AN IMPROVED DIFFERENTIAL EVOLUTION OPTIMIZATION ALGORITHM

Abazar Shamekhi[1]
Department of Mechanical Engineering, University of Alberta, Edmonton, AB, T6G 2G8, Canada
Email: Shamekhi@ualberta.ca

ABSTRACT
The most important challenge in some optimization problems is CPU time. The performance and convergence speed of optimization algorithms have the most important effect on CPU time. In this work, the original differential evolution algorithm has been modified in order to increase the convergence speed of the optimization algorithm. These modifications can increase the probability of selection of evolved individuals and consequently can increase the performance of differential evolution algorithm. The performance characteristics of the modified differential evolution algorithm have been compared with those of the original differential evolution algorithm. Results obtained show that the modified differential evolution algorithm is able to converge and find the optimum point faster compared to the original algorithm. These modifications have been applied not only to different versions of differential evolution algorithm but also to adaptive differential evolution algorithm in order to investigate the convergence speed of these modified methods. According to the results obtained, the proposed modifications were able to improve the performance of the different versions of differential evolution and even adaptive differential evolution algorithms. Finally, the proposed modifications have been tested on several optimization problems to evaluate the effect of these modifications on the convergence speed of the algorithm in different applications. In all cases, the modified differential evolution algorithm is faster than the original algorithm. The results of the proposed algorithm have been verified with analytical results.

Keywords: *Differential Evolution, Optimization, CPU time, Evolutionary algorithm, Stochastic optimization.*

1. INTRODUCTION

The history of optimization is as old as the history of life. All creatures try to optimize their routes in the complex conditions of life to survive. Since 1950 scientists were interested in using being patterns to design powerful optimization methods for solving complex problems. These kinds of optimization techniques are called evolutionary computation [1]. Evolutionary optimization algorithms have some advantages that allow optimization of fitness function of discrete variables. These kinds of techniques use iterative try and error progress, similar to development and evolution in living creatures to find the better fitness at the given circumstances. The origin of evolutionary computing techniques initiated in 1950s, with the idea of using Darwinian principles for automated problem solving. John Henry Holland from the University of Michigan at Ann Arbor, introduced genetic algorithm (GA) for solving practical optimization problems [2]. After that, many researchers employed genetic algorithm for optimization of scientific problems [3-5]. In 1995 R. Storn and K. V. Price invented another kind of evolutionary algorithm called Differential Evolution (DE) [6]. They introduced their new optimization algorithm at the First International Contest on Evolutionary Optimization in 1996 [7]. One of the advantages of DE algorithm compared to GA is that this method does not require the transformation of the variables into binary strings. Since 1996 differential evolution has been used in a wide spectrum of scientific problems such as physics [8], computer science [9], shape optimization [10-13], signal processing [14], control science [15], traffic control [16], manufacturing [17], management [18] and even economics [19-20].

Many researchers tried to improve the performance of the basic differential evolution algorithm in several specific applications. There are three major control parameters in the DE algorithm: the mutation scale factor F, the crossover constant Cr, and the population size Np. Gamperle et al. calculated different parameter settings for DE on the Sphere, Rosenbrock's, and Rastrigin's functions. They concluded that the capability of finding global optimized point and the convergence of DE are strongly dependent on the choice of control parameters Np, F, and Cr [21]. Therefore, some researchers started to develop several techniques to adaptively find optimal set of control parameters of DE in different applications [21-24]. Qin et al. proposed a self-adaptive DE (SaDE) algorithm, in which both trial vector generation strategies and their associated control parameter values are gradually self-adapted by learning from their previous experiences in generating promising solutions [25]. Ali and Törn [26] proposed that the mutation scaling factor, F should be diversified at early stages and intensified at latter stages. Abbass introduced self-adaptive multi-objective DE by encoding the value of Cr into each individual and simultaneously evolving it

[1] Tel: 1 780.492.0802

with other search variables [27]. Zaharie derived a closed form relation for lower limit of mutation scaling factor [28]. Mallipeddi and Suganthan studied the effect of population size on the quality of solutions using a set of five benchmark problems [29].
The most important challenge in some optimization problems is CPU time. The evaluation of fitness function in some engineering applications is very time consuming. Engineering designers employ optimization algorithms together with numerical simulation software to design better and cheaper products. Using finite element or finite volume based simulation software, they solve governing partial differential equations thousands of times to find optimal design. The performance and convergence speed of optimization algorithm have the most important effect on CPU time in this specific applications.
Although adaptive setting of differential evolution control parameters can improve the performance of DE algorithm in some specific applications, this work tries to look at DE algorithm from different perspective and improve the performance of different versions of differential evolution and even adaptive DE algorithms. In this study, the original differential evolution algorithm has been modified in order to increase the convergence speed of the optimization algorithm. The performance characteristics of the modified differential evolution algorithm have been compared with those of the original differential evolution algorithm. These modifications have been applied to different versions of differential evolution and adaptive DE algorithms in order to investigation the convergence speed of these modified methods. Finally, the proposed modifications have been tested on several optimization problems to evaluate the effect of these modifications on the convergence speed of the algorithm in different applications.

2. DIFFERENTIAL EVOLUTION

Differential Evolution (DE) is a vector population based stochastic optimization method which has been introduced in 1995 by Storn and Price. Like genetic algorithm (GA), this method is able to optimize objective functions which are function of discrete variables [30-31].
An unconstrained optimization problem can be stated as follows [32]:

$$\text{Find} \quad \vec{X} = \left(x_1, x_2, \ldots, x_n\right) \quad \text{which minimizes} \quad f(\vec{X}) \tag{1}$$

where $\vec{X}$ is an n-dimensional vector called design vector and $f(\vec{X})$ is the objective function.
Differential evolution can be briefly explained as follows:

2.1. The Population

Differential evolution is a population based optimization method. Assume the population contains *Np* individuals. $\vec{X}_{i,g}$ is the i^{th} individual of g^{th} generation of the population. The first population is selected randomly in differential evolution. Figure 1 shows the first random population in a two dimensional problem [33].

