraints, respectively. The functions f, g_j and h_j are real-valued functions, can be linear or nonlinear. Values l_i and u_i are the lower bound and upper bound of x_i, 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 ε constrained method [15] is the recently developed approach, which can be applied to various unconstrained direct search algorithms to obtain constrained optimization algorithms. The ε constrained method is described briefly in the following.

2.2. The ε constrained method

In the ε constrained method, the constraint violation is defined by the maximum of all constraints (Eq. (2)) or the sum of all constraints (Eq. (3))

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

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

where p is a positive number.

The ε constrained method uses the ε *level comparison* that is defined as an order relation on a pair of objective function value and constraint violation $(f(\mathbf{x}), \phi(\mathbf{x}))$. Let f_1 (f_2) and ϕ_1 (ϕ_2) be the function values and the constraint violation at a point $\mathbf{x}_1$ $(\mathbf{x}_2)$, respectively. Then, for any $\varepsilon \geq 0$, ε level comparisons $<_\varepsilon$ and $\leq_\varepsilon$ between (f_1, ϕ_1) and (f_2, ϕ_2) are defined as follows

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

$$(f_1, \phi_1) \leq_\varepsilon (f_2, \phi_2) \Leftrightarrow \begin{cases} f_1 \leq f_2, \text{ if } \phi_1, \phi_2 \leq \varepsilon \text{ or } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, \text{ otherwise} \end{cases} \tag{5}$$

When $\varepsilon = \infty$, the ε 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 $\phi(\mathbf{x})$ precedes the function value $f(\mathbf{x})$. Using the ε constrained method a constrained optimization problem is converted into an unconstrained one by replacing the ordinary comparison in direct search methods with the ε level comparison.

3. CONSTRAINED DE WITH THE NNC METHOD

3.1. Differential evolution

Differential Evolution (DE), which is originated by Storn and Price [20], 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 $f(\mathbf{x})$ over a continuous space: $\mathbf{x} = \{x_i\}$, $x_i \in [x_{i,\min}, x_{i,\max}]$, $i = 1, 2, \ldots, n$. For each generation G,

a population P of NP points $\mathbf{x}_k,\ k = 1, 2, \ldots, NP$, is utilized. The initial population is generated as

$$x_{k,i} = x_{i,\min} + rand[0,1].(x_{i,\max} - x_{i,\min}), \quad i = 1, 2, \ldots, n \tag{6}$$

where $rand[0,1]$ is a uniformly distributed random real value in the range [0, 1]. For each target point $\mathbf{x}_k,\ k = 1, 2, \ldots, NP$, a perturbed point $\mathbf{y}$ is generated according to

$$\mathbf{y} = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}), \tag{7}$$

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 $(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$.

Crossover is introduced to increase the diversity, creating a trial point $\mathbf{z}$ with its elements determined by

$$z_i = \begin{cases} y_i & \text{if } (rand[0,1] \leq Cr) \text{ or } (r = i) \\ x_{k,i} & \text{if } (rand[0,1] > Cr) \text{ and } (r \neq i) \end{cases} \tag{8}$$

Here, r is randomly chosen integer in the interval $[1, n]$; Cr is user-defined crossover constant in the interval [0, 1]. The new point $\mathbf{z}$ is then compared with $\mathbf{x}_k$. If $\mathbf{z}$ is better than $\mathbf{x}_k$ then $\mathbf{z}$ becomes a member in P of the next generation $(G + 1)$; otherwise, the old value $\mathbf{x}_k$ 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, $\mathbf{x}_{nn}$, in the population to compare with the target point. This method is named as *nearest neighbor comparison* (NNC). The point $\mathbf{x}_{nn}$ nearest to the trial point $\mathbf{z}$ is searched in the current population using distance measure. For this task, the following normalized distance measure is adopted.

