# A Differential Evolution Algorithm with a Variable Neighborhood Search for Constrained Function Optimization

M. Fatih Tasgetiren[1,*], P.N. Suganthan[2], Sel Ozcan[3], and Damla Kizilay[4]

[1] Industrial Engineering Department, Yasar University,
Selcuk Yasar Campus, Izmir, Turkey
fatih.tasgetiren@yasar.edu.tr
[2] School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore
epnsugan@ntu.edu.sg
[3] Industrial Engineering Department, Yasar University,
Selcuk Yasar Campus, Izmir, Turkey
sel.ozcan@yasar.edu.tr
[4] Industrial Engineering Department, Yasar University,
Selcuk Yasar Campus, Izmir, Turkey
damla.kizilay@yasar.edu.tr

**Abstract.** In this paper, a differential evolution algorithm based on a variable neighborhood search algorithm (DE_VNS) is proposed in order to solve the constrained real-parameter optimization problems. The performance of DE algorithm depends on the mutation strategies, crossover operators and control parameters. As a result, a DE_VNS algorithm that can employ multiple mutation operators in its VNS loops is proposed in order to further enhance the solution quality. We also present an idea of injecting some good dimensional values to the trial individual through the injection procedure. In addition, we also present a diversification procedure that is based on the inversion of the target individuals and injection of some good dimensional values from promising areas in the population by tournament selection. The computational results show that the simple DE_VNS algorithm was very competitive to some of the best performing algorithms from the literature.

## 1 Introduction

In general, a constrained optimization problem focuses on optimizing a vector $\vec{x}$ in order to minimize the following problem:

$$\min f(\vec{x}) \quad \vec{x} = (x_1, x_2, .., x_D) \in \Re^D \tag{1}$$

where $x \in F \subseteq S$. On the *search space* $S \subseteq \Re^D$, the objective function of a vector $\vec{x}$ is described as $f(\vec{x})$ and the *feasible region* is given on the set $F \subseteq S$. Usually,

* Corresponding author.



I. Fister and I. Fister Jr. (eds.), *Adaptation and Hybridization in Computational Intelligence*,
Adaptation, Learning, and Optimization 18, DOI: 10.1007/978-3-319-14400-9_8

$S$ is described as a $D$-dimensional space in $\Re^D$ and its domains of the decision variables are described by their search ranges as follows:

$$x_k^{\min} x_k \le x_k^{\max} \quad 1 \le k \le D \tag{2}$$

By using a set of $m$ additional constraints $(m \ge 0)$, the feasible region $F$ is described as follows:

$$g_i(\vec{x}) \le 0 \text{ , for } i = 1,..,p \text{ and} \tag{3}$$

$$h_j(\vec{x}) = 0 \text{ , for } j = p+1,..,m \text{ .} \tag{4}$$

In general, the equality constraints can be transformed into inequality form and can be combined with other inequality constraints as

$$\begin{aligned} G_i(\vec{x}) &= \max\{g_i(\vec{x}),0\} & i &= 1,...p \\ H_i(\vec{x}) &= \max\{|h_i(\vec{x})| - \delta,0\} & i &= p+1,...,m \\ \upsilon(\vec{x}) &= \frac{\sum_{i=1}^{p} G_i(\vec{x}) + \sum_{i=p+1}^{m} H_i(\vec{x})}{m} \end{aligned} \tag{5}$$

where $\upsilon(\vec{x})$ is the average violation of $m$ constraints. In addition, $\delta$ is a tolerance value for the equality constraints, which is in general taken as $\delta = 0.0001$ in the literature.

Differential evolution (DE) is one of the most sophisticated evolutionary algorithms, which is proposed by Storn and Price [38,39]. Moreover, DE is a population-based, stochastic global optimizer. So far, DE has been extensively employed to solve many real-parameter optimization problems. The surveys of DE can be found in Corne et al. [8], Lampinen [18], Babu and Onwubolu [4], Das and Suganthan [43], and Price et al. [27]. Recently, Elsayed et al. [32] proposed an algorithm framework with multiple search operators, where the performance of evolutionary algorithms can be enhanced through employing a self-adaptative strategy and multiple search operators. Furthermore, in Qin et al. [1], a self-adaptive DE variant, called saDE, was proposed, where parameter values were updated gradually with a learning mechanism. Similarly, in the study of Zhang et al. [48], different than traditional DE, an adaptive DE where control parameters were updated adaptively with optional external memory (JADE) was introduced. A composite DE (coDE) was presented in Yong et al. [47] through the use of efficient trial individual obtaining strategies and control parameters. Mallipedi et al. [31] introduced an ensemble idea in DE by considering multiple mutation strategies and control parameters so called EPSDE. Moreover, this ensemble idea was extended to constrained optimization problems to handle the constraints in Mallipedi et al. [30]. Tasgetiren et al. [25] presented an ensemble DE by assigning each individual to a different mutation strategy or a variable parameter search (VPS). On the other hand, Zamuda and Brest

