A Differential Evolution Algorithm with Q-Learning for Solving Engineering Design Problems

Damla Kizilay
Industrial Engineering Department, Izmir Democracy University
Izmir, Turkey
damla.kizilay@idu.edu.tr

M. Fatih Tasgetiren
International Logistics Management Department, Yasar University
Izmir, Turkey
fatih.tasgetiren@yasar.edu.tr

Hande Oztop
Industrial Engineering Department Yasar University
Izmir, Turkey
hande.oztop@yasar.edu.tr

Levent Kandiller
Industrial Engineering Department Yasar University
Izmir, Turkey
levent.kandiller@yasar.edu.tr

P. N. Suganthan
School of Electrical and Electronic Engineering Nanyang Technological University, Singapore
epnsugan@ntu.edu.sg

***Abstract*— In this paper, a differential evolution algorithm with Q-Learning (DE-QL) for solving engineering Design Problems (EDPs) is presented. As well known, the performance of a DE algorithm depends on the mutation strategy and its control parameters, namely, crossover and mutation rates. For this reason, the proposed DE-QL generates the trial population by using the QL method in such a way that the QL guides the selection of the mutation strategy amongst four distinct strategies as well as crossover and mutation rates from the Q table. The DE-QL algorithm is well equipped with the epsilon constraint handling method to balance the search between feasible regions and infeasible regions during the evolutionary process. Furthermore, a new mutation operator, namely DE/Best to current/1, is proposed in the DE-QL algorithm. In this paper, 57 EDPs provided in "Problem Definitions and Evaluation Criteria for the CEC 2020 Competition and Special Session on A Test-suite of Non-Convex Constrained Optimization Problems from the Real-World and Some Baseline Results" are tested by the DE-QL. We provide our results in Appendixes and will be evaluated with other competitors in the competition.**

Keywords*— *differential evolution, engineering design problems, reinforcement learning, epsilon constraint handling method

I. INTRODUCTION

Constrained optimization is a highly important field of research as most of the real-world optimization problems are related to at least one constraint. It is difficult to solve constrained optimization problems because the solution space is divided into feasible and infeasible by constrained conditions. We consider the constrained optimization problem as follow [1]:

$$\min f(x) = (x_1, x_2, .., x_D) \quad (1)$$

$$St:$$

$$g_i(x) \leq 0, i = 1, .., p \quad (2)$$

$$h_j(x) = 0, j = p+1, .., m \quad (3)$$

where D is the dimension of the problem. A solution is regarded as feasible if $g_i(x) \leq 0$, for $i = 1, \dots, p$, and $|h_j(x)| - \varepsilon \leq 0$, for $j = p+1, \dots, m$. The total violation $\bar{v}$ of decision vector, x is defined as

$$\vartheta(x) = \sum_{i=1}^{p} G_i(x) + \sum_{j=p+1}^{m} H_j(x), \quad (4)$$

where

$$G_i(x) = \begin{cases} g_i(x) & if\ g_i(x) > 0 \\ 0 & if\ g_i(x) \leq 0 \end{cases} \quad (5)$$

$$H_j(x) = \begin{cases} |h_j(x)| & if\ |h_j(x)| - \varepsilon > 0 \\ 0 & if\ |h_j(x)| - \varepsilon \leq 0 \end{cases} \quad (6)$$

Differential evolution (DE) was proposed by Storn and Price [2]. It is a global optimizer for continuous optimization problems with a real value objective function. Due to its simplicity and efficiency, DE has been successfully applied in many fields. Excellent review of DE on state-of-the-art research can be found in the survey [3], [4].

For the last two decades, DE has been applied to several real-world problems and classical benchmark problems due to its simplicity and robustness [5]–[11]. Several DE variants have been proposed for solving constrained real-parameter optimization problems. In [12], the author proposed a success-history based adaptive differential evolution algorithm, so-called SHADE, which is a novel, adaptive DE algorithm, and an improvement over JADE [13]. SHADE maintains a diverse set of parameters to guide control parameter adaptation as the search progresses for a more robust search [12]. L-SHADE [14] extends SHADE with linear population size reduction, which continually decreases the population size according to a linear function. One of the first modifications of DE for constrained problems was proposed by Lampinen [15]. Wu et al. [9] also proposed a promising approach with a variable reduction strategy handling the equality constraints. These adaptive variants have generated much better results than the traditional DEs in the literature. Recently, three sophisticated DE variants based on the epsilon constraint method are presented in [16]–[18]. In this paper, we present a differential evolution algorithm with Q-Learning (DE-QL) for solving engineering design problems (EDPs). In this paper, 57 EDPs provided in "Problem Definitions and Evaluation Criteria for the CEC 2020 Competition on EDPs" are presented for the CEC2020 competition. The results are evaluated with other algorithms in the competition.