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{i=1}^{n} \left(\frac{x_i - z_i}{\max\limits_k x_{k,i} - \min\limits_k x_{k,i}} \right)^2}, \tag{9}$$

where $d(\mathbf{x}, \mathbf{z})$ is distance between two points $\mathbf{x}$ and $\mathbf{z}$. Thus, point $\mathbf{x}_{nn}$ has smallest distance to $\mathbf{z}$. Comparison is then made between $\mathbf{x}_{nn}$ and $\mathbf{x}_k$. If $\mathbf{x}_{nn}$ is worse than $\mathbf{x}_k$, the trial point $\mathbf{z}$ is possibly not better than $\mathbf{x}_k$, 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 ε constrained method can be written as follows

If $(f(\mathbf{x}_{nn}), \phi(\mathbf{x}_{nn})) \leq_\varepsilon (f(\mathbf{x}_k), \phi(\mathbf{x}_k))$
Then *Evaluate* $\mathbf{z}$;
  If $(f(\mathbf{z}), \phi(\mathbf{z})) \leq_\varepsilon (f(\mathbf{x}_k), \phi(\mathbf{x}_k))$
  Then $\mathbf{x}_k = \mathbf{z}$;
  End
End

where the true values at the nearest neighbor point $(f(\mathbf{x}_{nn}), \phi(\mathbf{x}_{nn}))$ and the parent point $(f(\mathbf{x}_k), \phi(\mathbf{x}_k))$ 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 section, five benchmark engineering design problems are solved to test the performance of εDE-NNC. The problems are: the welded beam design [21], the tension/compression spring design [22], the pressure vessel design [23], speed reducer design [24], and the 200-bar plane truss sizing [25]. Due to space limitation, the formulations of these problems are omitted here.

The parameter setting for the ε level comparison is as follows: the constraint violation ϕ 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 of DE for welded beam design are: $NP = 30, F = 0.8, Cr = 0.9$; for spring design, pressure vessel design, and speed reducer design are: $NP = 65, F = 0.8, Cr = 0.9$; and for the 200-bar truss sizing are: $NP = 50, F = 0.5, Cr = 0.9$.

In the first set of experiments, the welded beam, the tension/compression spring, the pressure vessel, and speed reducer problems are considered. Firstly, the εDE-NNC is compared with εDE. 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. The average number of constraint evaluations and number of function evaluations over 50 random runs are given in Tab. 1. Secondly, εDE-NNC is compared with five other DE algorithms. The five DE algorithms are: (1) multiple trial vectors differential evolution (MDDE) [9], (2) differential evolution with level comparison (DELC) [26], (3) constrained modified differential evolution (COMDE) [27], (4) multi-view differential evolution (MVDE) [28], and (5) improved constrained differential evolution (rank-iMDDE) [29]. In these experiments, the optimization termination is controlled by the maximum number of evaluations, MaxNEs.

The optimal sizing of 200-bar plane truss presents a relative large-scale optimization with 29 design variables. The proposed εDE-NNC is compared with the adaptive differential evolution algorithm (ADEA) [25] and the adaptive differential evolution with optional external archive (JADE) [30]. Twenty runs are performed with termination criterion of maximum number of evaluations, MaxNEs = 20000.

Table 1. Average constraint evaluations and function evaluations over 50 random runs with the same stop condition (relative accuracy value $<$ 1e-4)

Problem	Method	fitness	No. evaluation		No. skip			Omit (%)
			#const	#func	#skip	Fail-skip	rate (%)	
Welded beam	εDE-NNC	1.72530398	3672	1315	3771	167	4.42	49.42
	εDE	1.72520666	7260	2734	-	-	-	-
Spring	εDE-NNC	0.01266614	6935	2104	7387	319	4.32	48.25
	εDE	0.01266572	13402	4578	-	-	-	-
Pressure vessel	εDE-NNC	6060.10916	5822	2632	5514	376	6.82	45.78
	εDE	6059.91411	10738	5163	-	-	-	-
Speed reducer	εDE-NNC	2995.38015	7117	3193	5874	552	9.40	26.76
	εDE	2994.99483	9717	4242	-	-	-	-