[3,13] introduced a variant of DE algorithm with a population reduction methodology. Elsayed et al. [34] recently developed a DE algorithm so called SAS-DE, in which an improvement method was used and was adaptively utilized for testing the CEC2010 constrained optimization benchmark instances. In the study of Gong et al. [46] DE with a ranking-based mutation operator was presented and various engineering problems were tested. Mohamed and Sabry [2] introduced a modified differential evolution algorithm (COMDE) including a new mutation and a dynamic non-linear increased crossover probability. On the other hand, Long et al. [45] presented a new hybrid DE-modified with augmented Lagrangian multiplier method in order for solving constrained optimization problems. Furthermore, various DE algorithms designed to solve constrained optimization problems can also be obtained in [5, 6, 15, 16, 17, 19, 21, 35, and 37] and a comprehensive survey of DE algorithms on constrained optimization can also be obtained in [7].

Having obtained successful results in the vpsDE algorithm in [23] as well as the ensemble concept in [24, 28, 29], this paper presents a DE_VNS algorithm to solve the benchmark instances in CEC2006 [20].

This paper is organized as follows. In Section 2, a basic DE algorithm is explained whereas Section 3 outlines the proposed DE_VNS algorithm. Constraint handling methods employed are given in Section 4. Furthermore, computational results are given in Section 5 and finally, Section 6 summarizes the conclusions.

## 2 Differential Evolution Algorithm

Due to the existence of several mutation strategies in the basic DE algorithms, for a general description, we employ the $DE/rand/1/bin$ variant of Storn and Price [38,39]. In the basic DE algorithm, the initial target population is established by $NP$ number of individuals. A target individual in the population contains a $D$-dimensional vector with parameter values. These parameter values are initially generated uniformly between predetermined search bounds between $x_{ij}^{\min}$ and $x_{ij}^{\max}$ as follows:

$$x_{ij}^{t} = x_{ij}^{\min} + \left(x_{ij}^{\max} - x_{ij}^{\min}\right) \times r \tag{6}$$

where $x_{ij}^{t}$ is the target individual at generation $t$. $r$ is a uniform random number generated within the range U[0,1].

Mutant population is established as follows: Two individuals are picked up from target population. Then, the weighted difference of them is added to a third individual in the target population. This can be achieved as follows:

$$v_{ij}^{t} = x_{aj}^{t-1} + F \times \left(x_{bj}^{t-1} - x_{cj}^{t-1}\right) \tag{7}$$

where $a$, $b$, and $c$ are three randomly chosen individuals from the target population such a way that $(a \neq b \neq c \neq i \in (1,..,NP))$ and $(j = 1,2,..,D)$. $F > 0$ is a mutation scale factor affecting the differential variation between two individuals.

In the next step, the trial individual can be obtained by making a uniform crossover between the target and mutant individuals as follows:

$$u_{ij}^{t} = \begin{cases} v_{ij}^{t} & if \quad r_{ij}^{t} \leq CR \quad or \quad j = D_j \\ x_{ij}^{t-1} & otherwise \end{cases} \tag{8}$$

where the $D_j$ denotes a randomly selected dimension. It guarantees that at least one parameter of each trial individual comes from the mutant individual. $CR$ is a crossover rate within the range $[0,1]$, and $r_{ij}^{t}$ is a random number generated from U[0,1]. In case of any violation of the parameter values of trial individual during the evolution, they are restricted to:

$$u_{ij}^{t} = x_{ij}^{\min} + \left(x_{ij}^{\max} - x_{ij}^{\min}\right) \times r_1 \qquad j = 1,2,..,D \tag{9}$$

Finally, a one-to-one comparison is made to select the better individual in terms of their fitness values as follows:

$$x_i^{t} = \begin{cases} u_i^{t} & if \quad f\left(u_i^{t}\right) \leq f\left(x_i^{t-1}\right) \\ x_i^{t-1} & otherwise \end{cases} \tag{10}$$