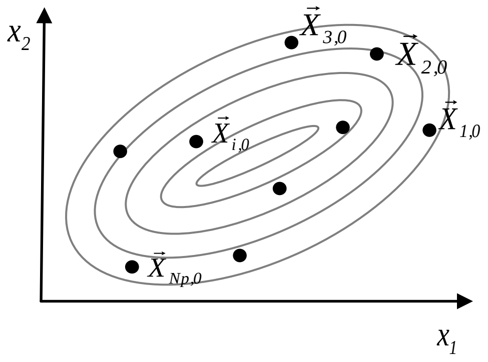

Figure.1 The first random population in a two dimensional problem

2.2. The Mutation

There are several techniques for mutation of individuals in differential evolution. In general, the mutant individual can be defined as follows [33]:

$$\vec{V}_{i,g} = \vec{Y}_{i,g} + F.\frac{1}{N}\sum_{n-0}^{N-1}\left(\vec{X}_{r(2n+1),g} - \vec{X}_{r(2n+2),g}\right) \tag{2}$$

where $\vec{Y}_{i,g}$ is the base vector and *F* is a constant parameter called mutation scale factor and subscript *r* shows that the individual is selected randomly in the population. Based on this general equation, there are four mutation techniques which are very popular in the literature:

DE/rand/1/bin: $$\vec{V}_{i,g} = \vec{X}_{r0,g} + F.\left(\vec{X}_{r1,g} - \vec{X}_{r2,g}\right) \tag{3}$$

DE/best/1/bin: $$\vec{V}_{i,g} = \vec{X}_{best,g} + F.\left(\vec{X}_{r1,g} - \vec{X}_{r2,g}\right) \tag{4}$$

DE/current-to-best/1/bin: $$\vec{V}_{i,g} = \vec{X}_{i,g} + F.\left(\vec{X}_{best,g} - \vec{X}_{i,g}\right) + F.\left(\vec{X}_{r1,g} - \vec{X}_{r2,g}\right) \tag{5}$$

DE/best/2/bin: $$\vec{V}_{i,g} = \vec{X}_{best,g} + F.\left(\vec{X}_{r1,g} - \vec{X}_{r2,g} + \vec{X}_{r3,g} - \vec{X}_{r4,g}\right) \tag{6}$$

where $\vec{X}_{best,g}$ is the individual which has the best fitness in the population.

2.3. The Crossover

The most common crossover in differential evolution is uniform crossover which can be defined as follows [33]:

$$\vec{U}_{i,g} = u_{j,i,g} = \left\{ \begin{matrix} v_{j,i,g} & if & r_j \le Cr \ or \ j = j_{rand} \\ x_{j,i,g} & if & r_j > Cr \end{matrix} \right\} \quad j = 1..n \tag{7}$$

where r_i is a uniformly distributed random variable ($0 \le r_i < 1$) and j_{rand} is a random index ($1 \le j_{rand} \le$ n) and *Cr* is a constant parameter called crossover constant.

2.4. The Selection

The final step in DE algorithm is the selection of the better individual for the minimization of the objective function $f(\vec{X})$. This process can be defined as follows [33]:

$$\vec{X}_{i,g+1} = \left\{ \begin{matrix} \vec{U}_{i,g} & if & f(\vec{U}_{i,g}) \le f(\vec{X}_{i,g}) \\ X_{i,g} & if & f(\vec{U}_{i,g}) > f(\vec{X}_{i,g}) \end{matrix} \right\} \tag{8}$$

The selection process involves a simple replacement of the original individual with the obtained new individual if it has a better fitness.

There are three control parameters in the DE algorithm: the mutation scale factor *F*, the crossover constant *Cr*, and the population size *Np*. Storn and Price suggested the following values for these control variables [33]:

$$\begin{aligned} &F \in [0.5, 1.0] \\ &Cr \in [0.8, 1.0] \\ &Np \approx 10.n \end{aligned} \tag{9}$$

2.5. Differential Evolution Algorithm

Table 1 shows the differential evolution algorithm (DE/rand/1/bin).

Table 1. Differential evolution algorithm (DE/rand/1/bin)

```
Step 1: Select Np individuals $\vec{X}_{i,g}$ randomly.
Step 2: For i = 1 to Np let $f_i = f(\vec{X}_{i,g})$
Step 3: While (convergence criterion not yet met) do steps 4 to 10
Step 4: For i = 1 to Np do steps 5 to 10
Step 5: Select three different random indexes $r_0$, $r_1$ and $r_2$ between 1 to Np ($i \ne r_0 \ne r_1 \ne r_2$)
Step 6: Let $\vec{V}_{i,g} = \vec{X}_{r0,g} + F.\left(\vec{X}_{r1,g} - \vec{X}_{r2,g}\right)$
Step 7: For j = 1 to n do steps 8 to 9
Step 8: Select randomly $r_j$ variable ($0 \le r_j < 1$) and $j_{rand}$ index ($1 \le j_{rand} \le n$)
Step 9: If $r_j \le Cr$ or $j = j_{rand}$ then $u_{j,i,g} = v_{j,i,g}$ else $u_{j,i,g} = x_{j,i,g}$
Step 10: If $f(\vec{U}_{i,g}) \le f_i$ then $\vec{X}_{i,g+1} = \vec{U}_{i,g}$ ; $f_i = f(\vec{U}_{i,g})$ else $\vec{X}_{i,g+1} = \vec{X}_{i,g}$
```

3. IMPROVED DE ALGORITHM

In many scientific applications such as shape optimization, the evaluation of fitness function is very time consuming. This condition makes the process of optimization very time consuming. Looking at differential evolution algorithm (DE/rand/1/bin), at each generation every individual evolves based on three other random selected individuals, one as the based individual and the difference of the other two as the mutating vector. At the end of the process, the evolved individual, $\vec{U}_{i,g}$ competes with the original individual, $\vec{X}_{i,g}$. In this competition if the fitness function of the evolved individual is better than the fitness function of the original individual, the original individual is replaced by the evolved individual. In fact, differential evolution algorithm finds the optimal point through an organized parallel interacting try and error process. Some of these tries are successful and increase the average fitness of the population and some of them are unsuccessful and are not able to do any change in the population. In every try, the fitness of the evolved individual should be evaluated which can be very time consuming in some scientific applications. If there was a way to decrease the number of unsuccessful tries in DE algorithm, the computational time of the optimization process could be decreased and consequently the performance of differential evolution would be increased. In the selection process of differential evolution, the evolved individual $\vec{U}_{i,g}$ competes with the original individual $\vec{X}_{i,g}$. In this process for the original individuals which have worse fitness compared to the average fitness of the population, the probability of replacement of them by evolved individuals is more than the original individuals which have better fitness compared to the average fitness of the population. Based on this interpretation, in this work a modified differential evolution algorithm is proposed. In the modified version of differential evolution algorithm, in each generation, all individuals of the population are sorted based on their fitness. As evolved individual is evaluated based on three random selected individuals, this modification does not change the differential evolution algorithm. After that, to increase the number of successful tries in selection process of differential evolution algorithm, only half the of population which has worse fitness is evolved in each generation. This modification can increase the probability of the selection of evolved individuals and consequently can increase the performance of differential evolution algorithm. Table 2 shows the modified differential evolution algorithm (Modified DE/rand/1/bin).