Table 2. The best results obtained by εDE-NNC for each problem

Problem	Best solution	Best fitness
Welded beam	0.205729639786079, 3.470488665628002, 9.036623910357633, 0.205729639786080	1.724852308597365
Spring	0.051689031917057, 0.356717038149551, 11.289006887322081	0.012665232788377
Pressure vessel	0.8125, 0.4375, 42.0984455958548, 176.6365958424412	6059.714335048453
Speed reducer	3.5, 0.7, 17, 7.3, 7.715319912497795, 3.350214666225438, 5.286654465026051	2994.471066247639
200-bar truss	0.104094378466599, 0.974548756359698, 0.110520001966675, 0.120671688064705, 1.970068748667075, 0.231492528644432, 0.104293151372446, 3.147051903202999, 0.131659736442437, 4.142349528145140, 0.317129899931068, 0.116306449963694, 5.409951291487695, 0.116082739727849, 6.461642218197368, 0.481690461779939, 0.333020346450255, 8.000360212524122, 0.141619982705281, 8.992097253445209, 0.761567718928248, 0.200262604294582, 11.100410974573968, 0.167270637939214, 12.165852816206018, 0.925872725405440, 6.080582730332873, 10.892946939448695, 14.020052424656935	25267.90428363515

The best result obtained by εDE-NNC in each problem is listed in Tab. 2.

4.2. Experimental results and discussion

4.2.1. *The welded beam design problem*

With the relative accuracy of 1e-4, 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. Fig. 1b shows the plot of success evaluation rate, defined by the rate of the success evaluations on actual evaluations. We can see that εDE-NNC has higher success rate than εDE.

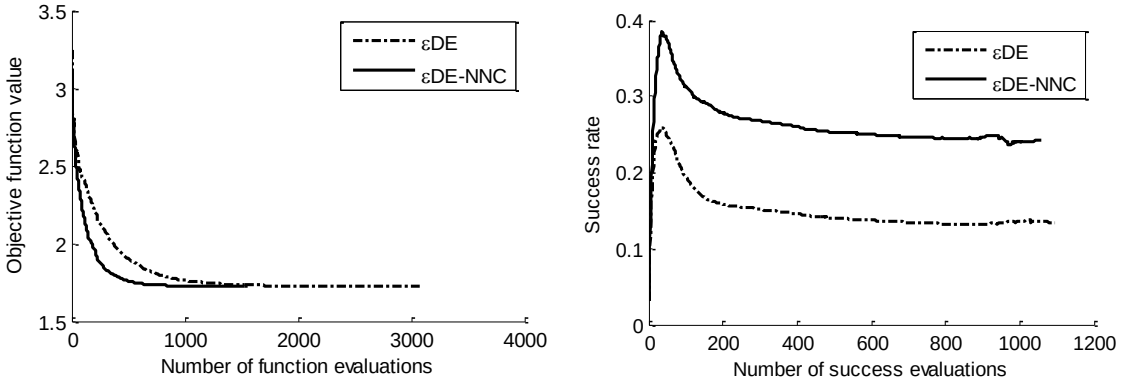


Fig. 1. Optimization of welded beam problem with accuracy of 1e-4

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 Tab. 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 that εDE-NNC can omit 49.42% evaluations, comparing with εDE. Moreover, there is only 4.42% of points skipped are good points, which implies that the judgment by the nearest neighbor comparison is quite accurate.

It is important to point out that εDE has better performance than various methods on the same problem as shown in [8].