## 3 Differential Evolution with Variable Neighborhood Search

To develop a DE with a variable neighborhood search (DE_VNS), we inspire from the VNS algorithm [26]. We take advantage of variable mutation strategies that affect the performance of DE algorithms [29]. We choose the following two mutation strategies to be employed in the VNS loop:

$$M_1 = DE/pbest/1/Bin \tag{11}$$
$$v_{ij}^{t} = x_{pj}^{t-1} + F \times \left(x_{bj}^{t-1} - x_{cj}^{t-1}\right)$$
$$M_2 = DE/rand/1/Bin \tag{12}$$
$$v_{ij}^{t} = x_{aj}^{t-1} + F \times \left(x_{bj}^{t-1} - x_{cj}^{t-1}\right)$$

In above mutation strategies, $x_p$ is the individual chosen by the tournament selection with size of 2. In other words, two individuals are randomly taken from the population, then the one with the better fitness value is chosen. To generate the trial individual in the DE_VNS algorithm, we define a neighborhood $N_k$ for a temporary individual $\tau$ by a mutation strategy and a crossover operator together as follows:

$$N_k(\tau) = M_k(v), CR(\tau, v) \tag{13}$$

Equation (13) indicates that in order to find a neighborhood of an individual $x_i$ (i.e., implicitly a trial individual $u_i$) , we use a mutation strategy $M_k$ to generate a mutant individual $v$ first, then we recombine mutant individual $v$ with the individual $\tau$ through crossover operator $CR$ , which is a typical binomial crossover operator in equation (8). We use the following two neighborhood structures to be used in the VNS algorithm to generate each trial individual as follows:

$$N_1(\tau) = M_1(v), CR(\tau, v) \text{ where } Cr = 0.9, F = 0.9 \tag{14}$$

$$N_2(\tau) = M_2(v), CR(\tau, v) \text{ where } Cr = U(0,1), F = U(0,1) \tag{15}$$

In other words, in the first neighborhood structure, we employ a very high mutation rate $F$ and a very high crossover rate $Cr$ . However, in the second neighborhood structure, we determine them uniformly in the range [0,1], randomly. With the above definitions and temporary individuals $\tau$ and $x^*$, we develop a VNS algorithm to generate a trial individual as shown in Fig. 1.

Procedure $VNS(x_i)$
$k_{\max} = 2$
$k = 1$
$\tau = x_i$
*do*{
    $x^* = N_k(\tau)$
    *if* $f(x^*) < f(\tau)$
        $\tau = x^*$
        $k = 1$
    *else*
        $k = k + 1$
}*while*$(k \le k_{\max})$
$u_i = \tau$
*return* $u_i$
*Endprocedure*

**Fig. 1.** VNS Algorithm

The performance of VNS algorithms depends on what strategy is used in the first neighborhood. As explained before, the equation (11) is used as the first neighborhood whereas the equation (12) is used as the second neighborhood. Note that as long as the first neighborhood improves the current solution, the neighborhood counter $k$

will be 1 indicating that the first neighborhood will be employed. Otherwise, the neighborhood counter $k$ will be increased to 2 indicating that the second neighborhood will be employed. If the second neighborhood improves the solution, it gets back to the first neighborhood again until the second neighborhood fails. It should be noted that when we compare two solutions in the VNS algorithm, we use the penalized fitness values obtained by the NFT method that will be explained in Section 4.

### 3.1 Initial Population

The initial target population is randomly established as explained before. In other words, NP individuals are established by equation (6). However, we also employ the opposition-based learning algorithm to enrich the initial population. Opposition-based learning (OBL) is proposed by [9]. It is a new method in computational intelligence field and has been applied successfully to further improve various heuristic optimization algorithms [40-42]. OBL is based on an idea that its opposite solution implies a chance to obtain a new solution closer to the global optimal. Inspired from OBL, a generalized OBL (GOBL) is introduced in [10-12]. Suppose that $x$ is the current solution with $x \in [a,b]$. Then its opposite solution is given by:

$$x^* = k(a+b) - x \tag{16}$$

In GOBL, opposite solutions are gathered by dynamically updated interval boundaries in the population as follows:

$$x_{ij}^* = kU(a_j, b_j) - x_{ij} \tag{17}$$

$$a_j = \min(x_{ij}), b_j = \max(x_{ij}) \tag{18}$$

$$x_{ij}^* = U(a_j, b_j) \quad if \quad x_{ij}^* < x_j^{\min} \quad or \quad x_{ij}^* > x_j^{\max}$$

$$i = 1,..,NP, \quad j = 1,..,D, \quad k = U[0,1] \tag{19}$$

After establishing and evaluating the target population, the above GOBL algorithm is also used to obtain the opposite target individual. The better one is retained in the target population.