Table 2. Modified differential evolution algorithm (Modified DE/rand/1/bin)

```
Step 1: Select Np individuals $\vec{X}_{i,g}$ randomly.
Step 2: For i = 1 to Np let $f_i = f(\vec{X}_{i,g})$
Step 3: If Np is even then let m = Np/2 else m = (Np+1)/2
Step 4: While (convergence criterion not yet met) do steps 5 to 15
Step 5: For i = 1 to Np do steps 6, 7
Step 6: For j = i+1 to Np do steps 7
Step 7: If $f_j > f_i$ then swap $\vec{X}_{i,g}, \vec{X}_{j,g}$ and swap $f_j, f_i$
Step 8: For i = 1 to m do steps 9 to 14
Step 9: Select three different random indexes $r_0$, $r_1$ and $r_2$ between 1 to Np ( $i \neq r_0 \neq r_1 \neq r_2$ )
Step 10: Let $\vec{V}_{i,g} = \vec{X}_{r0,g} + F.\left(\vec{X}_{r1,g} - \vec{X}_{r2,g}\right)$
Step 11: For j = 1 to n do steps 12 to 13
Step 12: Select randomly $r_j$ variable ($0 \le r_j < 1$) and $j_{rand}$ index ($1 \le j_{rand} \le n$)
Step 13: If $r_j \le Cr$ or $j=j_{rand}$ then $u_{j,i,g} = v_{j,i,g}$ else $u_{j,i,g} = x_{j,i,g}$
Step 14: If $f(\vec{U}_{i,g}) \le f_i$ then $\vec{X}_{i,g+1} = \vec{U}_{i,g}$ ; $f_i = f(\vec{U}_{i,g})$ else $\vec{X}_{i,g+1} = \vec{X}_{i,g}$
Step 15: For i = m+1 to Np let $\vec{X}_{i,g+1} = \vec{X}_{i,g}$
```

It is clear that in the modified algorithm, only half of the population having worse fitness is evolved in each generation. From other point of view, in the modified version of differential evolution algorithm, the individual which has better fitness remains unchanged and survives and the individual with worse fitness has less chance to survive.

4. Results and Discussion

To compare the performance of the modified differential evolution algorithm with the original one, the same optimization problem used by Storn and Price is considered [31, 33].

$$\text{Find } (x, y) \text{ which minimizes } f(x, y) \tag{10}$$

where,

$$\begin{aligned} f(x,y) &= 3(1-x)^2.\exp\left(-\left(x^2+(y+1)^2\right)\right) \\ &\quad -10\left(\frac{x}{5}-x^3-y^5\right).\exp\left(-\left(x^2+y^2\right)\right) \\ &\quad -\frac{1}{3}\exp\left(-\left((x+1)^2+y^2\right)\right) \end{aligned} \tag{11}$$

Figure 2 shows $f(x, y)$ in three-dimensional coordinate system.

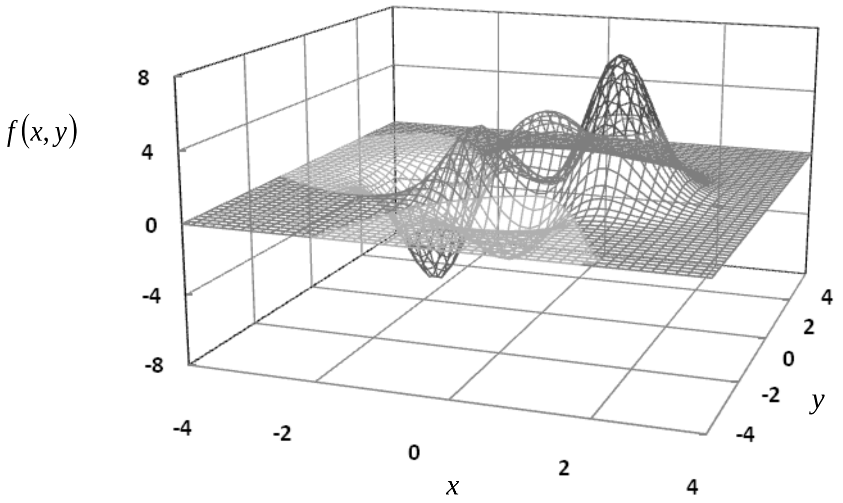

Figure. 2 $f(x, y)$ in three-dimensional coordinate system

Although the evaluation of this objective function is not time consuming, it can show the performance of the optimization algorithm. This function is minimized using both the original differential evolution and the modified differential evolution algorithms with the following conditions:

$$-3 \le x \le 3 \quad and \quad -3 \le y \le 3 \tag{12}$$

The control parameters of differential evolution are assumed to be:

$$F = 0.5 \ ; \ Cr = 0.8 \ and \ \ Np = 10. \tag{13}$$

As mentioned before, the process of evaluation of fitness function is very time consuming in many optimization problems. Therefore the number of fitness functions which have been evaluated in the optimization process can be a very good estimation of CPU time. Figures 3-5 compare the different positions of individuals of the original differential evolution algorithm with those of the modified differential evolution algorithm. Figures 6-8 illustrate the difference vector distribution of both the original and the modified differential evolution algorithms. It can be seen that modified differential evolution algorithm is able to converge and find the optimum point faster compared to the original algorithm.

x

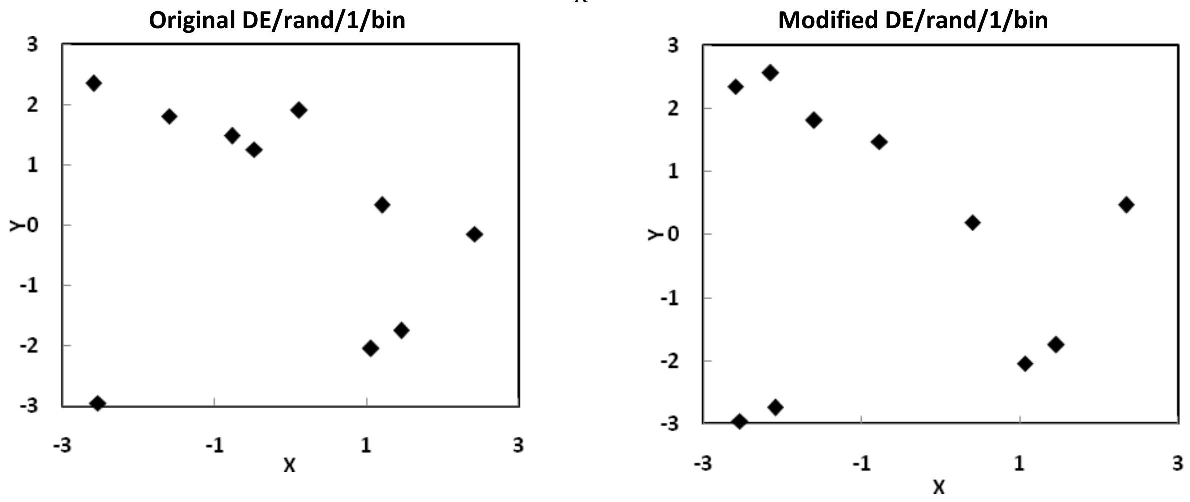

Figure. 3 Position of different individuals after 16 objective function values evaluated