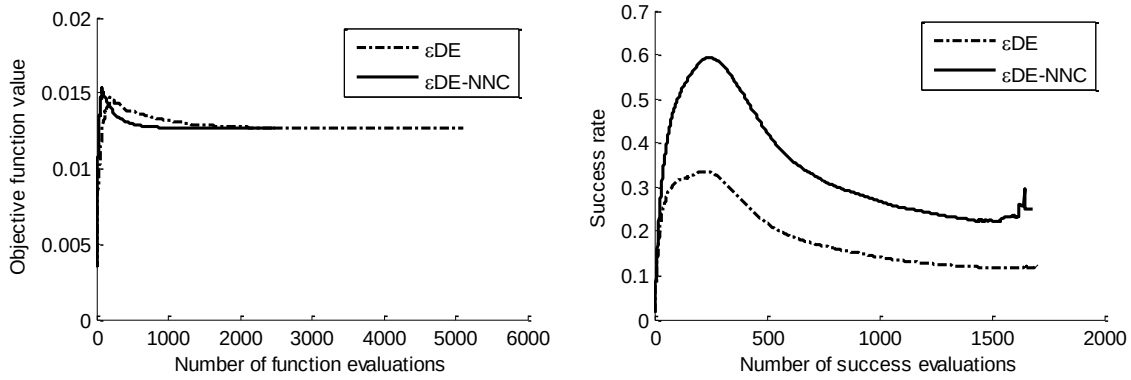
Tab. 3 shows the results obtained by εDE-NNC with 15000 constraint evaluations. The results of other algorithms are also listed. A result in boldface means a better (or best) solution obtained. From the results in Tab. 3, we can see that εDE-NNC obtains the optimal solution in all runs. Moreover, with the same MaxNEs, εDE-NNC gives smallest standard deviation value compared with other DE algorithms. The number of actual fitness evaluations is even much smaller as shown in the parentheses.

Table 3. Comparison on the results of welded beam design

Algorithm	Best	Mean	Worst	Std	MaxNEs
MDDE [9]	1.725	1.725	1.725	**1.00e-15**	24000
DELC [26]	**1.724852**	**1.724852**	**1.724852**	4.10e-13	20000
COMDE [27]	**1.724852309**	**1.724852309**	**1.724852309**	1.6e-12	20000
MVDE [28]	1.7248527	1.7248621	1.7249215	7.88e-06	**15000**
rank-iMDDE [29]	**1.724852309**	**1.724852309**	**1.724852309**	7.71e-11	**15000**
εDE-NNC	**1.724852308597**	**1.724852308597**	**1.724852308597**	**5.09e-15**	**15000** (5772)

4.2.2. *The tension/compression spring design problem*

Fig. 2a shows the plot of the best function values corresponding to relative accuracy of 1e-4 over the number of function evaluations. The figure shows that εDE-NNC is faster than εDE. Fig. 2b shows the plot of success evaluation rate. The success rate of εDE-NNC is obviously higher than that of εDE. The average number of evaluations of the constraints and the objective function are given in Tab. 1. It is shown that εDE-NNC can omit 48.25% evaluations, comparing with εDE. Moreover, there is only few percents (4.32%) of points skipped are wrongly judged.



(a) (b)

Fig. 2. Optimization of spring problem with accuracy of 1e-4

The optimization results with 20000 evaluations are compared with the results obtained by other DE algorithms in Tab. 4. We can observe that εDE-NNC can obtain the best solution with smallest standard deviation and gets the best mean value. With about one-half evaluations, εDE-NNC provides as good results as other algorithms, such as rank-iMDDE, COMDE, DELC.

Table 4. Comparison on the results of tension/compression spring design

Algorithm	Best	Mean	Worst	Std	MaxNEs
MDDE [9]	**0.012665**	0.012666	0.012674	2.00e-06	24000
DELC [26]	**0.012665233**	0.012665267	0.012665575	1.30e-07	20000
COMDE [27]	**0.012665232**	0.012667168	0.012676809	3.09e-06	24000
MVDE [28]	0.012665273	0.012667324	0.012719055	2.45e-06	**10000**
rank-iMDDE [29]	**0.012665233**	0.012665264	0.01266765	2.45e-07	19565
εDE-NNC	**0.012665232788**	**0.012665232792**	**0.012665232816**	**5.09e-12**	20000 (6630)
	0.0126652359810	0.012665280131	0.012665508356	**5.58e-08**	**10000**

4.2.3. *Pressure vessel design problem*

Experimental results on the problem with relative accuracy of 1e-4 are shown in Fig. 3 and Tab. 1. Clearly, εDE-NNC requires less evaluations of constraints and objective function than εDE. Comparing with εDE, εDE-NNC can omit 45.78% evaluations and has higher success evaluation rate. The wrong judgment is also low (6.82% of actual skipped evaluations as shown in Tab. 1).