### 3.2 Generation of Trial Population

Trial individuals are generated through the VNS algorithm explained before. Once each individual is obtained from the VNS algorithm, we further apply an injection procedure to trial individuals to diversify it and escape from the local minima. In the injection procedure, we select an individual from the target population by tournament selection with size of 2. Then depending on the injection probability, we inject some good dimensional values to the trial individuals in such a way that a uniform random number $r$ is less than the injection probability $iP$, that dimension is taken from individual $x_a$, which is determined by the tournament selection procedure. Otherwise, the dimension of the trial individual is retained. The injection procedure is given in Fig. 2.

```
for  i = 1  to  NP
  for  j = 1  to  D
    if  (r < iP)  then
      x_aj = TournamentSelect()
      u_ij = x_aj
    else
      u_ij = u_ij
  endfor
endfor
```

**Fig. 2.** Injection Procedure

### 3.3 Selection

When the selection for the next generation is carried out, we employ the EC and SF constraint handling methods that will be summarized in Section 4 as follows: For each individual in the trial population, we check the $\varepsilon(t)$ level. If the constraint violation is less than $\varepsilon(t)$ level, we treat the trial individual as a feasible solution. Then we employ the SF method whether or not the trial individual will survive to be in the next generation. In addition, we simply use the SF method to update the best so far solution in the population.

### 3.4 Diversification

In order to further diversify the target population, we propose a diversification mechanism based on the inversion of the dimensional values of the target individual and an injection procedure explained above. For a small portion of the target population, following diversification procedure is applied to the randomly selected individuals as shown in Fig. 3.

```
x_aj = RandomlySelect()
x_aj = invert(x_aj)
for  j = 1  to  D
  if  (r < iP)  then
    x_bj = TournamentSelect()
    x_aj = x_bj
  else
    x_aj = x_aj
endfor
```

**Fig. 3.** Diversification Procedure

# 4 Constraint Handling

Evolutionary algorithms can yield infeasible solutions. In this case, the general tendency is to utilize some constraint handling approaches [7, 49]. In this paper, we used the following constraint handling methods:

## 4.1 Superiority of Feasible Solutions (SF)

When using SF [12] for evaluating two solutions such as $x_a$ and $x_b$, $x_a$ is considered to be better than $x_b$ under the following conditions for a minimization problem: (i) solution $x_a$ is feasible and solution $x_b$ is not; (ii) both solutions are feasible but $x_a$ has a smaller objective function value than $x_b$;(iii) both solutions are infeasible, but $x_a$ has a smaller overall constraint violation amount $\upsilon(x)$ that can be calculated by using Eq. (5).

## 4.2 The Adaptive Penalty Function (NFT)

In [36], an adaptive penalty approach is proposed. In adaptive penalty function, the idea of *near feasibility threshold* so called NFT is presented, where the solutions within feasible region and the NFT-neighborhood of the infeasible region are favored. Furthermore, an adaptive part is included in the penalty method to differentiate the gap between the best feasible value and best infeasible value found so far. Then the adaptive penalty function is given as follows:

$$f_p(x) = f(x) + (f_{feas} - f_{all}) \sum_{i=1}^{m} \left( \frac{v_i(x)}{NFT_i} \right)^{\alpha_i} \tag{20}$$

where $f_{all}$ is the unpenalized value of the best solution obtained so far whereas $f_{feas}$ is the value of the best feasible solution yet obtained. As mentioned in [7], the adaptive term may result in zero-or over-penalty. Due to this reason, we only take the dynamic part of the above penalty function with NFT threshold into account as follows:

$$f_p(x) = f(x) + \sum_{i=1}^{p} \left( \frac{G_i(x)}{NFT_i} \right)^{\alpha} + \sum_{j=p+1}^{m} \left( \frac{H_j(x)}{NFT_j} \right)^{\alpha} \tag{21}$$

The basic form of the NFT method is presented as $NFT = \frac{NFT_0}{1 + \lambda * t}$ where $NFT_0$ is the initial value of the NFT method; $\lambda$ and $t$ are user-defined positive value and generation counter, respectively. $\alpha$ is severity parameter. Because of the conversion process of the equality constraints to the inequality constraints by subtracting $\delta$ from the absolute value of the constraint value and $\delta$ is determined beforehand, the $NFT_0$ is chosen as 1e-4.