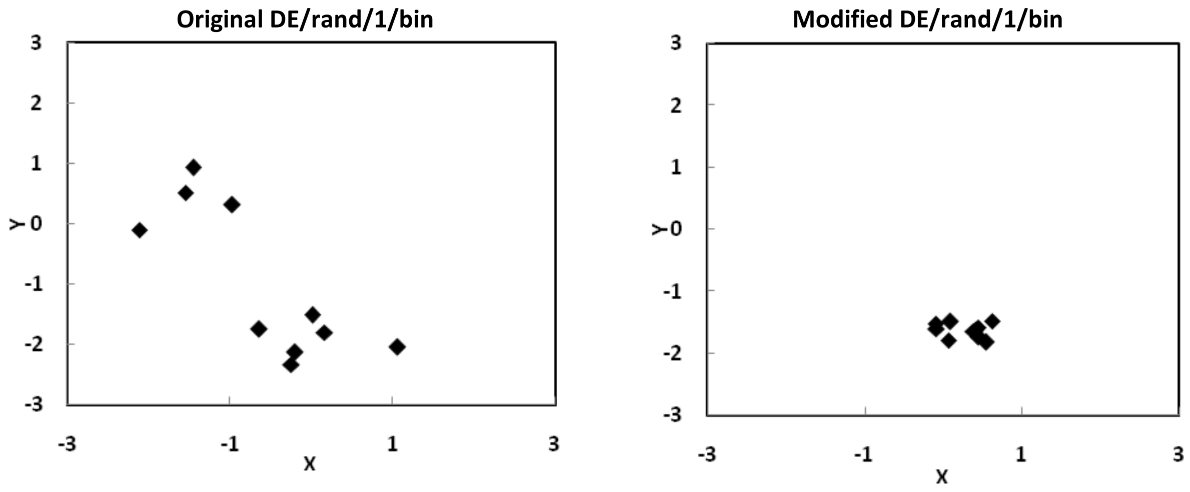

Figure. 4 Position of different individuals after 95 objective function values evaluated

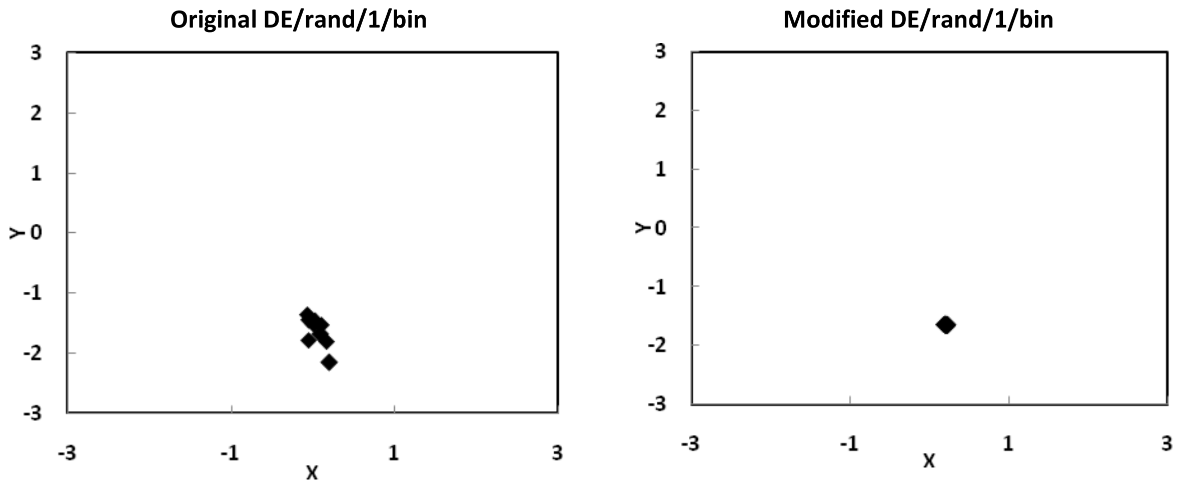

Figure. 5 Position of different individuals after 198 objective function values evaluated

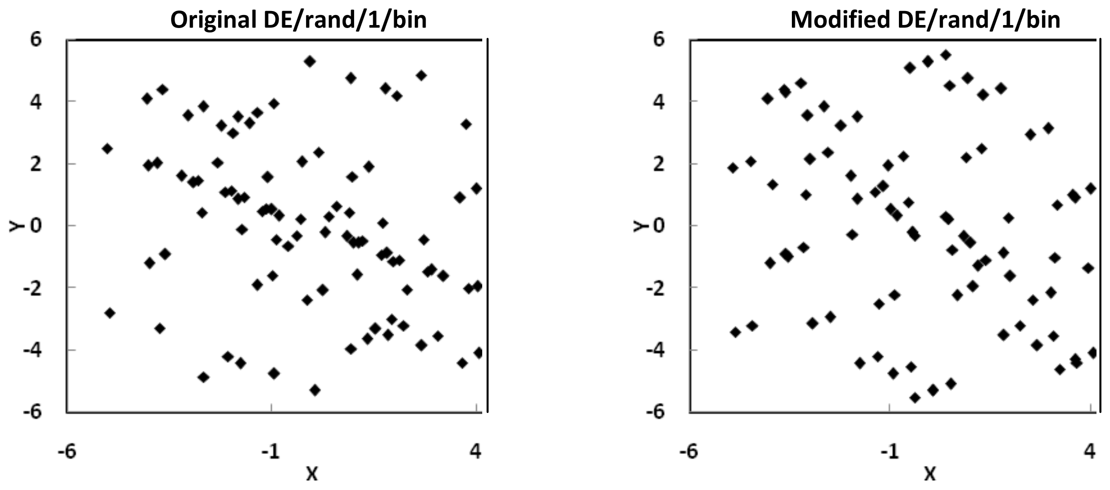

Figure. 6 Difference vector distribution after 16 objective function values evaluated

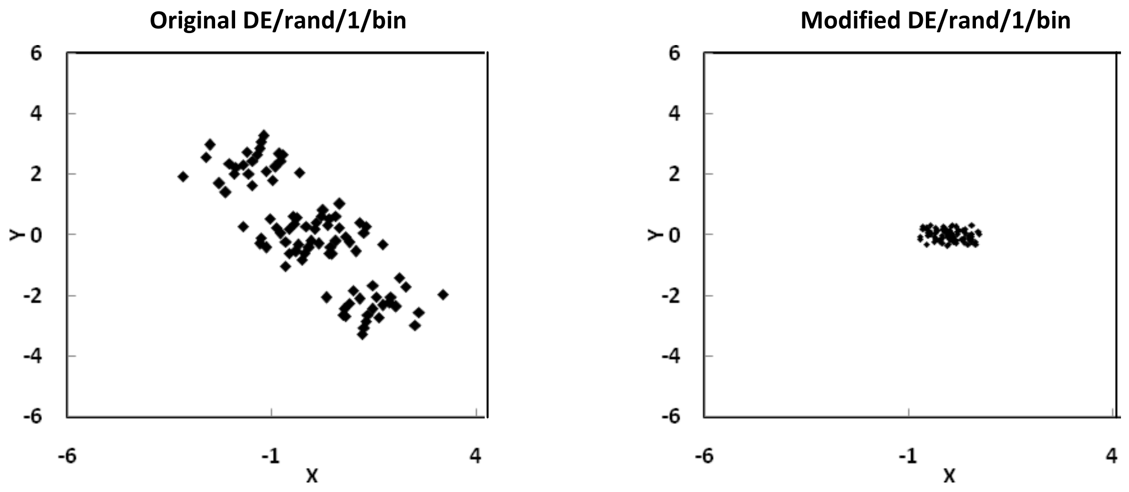

Figure. 7 Difference vector distribution after 95 objective function values evaluated