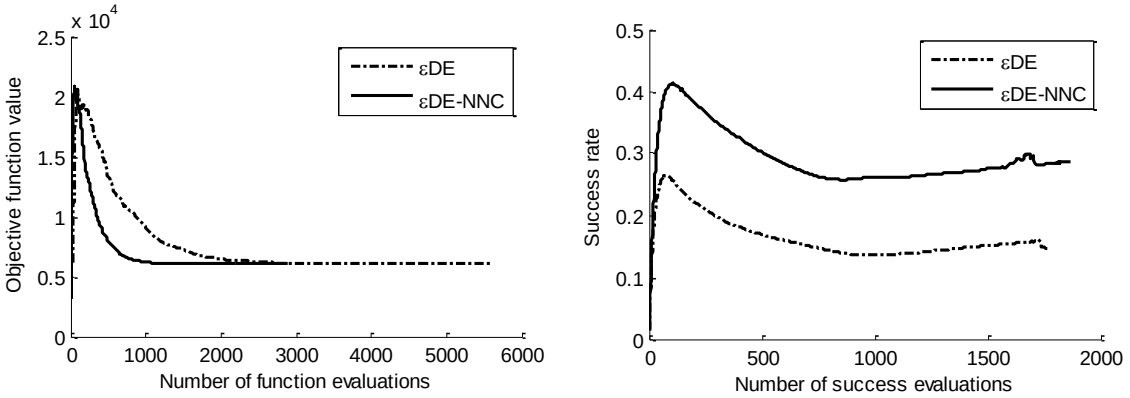


Fig. 3. Optimization of pressure vessel problem with accuracy of 1e-4

For this problem, there are five algorithms (i.e., MDDE, DELC, COMDE, rank-iMDDE, and εDE-NNC) that can obtain the optimal solution. According to the results given in Tab. 5, the proposed εDE-NNC is superior to MDDE, DELC, and COMDE with respect to the number of evaluations and has smaller standard deviation than that of rank-iMDDE.

Table 5. Comparison on the results of pressure vessel design

Algorithm	Best	Mean	Worst	Std	MaxNFEs
MDDE [9]	**6059.702**	**6059.702**	**6059.702**	**1.00e-12**	24000
DELC [26]	**6059.7143**	**6059.7143**	**6059.7143**	**2.10e-11**	30000
COMDE [27]	**6059.714335**	**6059.714335**	**6059.714335**	3.62e-10	30000
MVDE [28]	6059.714387	6059.997236	6090.533528	2.91e+00	**15000**
rank-iMDDE [29]	**6059.714335**	**6059.714335**	**6059.714335**	7.47e-07	**15000**
εDE-NNC	**6059.714335048**	**6059.714335049**	**6059.714335051**	9.08e-10	**15000** (8708)

4.2.4. *Speed reducer design problem*

For this problem, εDE-NNC is also faster and requires less function evaluations than εDE (Fig. 4). There are 26.76% evaluations reduced with εDE-NNC, comparing with εDE (Tab. 1). The wrong judgment is less than ten percent (9.4%) of actual omitted evaluations.

Tab. 6 shows the results with 20000 evaluations. In this problem, εDE-NNC can obtain the optimal solution. However, it is slightly worse than rank-iMDDE and COMDE.