The remaining part of the paper is organized as follows. Section II explains the basic differential evolution. Section III gives details of the DE-QL algorithm. Computational results of

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

benchmark instances are shown in Section IV, whereas Section V summarizes the concluding remarks.

II. DE ALGORITHM

The traditional DE algorithm begins with a population of NP individuals. Each individual in NP has a D-dimensional vector with parameter values. Each vector is obtained randomly and uniformly within the search ranges $[x_{ij}^{min}, x_{ij}^{max}]$ as follows:

$$x_{ij}^{0} = x_{ij}^{min} + \left(x_{ij}^{max} - x_{ij}^{min}\right) \times r \quad (7)$$

where x_{ij}^{g} is the i^{th} target individual with respect to j^{th} dimension at generation g; and r is a uniform random number in [0,1]. Note that for each individual in the population, we keep the following information: $f(x)$ is the objective function value; total violation $v(x)$; then, $pf(x) = f(x) + v(x)$ is the penalized fitness value and; $fF(x)$ is a feasibility flag of a solution, where it is 0 if the solution is feasible, and 1 otherwise.

In each generation, mutation and crossover operators with parameters F and C_r are applied to each individual x_i ($i = 1,..,NP$). First a mutant individual v_i and then, a trial individual u_i is generated. If $f(u_i)$ is better than $f(x_i)$, x_i will be replaced by u_i. The algorithm evolves the population until the termination criterion (TC) is achieved, and then the best solution of the population is reported. The pseudo-code of DE is shown in Fig. 1.

Step 1. Determine parameters: NP, F, C_r and TC

Step 2. Initialization: Randomly generate a population $NP = \{x_1,..,x_n\}$ and evaluate each solution in NP.

Step 3. Population update: For each individual

a. Perform a mutation operator on x_i to generate v_i.
b. Perform a crossover operator on v_i and x_i to generate a trial vector u_i.
c. Update target individual replace x_i with u_i if $f(u_i) \leq f(x_i)$

Step 4: Termination: If stopping criterion is satisfied, report the best solution x_{best} in NP. Otherwise, go to Step 3.

Fig. 1. Differential evolution Algorithm

Some traditional mutation strategies are presented in the literature as follows:

S1. DE/rand/1:

$$v_i^g = x_a^{g-1} + F \times (x_b^{g-1} - x_c^{g-1}) \quad (8)$$

S2. DE/current to best/1:

$$v_i^g = x_i^{g-1} + F\left(x_{best}^{g-1} - x_i^{g-1}\right) + F(x_a^{g-1} - A_b^{g-1}) \quad (9)$$

S3. DE/Current to pbest/1:

$$v_i^g = x_i^{g-1} + F\left(x_{pbest}^{g-1} - x_i^{g-1}\right) + F(x_a^{g-1} - A_b^{g-1}) \quad (10)$$

S4. DE/Best from Best to current/1:

$$v_i^g = x_{best}^{g-1} + U(-1,1) \times (x_{best}^{g-1} - x_i^{g-1}), \quad (11)$$

where a, b, c are three randomly chosen individuals from the target population such that $(a \neq b \neq c \in (1,..,NP))$ and $j = 1,..,D$. $F \in (0,2)$ is a mutation scale factor affecting the differential variation between the two individuals from the population. Note that x_{pbest}^{g-1} is selected by tournament selection with size 2 and $U(-1,1)$ is a uniform random number between -1 and 1. Note that DE/Best to current/1 is presented for the first time in this paper. In addition to the above, note that these mutation strategies are used in the DE-QL algorithm.

During the generation of mutant individuals, they might be outside the search ranges. For this reason, parameter values violating the search range are restricted to below in this paper:

$$v_i^g = \begin{cases} x_i^{min} & if\ v_i^g < x_i^{min} \\ x_i^{max} & if\ v_i^g > x_i^{min} \end{cases} \quad (12)$$

Trial individuals are obtained by recombining mutant individuals with their corresponding target individuals as follows:

$$u_i^g = \begin{cases} v_i^g & if\ r_i^g \leq C_r\ or\ j = D_j \\ x_i^{g-1} & otherwise \end{cases} \quad (13)$$

where the index D_j is a randomly chosen dimension ($j = 1,..,D$). It makes sure that at least one parameter of the trial individual u_i^g will be different from the target individual x_i^{g-1}. C_r is a user-defined crossover constant in the range $[0,1]$, and r_i^g is a uniform random number in [0,1]...