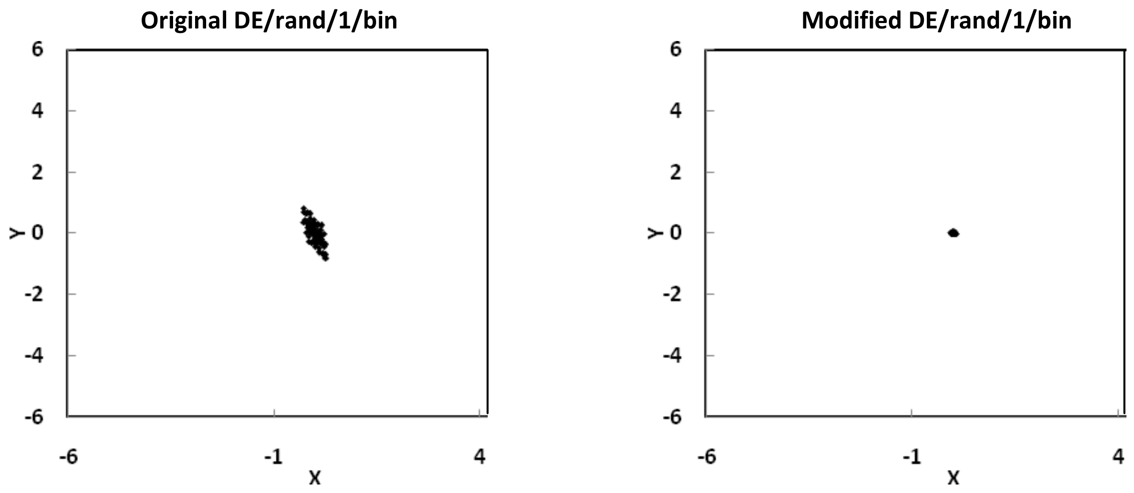

Figure. 8 Difference vector distribution after 198 objective function values evaluated

Figure 9 shows the average fitness in each generation of the original and modified differential evolution algorithms. Similarly, Figure 10 compares the best fitness of individuals of the original differential evolution algorithm with that of the modified algorithm. According to these graphs, in the original differential evolution algorithm the average fitness of the population converges after 286 objective function evaluations and best fitness of the population converges after 231 objective function evaluations (convergence criteria: error less than 1%). While in the modified differential evolution algorithm, the average fitness of the population converges after 162 objective function evaluations and best fitness of the population converges after 132 objective function evaluations. These results show that the modified differential evolution algorithm is about 43% faster than the original differential evolution algorithm in this particular problem (we assumed the major CPU time is related to fitness evaluation). Figure 11 compares the number of successful tries in the selection process of the original differential evolution algorithm with that of the modified algorithm. It is clear that the modified differential evolution algorithm was able to increase the number of successful tries and consequently increase the performance of the optimization algorithm.

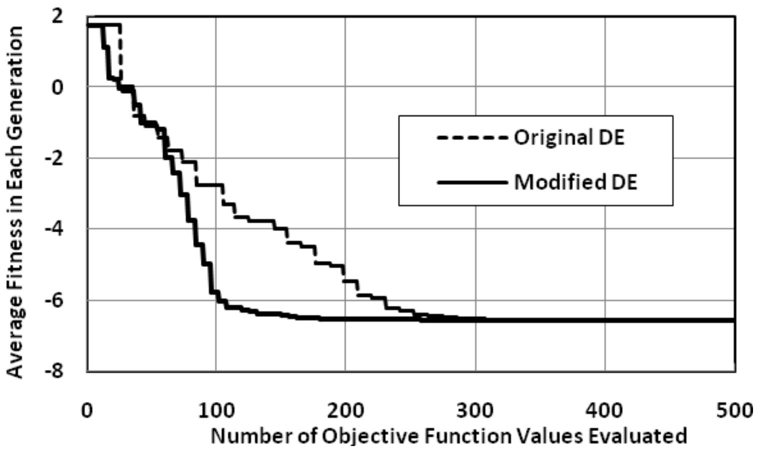

Figure. 9 Average fitness in each generation of the original and modified differential evolution algorithms (DE/rand/1/bin version)

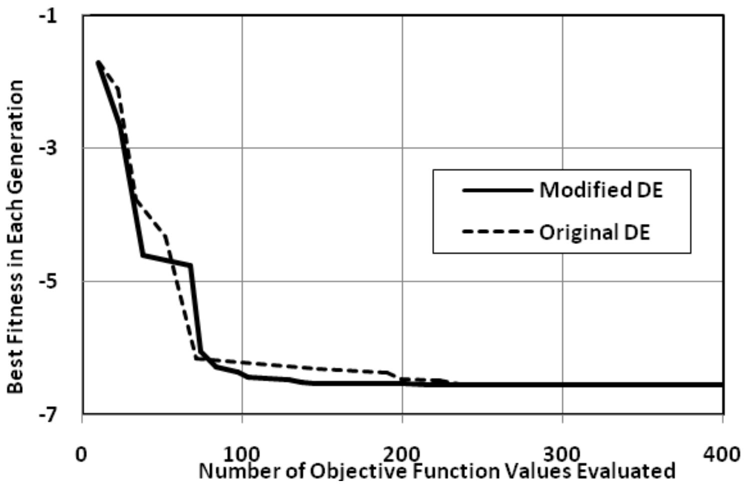

Figure. 10 Best fitness of individuals of the original and the modified differential evolution algorithms (DE/rand/1/bin version)

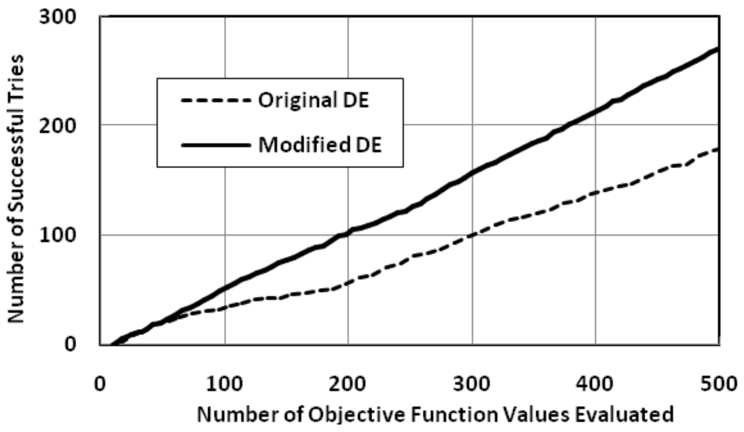

Figure. 11 Number of successful tries in the selection process of the original and modified differential evolution algorithms (DE/rand/1/bin version)

Figures 12-14 show the average fitness in each generation of the different versions of the original and modified differential evolution algorithms. According to the results obtained, these modifications were able to improve the performance of the different versions of differential evolution algorithm.