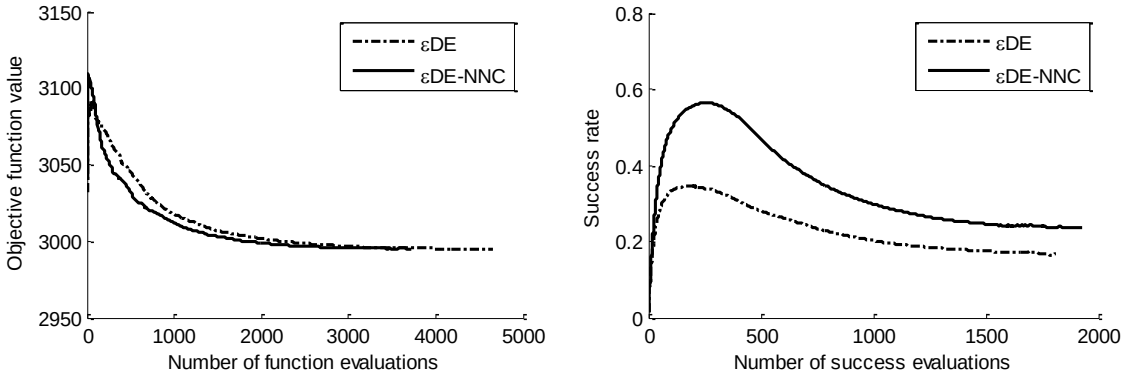


Fig. 4. Optimization of speed reducer problem with accuracy of 1e-4

4.2.5. *200-bar truss sizing problem*

The displacement and stress of the structure are calculated by finite-element analysis. The optimization results with 20000 evaluations are compared with the results obtained by ADEA and JADE given in [25] (Tab. 7). We can observe that εDE-NNC can obtain better solution than ADEA and JADE.

Table 6. Comparison on the results of speed reducer design

Algorithm	Best	Mean	Worst	Std	MaxNEs
MDDE [9]	2996.357	2996.367	2996.369	8.20e-03	24000
DELC [26]	**2994.471066**	**2994.471066**	**2994.471066**	**1.90e-12**	30000
COMDE [27]	**2994.471066**	**2994.471066**	**2994.471066**	**1.54e-12**	21000
MVDE [28]	**2994.471066**	**2994.471066**	2994.471069	2.82e-07	30000
rank-iMDDE [29]	**2994.471066**	**2994.471066**	**2994.471066**	**7.93e-13**	**19920**
εDE-NNC	**2994.471066247**	2994.471069502	2994.471079142	2.73E-06	20000 (9052)

Table 7. Comparison on the results of 200-bar truss sizing problem

Algorithm	Best	Mean	Worst	Std	MaxNEs
JADE [25]	25610.2086	25985.05665	-	**177.03358**	20000
ADEA [25]	25800.5708	26851.1460	-	1038.1452	20000
εDE-NNC	**25267.904283635**	**25432.171192683**	26298.638429529	224.06769	20000 (5618)

5. CONCLUSION

This paper presented the combination of the nearest neighbor comparison, previously used within unconstrained optimization, and the ε constrained method for handling constraints and proposed the εDE-NNC for constrained engineering design problem. The performance of εDE-NNC was evaluated by five widely used engineering benchmark design problems. It was observed that εDE-NNC reduced the evaluations of the constraints and objective function about 26% to 49% compared to εDE. With low function evaluation requirement, εDE-NNC is also very competitive when comparing with other DE algorithms. Therefore, the εDE-NNC can solve constrained engineering optimization problems very effectively, especially for the problems with expensive objective functions.

ACKNOWLEDGMENTS

This work is supported by National University of Civil Engineering, Vietnam (NUCE) under grant research number 98-2015/KHXD.

REFERENCES

[1] A. Ravindran, G. V. Reklaitis, and K. M. Ragsdell. *Engineering optimization: methods and applications*. John Wiley & Sons, (2006).