### 4.3 $\varepsilon$ -Constraint (EC)

The $\varepsilon$-constraint handling method was proposed in [44] in which the constraint relaxation is monitored by $\varepsilon$ parameter. A proper control of the $\varepsilon$ parameter is necessary while obtaining good feasible solutions for problems with equality constraints [44]. The $\varepsilon$ level is updated according to the control generation $t_C$. After $t$ exceeds $t_C$, the $\varepsilon$ level is set to zero to end up with feasible solutions. The main idea lies behind the EC method is that solutions having violations less than $\varepsilon(t)$ are considered to be feasible solutions when making selection for the next generation. The general framework is given as follows:

$$\varepsilon(0) = \upsilon(x_\theta) \tag{22}$$

$$\varepsilon(t) = \begin{cases} \varepsilon(0)\left(1 - \dfrac{t}{t_C}\right)^{cp}, & 0 < t < t_C \\ 0, & t \geq t_C \end{cases}. \tag{23}$$

where $x_\theta$ is the top $\theta$-th individual.

## 5 Computational Results

The DE_VNS algorithm was coded in C++ and run on an Intel P4 1.33 GHz Laptop PC with 256MB memory. The population size is taken as NP=60. The NFT0 is fixed at 0.0001. Injection probability is taken as 0.005 whereas the diversification probability is taken as 0.05. For the EC constraint handling method, following parameters are used as $\theta = 0.25 \times NP$, $t_C = 0.4 * MaxGen$ and $cp = 2$. We carried out 30 replications for each benchmark problem and average, minimum and standard deviation of 30 replications are provided. Note that real numbers are rounded to zero after 10 digits in the standard deviation calculations.

We compare our algorithm to the best performing algorithms from the literature such as MDE [22], ECHT-EP2[30] and SAMO-DE [32]. The computational results are given in Table 1. As seen from Table 1, the DE_VNS algorithm was able to find the optimal solutions with zero standard deviations for 13 out of 22 benchmark problems. The DE_VNS algorithm was slightly better than SAMO-DE because it was able to find 12 optimal solutions with zero standard deviations. The performance of the ECHT-EP2 was slightly better than DE_VNS and SAMO-DE since it was able find 14 optimal solutions with zero standard deviations. The clear winner was the MDE algorithm due to the fact that it was able to find 19 optimal solutions with zero standard deviations. However, DE_VNS, SAMO_DE and ECHT-EP2 algorithms were run for 240000 function evaluations whereas MDE was run for 500000 function evaluations. In 4 benchmark functions, the standard deviation of the DE_VNS algorithm was smaller than both SAMO_DE and ECHT-EP2, respectively. Together with all

algorithms compared, the DE_VNS algorithm was able to find the optimal solutions in all 30 replications. In other words, feasibility rate was 100 %. In summary, the simple DE_VNS algorithm was competitive to the best performing algorithms from the literature.

**Table 1.** Computational Results of DE-VNS, SAMO-DE, MDE, ECHT-EP2 FOR CEC2006 test problems

| Problem | FEs | DE-VNS | SAMO-DE | MDE | ECHT-EP2 |
|---|---|---|---|---|---|
| | | 240,000 | 240,000 | 500,000 | 240,000 |
| g01 | Best | **-15.0000** | **-15.0000** | **-15.0000** | **-15.0000** |
| | Avg | **-15.0000** | **-15.0000** | **-15.0000** | **-15.0000** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g02 | Best | **-0.8036191** | **-0.8036191** | **-0.8036191** | **-0.8036191** |
| | Avg | -0.789822 | -0.79873521 | -0.78616 | **-0.7998220** |
| | Std | 1.87E-02 | 8.80050E-03 | 1.26E-02 | 6.29E-03 |
| g03 | Best | **-1.0005** | **-1.0005** | **-1.0005** | **-1.0005** |
| | Avg | **-1.0005** | **-1.0005** | **-1.0005** | **-1.0005** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g04 | Best | **-30665.5386** | **-30665.5386** | **-30665.539** | **-30665.539** |
| | Avg | **-30665.5386** | **-30665.5386** | **-30665.539** | **-30665.539** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g05 | Best | **5126.497** | **5126.497** | **5126.497** | **5126.497** |
| | Avg | **5126.497** | **5126.497** | **5126.497** | **5126.497** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g06 | Best | **-6961.813875** | **-6961.813875** | **-6961.814** | **-6961.814** |
| | Avg | **-6961.813875** | **-6961.813875** | **-6961.814** | **-6961.814** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g07 | Best | **24.3062** | **24.3062** | **24.3062** | **24.3062** |
| | Avg | **24.306209** | 24.3096 | **24.3062** | 24.3063 |
| | Std | 2.17E-07 | 1.58880E-03 | **0.00E-00** | 3.19E-05 |
| g08 | Best | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** |
| | Avg | **-0.095825** | **-0.095825** | **-0.095825** | **-0.095825** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.0E-00** |
| g09 | Best | **680.630** | **680.630** | **680.630** | **680.630** |
| | Avg | **680.630** | **680.630** | **680.630** | **680.630** |
| | Std | **0.00E-00** | 1.15670E-05 | **0.00E-00** | **0.00E-00** |
| g10 | Best | **7049.24802** | 7049.24810 | **7049.24802** | 7049.2483 |
| | Avg | 7049.24803 | 7059.81345 | **7049.24802** | 7049.2490 |
| | Std | 4.02E-05 | 7.856E-00 | **0.00E-00** | 6.60E -04 |
| g11 | Best | **0.7499** | **0.7499** | **0.7499** | **0.7499** |
| | Avg | **0.7499** | **0.7499** | **0.7499** | **0.7499** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g12 | Best | **-1.0000** | **-1.0000** | **-1.0000** | **-1.0000** |
| | Avg | **-1.0000** | **-1.0000** | **-1.0000** | **-1.0000** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g13 | Best | **0.053942** | **0.053942** | **0.053942** | **0.053942** |
| | Avg | **0.053942** | **0.053942** | **0.053942** | **0.053942** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |