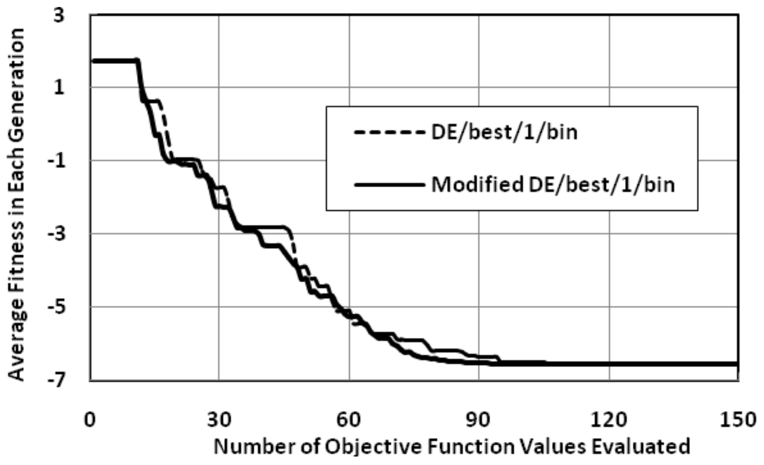

Figure. 12 Average fitness in each generation of the original and modified differential evolution algorithms (DE/best/1/bin version)

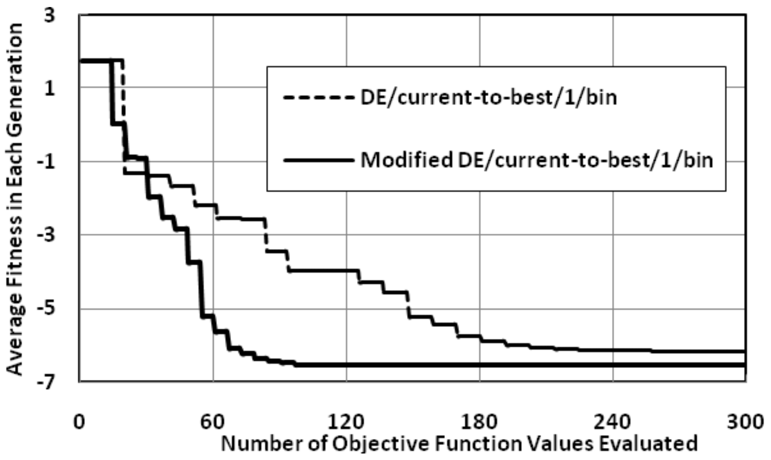

Figure. 13 Average fitness in each generation of the original and modified differential evolution algorithms (DE/current-to-best/1/bin version)

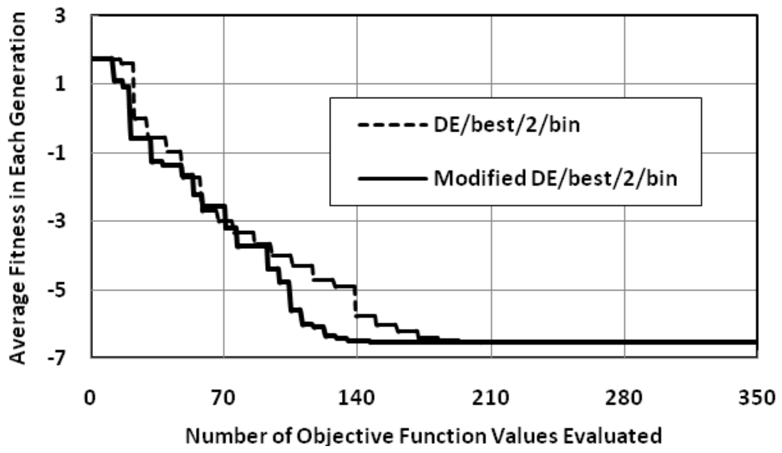

Figure. 14 Average fitness in each generation of the original and modified differential evolution algorithms (DE/best/2/bin version)

There are three control parameters in the DE algorithm: the mutation scale factor *F*, the crossover constant *Cr*, and the population size *NP*. In the standard differential evolution, which was proposed by Price and Storn, these parameters were assumed to be constant. Some researchers have tried to improve the performance of differential evolution algorithm by adaptively tuning these control parameters. Ali and Törn [26] proposed that the mutation scaling factor, *F* should be diversified at early stages and intensified at latter stages. They modified the mutation scale factor by the following equation [26]:

$$F = \left\{ \begin{matrix} \max\left\{ l_{\min}, 1 - \left| \frac{f_{\max}}{f_{\min}} \right| \right\} \ if \ \left| \frac{f_{\max}}{f_{\min}} \right| < 1 \\ \max\left\{ l_{\min}, 1 - \left| \frac{f_{\min}}{f_{\max}} \right| \right\} \ otherwise \end{matrix} \right\} \tag{14}$$

where $l_{\min} = 0.4$ and f_{min} and f_{max} are the minimum and maximum objective function values over the individuals of the population, which are obtained in a generation. Figure 15 compares the average fitness in each generation of the adaptive DE/rand/1/bin with the modified adaptive DE/rand/1/bin algorithms. It is clear that the proposed modifications are even able to improve the performance of the adaptive differential evolution algorithm by 45% (convergence criteria: error less than 1%).

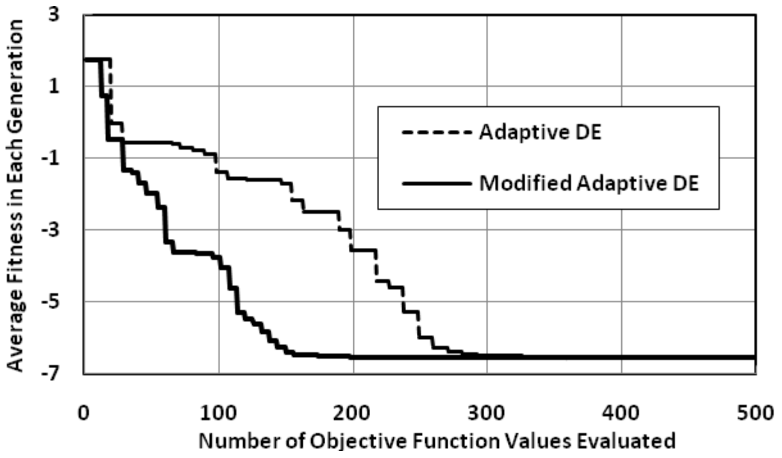

Figure. 15 Average fitness in each generation of the adaptive DE and modified adaptive DE algorithms (Adaptive DE/rand/1/bin by Ali & Törn [26])