III. DE_QL ALGORITHM

A. Q-Learning Procedure

In the DE-QL algorithm, the mutation strategy, as well as mutation and crossover rates, are all determined by the Q-Learning algorithm. The Q-learning (QL) is one of the widely used reinforcement learning algorithms [26]. The QL aims to choose an appropriate action based on experience by interacting with the environment. Once the agent (learner) performs a chosen action, it obtains a reward or penalty. Then, it learns to choose the best action to perform by assessing the action alternatives using the cumulative rewards (Q-values).

The Q-value can be calculated for each state-action pair by a Q-learning function given in Eq. (5). Then, Q-values are kept for all state-action pairs in a Q-value table. Let $S = [s_1, s_2, ... s_p]$ be the set of states, $A = [a_1, a_2, ... a_p]$ be the set of actions, r_{t+1} be the reward, $lf \in [0,1]$ be the learning factor, $df \in [0,1]$ be the discount factor and $Q(s_t, a_t)$ be the Q-value at time t. The learner aims to maximize its total reward.

$$Q_{t+1}(s_t, a_t) = Q(s_t, a_t) + lf[r_{t+1} + df * max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (14)$$

In the DE-QL algorithm, we assume that there is only one state for each parameter, where the reward value of 1 is given to an action that improves the current solution. We determine the mutation strategy (S), crossover rate (C_r), and mutation rate (F) from the Q-Table. Briefly, at each function evaluation, we update the Q-values of the chosen actions for all parameters through the Q-learning function. In other words, if a target individual is improved by a chosen action list, a reward 1 is assigned to each of the chosen action lists, and Q-Table is updated. Then, in the next function evaluation, the algorithm chooses the best action (value/strategy) for each parameter with the maximum Q-value. Note that, in the DE-QL algorithm, we also choose the actions of the parameters randomly with a small jumping probability (jP) to escape from local minima. The action list of each parameter is given in Table I.

TABLE I. ACTION LIST OF THE PARAMETERS

Parameter	Action List
S	{S1, S2, S3, S4}
F	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}
C_r	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}

B. Constraint Handling

For the next generation, we employ the superiority of feasible solutions (SF) and the ϵ-constraint handling method. As well-known, evolutionary operators can generate infeasible solutions. In this case, care must be taken with them violating the constraints. There exist different approaches to handle the constraints [4].

Deb [24] was the first to propose the superiority of feasible solutions (SF) for constrained optimization based on lexicographic ordering where constraint violation and objective function value are distinguished. The aim is to optimize both constraint violation and objective function by a lexicographic order where constraint violation precedes objective function value. In SF, when two solutions x_a and x_b are evaluated, a_a is deemed to be superior to x_b under the following conditions for a minimization problem:

- x_a is feasible and x_b is not.
- x_a and x_b are both feasible and x_a has a smaller objective function value than x_b.
- x_a and x_b are both infeasible, but x_a has a smaller overall constraint violation $viol(x)$ as computed by using Eq. (2).

Takahama and Sakai [25] developed the ε-constrained method, where the relaxation of the constraints is controlled by using the ϵ parameter. The ϵ level is updated until the generation counter g reaches the control generation g_C. After the generation counter exceeds g_C, the ϵ level is set to zero to obtain solutions with no constraint violation. The idea behind the EC method is that solutions with the total violation less that $\epsilon(g)$ are treated as feasible solutions when comparing two solutions. The ϵ-constraint handling method can be summarized as follows:

$$\epsilon(0) = \vartheta(x_\theta) \tag{15}$$

$$\epsilon(g) = \begin{cases} \epsilon(0)\left(1-\frac{g}{g_C}\right)^{cp} & 0 < g < g_C \\ 0 & g > g_C \end{cases} \tag{16}$$

where x_θ is the top θ^{th} individual.

Briefly, when we compare target individual with a trial individual by SF method, however, first, we assume that any individual having lower constraint violation than $\epsilon(g)$ level, we treat them as a feasible solution. Then, the SF method makes a selection between x_i and u_i.