**Table 1.** (*continued*)

| | | | | | |
|---|---|---|---|---|---|
| g14 | Best | **-47.76489** | **-47.76489** | **-47.764887** | **-47.7649** |
| | Avg | **-47.76489** | -47.68115 | -47.764874 | -47.7648 |
| | Std | 4.64E-06 | 4.04300E-02 | 1.400E-05 | 2.72E-05 |
| g15 | Best | **961.71502** | **961.71502** | **961.71502** | **961.71502** |
| | Avg | **961.71502** | **961.71502** | **961.71502** | **961.71502** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g16 | Best | **-1.905155** | **-1.905155** | **-1.905155** | **-1.905155** |
| | Avg | **-1.905155** | **-1.905155** | **-1.905155** | **-1.905155** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |
| g17 | Best | **8853.5397** | **8853.5397** | **8853.5397** | **8853.5397** |
| | Avg | 8877.3107 | **8853.5397** | **8853.5397** | **8853.5397** |
| | Std | 3.94E+01 | 1.15E-05 | **0.00E-00** | 2.13E -08 |
| g18 | Best | **-0.866025** | **-0.866025** | **-0.866025** | **-0.866025** |
| | Avg | -0.834185 | -0.866024 | **-0.866025** | **-0.866025** |
| | Std | 7.12E-02 | 7.04367E-07 | **0.00E-00** | **0.00E-00** |
| g19 | Best | 32.656077 | **32.655593** | 32.655693 | 32.6591 |
| | Avg | 32.685099 | 32.757340 | 33.34125 | 32.6623 |
| | Std | 3.73E-02 | 6.145E-02 | 8.475E-01 | 3.4E -03 |
| g21 | Best | **193.72451** | **193.72451** | **193.72451** | 193.7246 |
| | Avg | 193.72456 | 193.771375 | **193.72451** | 193.7438 |
| | Std | 2.84E-04 | 1.9643E-02 | **0.00E-00** | 1.65E-02 |
| g23 | Best | -400.0527 | -396.165732 | **-400.0551** | -398.9731 |
| | Avg | -372.9920 | -360.817656 | **-400.0551** | -373.2178 |
| | Std | 5.75E+01 | 1.9623E+01 | **0.00E-00** | 3.37E+01 |
| g24 | Best | **-5.508013** | **-5.508013** | **-5.508013** | **-5.508013** |
| | Avg | **-5.508013** | **-5.508013** | **-5.508013** | **-5.508013** |
| | Std | **0.00E-00** | **0.00E-00** | **0.00E-00** | **0.00E-00** |

## 6 Conclusions

In this paper, a differential evolution algorithm with a variable neighborhood search algorithm (DE) is presented to solve the constrained real-parameter optimization problems. The performance of DE depends on the selection of mutation strategies and crossover operators as well as control parameters. For this reason, we developed a DE_VNS algorithm that can employ multiple mutation operators in its VNS loops to further improve the solution quality. We also present an idea of injecting some good dimensional values to the trial individual from population through the injection procedure. In addition, we also present a diversification procedure that is based on the inversion of the target individuals and injection of some good dimensional values from promising areas in the target population by tournament selection. The computational results show that the simple DE_VNS algorithm was very competitive to some of the best performing algorithms from the literature. For the future work, we will develop some DE algorithm taking advantage of the idea of neighborhood change of VNS algorithms for both constrained and unconstrained real parameter optimization problems.