To find the performance of the modified differential evolution algorithm in more complex and realistic scientific problems, a shape optimization problem has been solved using original and modified differential evolution algorithms. In this problem, differential evolution algorithm is used for finding a curve joining the points (x_1, y_1) and (x_2, y_2) that yields a surface of revolution of minimum area when revolved about the *x* axis. In other words, we must minimize

$$I = \int_{x_1}^{x_2} 2\pi y\sqrt{1+y'^2}\,dx \tag{15}$$

For this purpose, using 10 design points (x_i are fixed and y_i are variable) the curve joining (x_1, y_1) and (x_2, y_2) is defined through cubic spline curve. Simpson rule of order $\left(\frac{1}{N^4}\right)$ is used for integration. The control parameters of differential evolution are assumed to be:

$$F = 0.5\ ;\ Cr = 0.8\ \ and\ \ Np = 50 \tag{16}$$

For $(x_1, y_1) = (1, 7.5243)$ and $(x_2, y_2) = (4, 2.2552)$ the analytical solution is as follows [34]:

$$y = 2Cosh\left(\frac{x-5}{2}\right) \tag{17}$$

Figure 16 shows the average fitness in each generation of the original and modified differential evolution algorithm. According to this graph, in the original differential evolution algorithm the average fitness of the population converges after 382 seconds (convergence criteria: error less than 1%), while in the modified differential evolution algorithm, the average fitness of the population converges after 237 seconds. This result shows that the modified differential evolution algorithm is about 38% faster than the original one in this particular problem. Figure 17 shows the optimized curve of the revolution which is completely close to the analytical solution. Table 3 shows the average CPU time of different optimization problems which have been solved using the original and modified differential evolution algorithms. The average CPU time is based on 50 different runs for every optimization problem. In all cases, the modified differential evolution algorithm is faster than the original one.

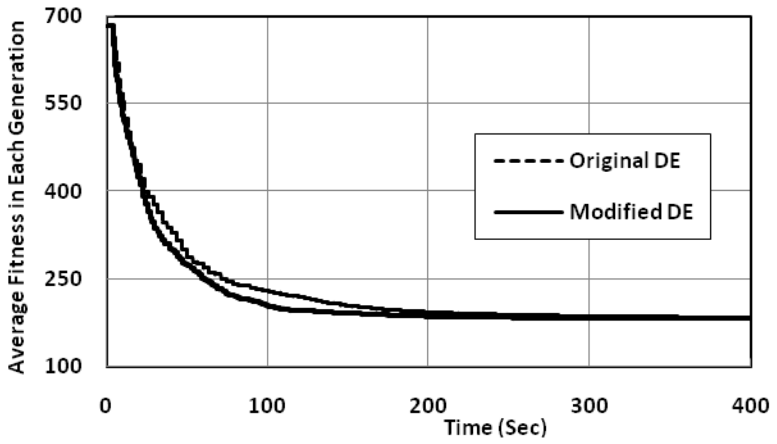

Figure. 16 Average fitness in each generation of the original and modified differential evolution algorithms

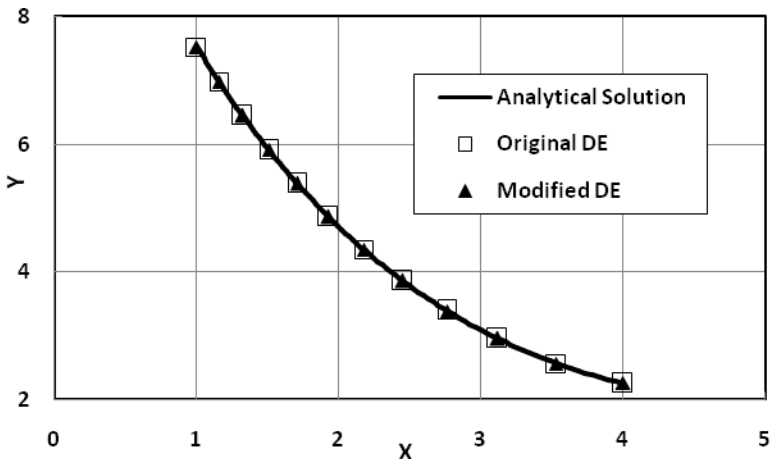

Figure. 17 Optimized curve of revolution

Table 3. Average CPU time of different optimization problems which has been solved using the original and modified differential evolution algorithms (average CPU time of 50 different runs)

	Average Convergence Time (Convergence criteria: error less than 1%)	
Optimization Problem	Original DE/rand/1/bin	Modified DE/rand/1/bin
$I = \int_{x_1}^{x_2} y^2 dx$	**494 Sec**	**403 Sec**
$I = \int_{x_1}^{x_2} \sqrt{1+y'^2}\,dx$	**323 Sec**	**258 Sec**
$I = \int_{x_1}^{x_2} 2\pi y\sqrt{1+y'^2}\,dx$	**382 Sec**	**237 Sec**
$I = \int_{x_1}^{x_2} 2\pi x\sqrt{1+y'^2}\,dx$	**779 Sec**	**592 Sec**
$I = \int_{x_1}^{x_2} \pi y^2\sqrt{1+y'^2}\,dx$	**407 Sec**	**323 Sec**
$I = \int_{x_1}^{x_2} \pi x^2\sqrt{1+y'^2}\,dx$	**502 Sec**	**388 Sec**
$I = \int_{x_1}^{x_2} \frac{\sqrt{1+y'^2}}{y}\,dx$	**420 Sec**	**306 Sec**
$I = \int_{x_1}^{x_2} \frac{\sqrt{1+y'^2}}{x}\,dx$	**413 Sec**	**324 Sec**
$I = \int_{x_1}^{x_2} \frac{\sqrt{1+y'^2}}{\sqrt{2gy}}\,dx$	**356 Sec**	**238 Sec**
$I = \int_{x_1}^{x_2} \frac{\sqrt{1+y'^2}}{\sqrt{2gx}}\,dx$	**404 Sec**	**307 Sec**

5. Conclusion

Since the invention of differential evolution in 1995 many researchers have tried to improve the performance of the basic differential evolution algorithm in several specific applications. The most important challenge in many optimization problems is CPU time. Some researchers started to develop several techniques to adaptively find optimal set of control parameters of DE in order to increase the convergence speed of the optimization algorithm. While differential evolution algorithm finds the optimal point through an organized parallel interacting try and error process, if there was a way to decrease the number of unsuccessful tries in DE algorithm, the performance of differential evolution would be increased. In this study, the original differential evolution algorithm has been modified in order to increase the convergence speed of the optimization algorithm. In the modified version of the differential evolution algorithm, in each generation, all individuals of population are sorted based on their fitness. After that, to increase the number of successful tries in the selection process of the differential evolution algorithm, only half of the population which has worse fitness is evolved in each generation. This modification can increase the probability of selection of evolved individuals and consequently can increase the performance of the differential evolution algorithm. On the other hand, in the modified version of the differential evolution algorithm, the individual which has better fitness remains unchanged and survives while the individual with worse fitness has less chance to survive. The performance characteristics of the modified differential evolution algorithm have been compared with those of the original differential evolution algorithm. Results obtained show that the modified differential evolution algorithm is able to converge and find the optimum point up to 43% faster compared to the original algorithm. These modifications have been applied not only to the different versions of differential evolution algorithm but also to the adaptive differential evolution algorithm. According to the results obtained, these modifications were able to improve the performance of the different versions of differential evolution and even adaptive DE algorithms by 45%. Finally, the proposed modifications have been tested on several optimization problems to evaluate the effect of these modifications on the convergence speed of the algorithm in different applications. In all cases, the modified differential evolution algorithm is 20 to 40 percent faster than the original one. The results obtained completely agree with analytical results.