[2] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, **4**, (1), (1996), pp. 1–32.
[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**, (11), (2002), pp. 1245–1287.
[4] E. Mezura-Montes and C. A. C. Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, **1**, (4), (2011), pp. 173–194.
[5] R. Storn and K. Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, **11**, (4), (1997), pp. 341–359.
[6] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, **15**, (1), (2011), pp. 4–31.
[7] R. Storn. System design by constraint adaptation and differential evolution. *IEEE Transactions on Evolutionary Computation*, **3**, (1), (1999), pp. 22–34.
[8] 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*, Vol. 3, (2006), pp. 2322–2327.
[9] E. Mezura-Montes, C. A. C. Coello, J. Velázquez-Reyes, and L. Muñoz-Dávila. Multiple trial vectors in differential evolution for engineering design. *Engineering Optimization*, **39**, (5), (2007), pp. 567–589.
[10] Y. Wang and Z. Cai. Constrained evolutionary optimization by means of $(\mu\mu+\lambda\lambda)$-differential evolution and improved adaptive trade-off model. *Evolutionary Computation*, **19**, (2), (2011), pp. 249–285.
[11] Y. Wang and Z. Cai. Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *Evolutionary Computation, IEEE Transactions on*, **16**, (1), (2012), pp. 117–134.
[12] E. K. da Silva, H. J. C. Barbosa, and A. C. C. Lemonge. An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization. *Optimization and Engineering*, **12**, (1-2), (2011), pp. 31–54.
[13] H. A. Pham. Reduction of function evaluation in differential evolution using nearest neighbor comparison. *Vietnam Journal of Computer Science*, **2**, (2), (2015), pp. 121–131.
[14] H. A. Pham, X. T. Nguyen, and V. H. Nguyen. Fuzzy structural analysis using improved differential evolutionary optimization. In *Proceedings of the International Conference on Engineering Mechanics and Automation (ICEMA 3)*, Hanoi, (2014). pp. 492–498.
[15] T. Takahama and S. Sakai. Constrained optimization by ε constrained particle swarm optimizer with ε-level control. In *Proceedings of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST05)*. Springer, (2005), pp. 1019–1029.
[16] T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, (2006), pp. 308–315.
[17] T. Takahama and S. Sakai. Fast and stable constrained optimization by the ε constrained differential evolution. *Pacific Journal of Optimization*, **5**, (2), (2009), pp. 261–282.
[18] T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, (2010), pp. 1680–1688.

[19] T. Takahama and S. Sakai. Efficient constrained optimization by the ε constrained adaptive differential evolution. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, (2010), pp. 2052–2059.
[20] R. Storn and K. Price. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. International Computer Science Institute, Berkeley, (1995).
[21] S. S. Rao. *Engineering optimization*. New York: Wiley, 3rd edition, (1996).
[22] J. Arora. *Introduction to optimum design*. McGrawHill, (1989).
[23] E. Sandgren. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, **112**, (2), (1990), pp. 223–229.
[24] J. Golinski. An adaptive optimization system applied to machine synthesis. *Mechanism and Machine Theory*, **8**, (4), (1974), pp. 419–436.
[25] S. Bureerat and N. Pholdee. Optimal truss sizing using an adaptive differential evolution algorithm. *Journal of Computing in Civil Engineering*, (2015). Doi:10.1061/(ASCE)CP.1943-5487.0000487.
[26] L. Wang and L.-P. Li. An effective differential evolution with level comparison for constrained engineering design. *Structural and Multidisciplinary Optimization*, **41**, (6), (2010), pp. 947–963.
[27] A. W. Mohamed and H. Z. Sabry. Constrained optimization based on modified differential evolution algorithm. *Information Sciences*, **194**, (2012), pp. 171–208.
[28] V. V. De Melo and G. L. C. Carosio. Investigating multi-view differential evolution for solving constrained engineering design problems. *Expert Systems with Applications*, **40**, (9), (2013), pp. 3370–3377.
[29] W. Gong, Z. Cai, and D. Liang. Engineering optimization by means of an improved constrained differential evolution. *Computer Methods in Applied Mechanics and Engineering*, **268**, (2014), pp. 884–904.
[30] J. Zhang and A. C. Sanderson. JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, **13**, (5), (2009), pp. 945–958.