## References

[1] Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential Evolution Algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13, 398–417 (2009)
[2] Mohamed, A.W., Sabry, H.Z.: Constrained optimization based on modified differential evolution algorithm. Information Sciences 194, 171–208 (2012)
[3] Zamuda, A., Brest, J.: Population reduction differential evolution with multiple mutation strategies in real world industry challenges. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M., et al. (eds.) EC 2012 and SIDE 2012. LNCS, vol. 7269, pp. 154–161. Springer, Heidelberg (2012)
[4] Babu, B.V., Onwubolu, G.C. (eds.): New Optimization Techniques in Engineering. STUDFUZZ, vol. 141. Springer, Heidelberg (2004)
[5] Becerra, R.L., Coello, C.C.A.: Cultural Differential Evolution for Constrained Optimization. Comput. Methods Appl. Mech. Engrg. (2005)
[6] Chiou, J.-P., Wang, F.-S.: Hybrid Method of Evolutionary Algorithms for Static and Dynamic Optimization Problems with Applications to a Fed-Batch fermantation Process. Computers and Chemical Engineering 23, 1277–1291 (1999)
[7] Coello, C.C.A.: Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art. Comput. Methods Appl. Mech. Engrg. 191(11-12), 1245–1287 (2002)
[8] Part Two: Differential Evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 77–158. McGraw-Hill (1999)
[9] Tizhoosh, H.R.: Opposition-based learning: a new scheme for machine intelligence. In: Proceedings of International Conference on Computational Intelligence for Modeling Control and Automation, pp. 695–701 (2005)
[10] Wang, H., Wu, Z.J., Rahnamayan, S.: Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. Soft Comput. 15(11), 2127–2140 (2011)
[11] Wang, H., Wu, Z.J., Rahnamayan, S., Kang, L.S.: A scalability test for accelerated DE using generalized opposition-based learning. In: Proceedings of International Conference on Intelligent System Design and Applications, pp. 1090–1095 (2009)
[12] Wang, H., Wu, Z.J., Rahnamayan, S., Liu, Y., Ventresca, M.: Enhancing particle swarm optimization using generalized opposition-based learning. Inform. Sci. 181(20), 4699–4714 (2011)
[13] Brest, J., Sepesy Maucec, M.: Population size reduction for the differential evolution algorithm. Appl. Intell., 228–247 (2008)
[14] Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186, 311–338 (2000)
[15] Koziel, S., Michalewicz, Z.: Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. Evol. Comput. 7(1), 19–44 (1999)
[16] Lampinen, J.: Multi-Constrained Optimization by the Differential Evolution. In: Proc. of the IASTED International Conference Artificial Intelligence Applications (AIA 2001), pp. 177–184 (2001)
[17] Lampinen, J.: Solving Problems Subject to Multiple Nonlinear Constraints by the Differential Evolution. In: Proc. of the 7th International Conference on Soft Computing, MENDEL 2001, pp. 50–57 (2001)