6. Acknowledgments

The author would like to thank Dr. Carlos F. Lange for his help and supports in this work.

7. References

[1] Gen, M. and Cheng, A.: Genetic Algorithms and Engineering Optimization, John Wiley & Sons, Inc., Toronto (2000)

[2] Holland, J. H.: Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press (1975)

[3] Goldberg, D. E. and Samtani, M. P.: Engineering Optimization Via Genetic Algorithm, ASCE, 471-482 (1986)

[4] Grefenstette, J. J.: Optimization of Control Parameters for Genetic Algorithm, IEEE Transactions on Systems, Man and Cybernetics, v SMC-16, 1, 122-128 (1986)

[5] Goldberg, D. E. and Chie, H. K.: Genetic algorithm in Pipeline Optimization, Journal of Computing in Civil Engineering, 1, 2, 128-141 (1987)

[6] Storn, R. and K. V. Price: Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces, ICSI, USA, Tech. Rep. TR-95-012 (1995)

[7] Storn, R. and K. Price: Minimizing the real functions of the ICEC'96 contest by differential evolution, Proceedings of the IEEE Conference on Evolutionary Computation, 842-844 (1996)

[8] Sabat, S. L. and Kumar, K. S. and Rangababu, P.: Differential evolution algorithm for motion estimation, Proceedings of Multi-Disciplinary Trends in Artificial Intelligence - 5th International Workshop, 7080 LNAI, 309-316, MIWAI (2011)

[9] Fabris, F. and Krohling, R. A.: A co-evolutionary differential evolution algorithm for solving min-max optimization problems implemented on GPU using C-CUDA, Expert Systems with Applications, 39, 12, 10324-10333 (2012)

[10] Madavan, N. K.: On improving efficiency of differential evolution for aerodynamic shape optimization applications, Collection of Technical Papers - 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 6, 3590-3609 (2004)

[11] Song, L. and Feng, Z. and Li, J.: Shape optimization of turbine stage using adaptive range differential evolution and three-dimensional Navier-stokes solver, Proceedings of the ASME Turbo Expo, 6, 1033-1040 (2005)

[12] Chi, Y. C. and Fang, J. and Rao, D. L. and Cai, G. B.: Self-adaptive center-mutation differential evolution and its application to shape optimization design of a turbine blade, Hangkong Dongli Xuebao/Journal of Aerospace Power, 25, 8, 1849-1854 (2010)

[13] Shamekhi, A.: Shape Optimization of U-Bend Passage Using Differential Evolution and Finite Element Method, 9-12 May, Canmore, Canada (2012)

[14] Biswal, B. and Biswal, M. K. and Mishra, S.: Time frequency analysis and power signal disturbance classification using support vector machine and differential evolution algorithm, International Journal of Knowledge-Based and Intelligent Engineering Systems, 16, 3, 199-214 (2012)

[15] Lu, H. C. and Chang, M. H. and Tsai, C. H.: Parameter estimation of fuzzy neural network controller based on a modified differential evolution, Neurocomputing, 89, 178-192 (2012)

[16] Zhang, M. and Zhao, S. and Lv, J. and Qian, Y.: Multi-phase urban traffic signal real-time control with multi-objective discrete differential evolution, Proceedings of International Conference on Electronic Computer Technology, ICECT 2009, 296-300 (2009)

[17] Noktehdan, A. and Karimi, B. and Husseinzadeh Kashani, A.: A differential evolution algorithm for the manufacturing cell formation problem using group based operators, Expert Systems with Applications, 37, 7, 4822-4829 (2010)

[18] Wang, X. and Xu, G.: Hybrid differential evolution algorithm for traveling salesman problem, International Conference on Advanced in Control Engineering and Information Science, CEIS (2011)

[19] Zhang, R. and Song, S. and Wu, C.: A hybrid differential evolution algorithm for job shop scheduling problems with expected total tardiness criterion, Applied Soft Computing Journal, Mathematical Problems in Engineering, 2011 (2011)

[20] Zhang, R.: A differential evolution algorithm for job shop scheduling problems involving due date determination decisions, International Journal of Digital Content Technology and its Applications, 5, 7, 388-396 (2011)

[21] Gamperle, R. and S. D. Muller and A. Koumoutsakos: Parameter study for differential evolution, In proc. WSEAS NNA-FSFS-EC, Interlaken, Switzerland, 293–298 (2002)

[22] Beyer, H.-G. and K. Deb: On self-adapting features in real-parameter evolutionary algorithms, IEEE Trans. Evol. Comput., 5, 3, 250–270 (2001)

[23] Brest, J. and S. Greiner, B. Boskovic, M. Mernik, and V. Zumer: Selfadapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput., 10, 6, 646–657 (2006)

[24] Liu, J. and and J. Lampinen: A fuzzy adaptive differential evolution algorithm, Soft Comput. A Fusion Founda. Methodol. Applicat., 9, 6, 448–462 (2005)

[25] Qin, A. K. and V. L. Huang and P. N. Suganthan: Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput., 13, 2, 398–417 (2009)

[26] Ali, M. M. and A. Törn: Population set based global optimization algorithms: Some modifications and numerical studies, Comput. Oper. Res., 31, 10, 1703–1725 (2004)

[27] Abbass, H.: The self-adaptive Pareto differential evolution algorithm, in Proc. Congr. Evol. Comput., 1, 831–836, May (2002)

[28] Zaharie, D.: Critical values for the control parameters of differential evolution algorithms, in Proc. 8th Int. Mendel Conf. Soft Comput., 62–67 (2002)

[29] Mallipeddi, R. and P. N. Suganthan: Empirical study on the effect of population size on differential evolution algorithm, in Proc. IEEE Congr. Evol. Comput., 3663–3670, June (2008)

[30] Price, K. and R. Storn: Genetic algorithms: Differential evolution, Dr. Dobb's Journal of Software Tools for Professional Programmer, 22, 4, 18, Apr (1997)

[31] Price, K. V. and R. M. Storn and J. A. Lampinen: Differential Evolution A Practical Approach to Global Optimization, Springer-Verlag Berlin Heidelberg (2005)

[32] Rao, S. S.: Engineering Optimization Theory and Practice, 4th Edition, John Wiley & Sons, Inc., Hoboken, New Jersey (2009)

[33] Storn, R.: Differential Evolution Research – Trends and Open Questions, Advances in Differential Evolution, Springer-Verlag Berlin Heidelberg, pp. 1-31 (2008)

[34] Simmons, G.F. and J. S. Robertson: Differential Equations with Applications and Historical Notes, 2nd Edition, McGraw-Hill, Inc. (1991)