[18] Lampinen, J.: A Bibliography of Differential Evolution Algorithm. Technical Report, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing (2001)
[19] Lampinen, J.: A Constraint Handling approach for the Differential evolution Algorithm. In: Proc. of the Congress on Evolutionary Computation (CEC 2002), pp. 1468–1473 (2002)
[20] Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello Coello, C.A., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006, Special Session on Constrained Real-Parameter Optimization. Technical Report, Nanyang Technological University, Singapore (2005)
[21] Lin, Y.-C., Hwang, K.-S., Wang, F.-S.: Hybrid Differential Evolution with Multiplier updating method for Nonlinear Constrained Optimization. In: Proc. of the Congress on Evolutionary Computation (CEC 2002), pp. 872–877 (2002)
[22] Mezura-Montes, E., Velazquez-Reyes, J., Coello, C.: Modified differential evolution for constrained optimization. In: IEEE Congress on Evolutionary Computation, pp. 25–32 (2006)
[23] Tasgetiren, M.F., Suganthan, P.N., Pan, Q.-K., Liang, Y.-C.: A Differential Evolution Algorithm with a variable Parameter Search for Real-Parameter Continuous Function Optimization. In: The Proceeding of the World Congress on Evolutionary Computation (CEC 2009), Norway, pp. 1247–1254 (2009)
[24] Tasgetiren, M.F., Suganthan, P.N., Pan, Q.-K.: An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. Applied Mathematics and Computation 215(9), 3356–3368 (2010)
[25] Tasgetiren, M.F., Suganthan, P.N., Pan, Q.-K., Mallipedi, R., Sarman, S.: An ensemble of differential evolution algorithms for constrained function optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
[26] Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research 24, 1097–1100 (1997)
[27] Price, K., Storn, R., Lampinen, J.: Differential Evolution – A Practical Approach to Global Optimization. Springer (2005)
[28] Pan, Q.-K., Suganthan, P.N., Tasgetiren, M.F.: A Harmony Search Algorithm with Ensemble of Parameter Sets. In: IEEE Congress on Evolutionary Computation, CEC 2009, May 18-21, pp. 1815–1820 (2009)
[29] Gämperle, R., Müller, S.D., Koumoutsakos, P.: A parameter study for differential evolution. In: Proc. WSEAS Int. Conf. Advances Intell. Syst., Fuzzy Syst., Evol. Comput., pp. 293–298 (2002)
[30] Mallipedi, R., Suganthan, P.N.: Ensemble of constraint handling techniques. IEEE Trans. Evol. Comput. 14, 561–579 (2010)
[31] Mallipedi, R., Mallipedi, S., Suganthan, P.N., Tasgetiren, M.F.: Differential Evolution Algorithm with ensemble of parameters and mutation strategies. Applied Soft Comput. 11, 1679–1696 (2011)
[32] Elsayed, S.M., Sarker, R.A., Essam, D.L.: Multi-operator based evolutionary algorithms for solving constrained optimization problems. Computers & Operations Research 38, 1877–1896 (2011)
[33] Elsayed, S.M., Sarker, R.A., Essam, D.L.: On an evolutionary approach for constrained optimization problem solving. Applied Soft Computing 12, 3208–3227 (2012)
[34] Elsayed, S.M., Sarker, R.A., Essam, D.L.: A self-adaptive combined strategies algorithm for constrained optimization using differential evolution. Applied Mathematics and Computation 241, 267–282 (2014)

[35] Sarimveis, H., Nikolakopoulos, A.: A Line Up Evolutionary Algorithm for Solving Nonlinear Constrained Optimization Problems. Computers & Operations Research 32, 1499–1514 (2005)
[36] Smith, A.E., Tate, D.M.: Genetic Optimization Using a Penalty Function. In: Forrest, S. (ed.) Proc. of the Fifth International Conference on genetic Algorithms, pp. 499–503. Morgan Kaufmann (1993)
[37] Storn, R.: System Design by Constraint Adaptation and Differential Evolution. IEEE Transactions on Evolutionary Computation 3, 22–34 (1999)
[38] Storn, R., Price, K.: Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, ICSI (1995)
[39] Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space. Journal of Global Optimization 11, 341–359 (1997)
[40] Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-based differential evolution algorithms. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2010–2017 (2006)
[41] Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-based differential evolution for optimization of noisy problems. In: Proceedings of IEEE Congress on Evolutionary Computation (1872)
[42] Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-based differential evolution. IEEE Trans. Evol. Comput. 12(1), 64–79 (2008)
[43] Das, S., Suganthan, P.N.: Differential Evolution: A Survey of the State-of-the-Art. IEEE Trans. Evolutionary Computation 15(1), 4–31 (2011)
[44] Takahama, T., Sakai, S.: Constrained Optimization by the Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites. In: IEEE Congress on Evolutionary Computation, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp. 1–8 (2006)
[45] Long, W., Liang, X., Huang, Y., Chen, Y.: A hybrid differential evolution augmented Lagrangian method for constrained numerical and engineering optimization. Computer-Aided Design 45, 1562–1574 (2013)
[46] Gong, W., Cai, Z., Liang, D.: Engineering Optimization by means of an improved constrained differential evolution. Comput. Methods Appl. Mech. Engrg. 268, 884–904 (2014)
[47] Wang, Y., Cai, Z., Qingfu, Z.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. 15, 55–66 (2011)
[48] Jingqiao, Z., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archieve. IEEE Trans. Evol. Comput. 13, 945–958 (2009)
[49] Iztok, F., Marjan, M., Bogdan, F.: Graph 3-coloring with a hybrid self-adaptive evolutionary algorithm. Computational Optimization and Applications 54(3), 741–770 (2013)