C. Generalized Opposition-Based Learning

Opposition-based learning (OBL) is proposed by [27], which is a new method in computational intelligence and has been applied successfully to further improve various heuristic optimization algorithms [28-30]. OBL is based on the idea that when evaluating a solution to a given problem, its opposite solution gives a chance to find a candidate solution closer to the global optimal. Inspired from OBL, a generalized OBL $(GOBL)$ is introduced in [31-33]. Suppose that x is the current solution with $x \in [a, b]$. Then its opposite solution is given by:

$$x^* = k(a+b) - x \tag{17}$$

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

$$x_{ij}^* = k\left[a_j^g + b_j^g\right] - x_{ij}^g \tag{18}$$

$$a_j^g = min\left(x_{ij}^g\right), \quad b_j^g = max\left(x_{ij}^g\right) \tag{19}$$

$$x_{ij}^* = rand\left(a_j^g, b_j^g\right) \quad if\ x_{ij}^* < x_{min}\ or\ x_{ij}^* > x_{max}$$
$$i = 1,..,NP, j = 1,..,D, k = rand[0,1] \tag{20}$$

We employ the GOBL algorithm to further improve each trial individual u_i^g obtained by the DE_QL algorithm.

D. VNS Local Search

To develop a VNS local search, we employ the idea of using multiple neighborhood structures from the variable neighborhood search algorithm from [34]. We use two different mutation strategies in a VNS loop as well as the crossover operator to further improve trial individuals. We choose the S1 and S4 and mutation strategies as well as a uniform crossover to be employed in the VNS loop as follows: Note that tV, tU are temporary individuals.

$Procedure\ VNS(u_i^g)$
$k_{max} = 2$
$k = 1$
$tX = u_i^g$
$Generate\ Cr\ and\ F\ from\ Q_{t+1}(s_1, a_t)Q$
$do\{$
  $if\ (k == 1)\{$
    $tV = x_a^{g-1} + F \times (x_b^{g-1} - x_c^{g-1})$
    $tU = Crossover\ tV\ with\ tX$
  $\}else\ if\ (k == 2)\{$
    $tV = x_{best}^{g-1} + U(-1,1) \times (x_{best}^{g-1} - x_i^{g-1})$
    $tU = Crossover\ tV\ with\ tX$
  $endif$
  $if\ \left(pf(tU) < pf(tX)\right)\ then\{$
    $tX = tU$
    $k = 0$
    $R = 1\ and\ update\ Q - Table$
  $\}else\{$
    $k = k + 1$
  $endif$
$\}\ while\ (k \le k_{max})$
$u_i = tX$
$Update\ x_{best}^{g-1}\ with\ u_i^g$
$return\ u_i$
$endprocedure$

Fig. 2. Outline of VNS Local Search

We employ the VNS local search to further improve each trial individual u_i^g obtained by GOBL algorithm.

E. Linear Population Reduction Strategy

In order to improve the performance of DE-QL, a population size reduction strategy is used as in the original LSHADE. The population size NP dynamically decreases with the increasing of FEs according to Eq (X)

$$NP = Round\left(NP_{max} - \frac{FEs}{MaxFEs}(NP_{max} - NP_{min})\right) \quad (21)$$

F. Strategy and Parameter Selection

As mentioned before, mutation strategy, crossover and mutation rates are selected from the action lists maximizing the QT, but the random selection is also carried out to escape from local minima.

G. Selection for Next Generation

As mentioned before, when we compare target individual with a trial individual by SF method, however, first, we assume that any individual having lower constraint violation than $\epsilon(g)$ level, we treat them as a feasible solution. Then, the SF method makes a selection between x_i and u_i. Ultimately, the outline of the DE-QL algorithm is given in Fig. 3.

Step 1. **Determine parameters**
Step 2. **Initialize population**
Step 3. **Evaluate initial population**
  a. $f(x_1^0), .., f(x_{NP}^0)$
  b. Determine $x_{best}, \vartheta(x_i^0), pf(x_i^0), F(x_i^0); i = 1, .., NP)$
Step 4. **Population update:** For each x_i^{g-1}
  a. Choose a mutation strategy from Q-Table with jP
  b. Choose the mutation rate from Q-Table with jP
  c. Choose the crossover rate from Q-Table with jP
  d. Generate trial individual u_i^g
  e. Apply GOBL to u_i^g
  f. Apply VNS to u_i^g
  g. Update feasibility flag of u_i^g
    If $\left(\vartheta(u_i^g) \le \epsilon(g)\right)$, then $fF(u_i^g) = 0$; Else $fF(u_i^g) = 1$
  h. Make a selection
    If $\left(fF(u_i^g) = 0 \wedge fF(x_i^{g-1}) = 1\right)$
    Then $x_i^g = u_i^g \; and \; R = 1 \; and \; update \; Q - Table$
    Else If $\left(fF(u_i^g) = 0 \wedge fF(x_i^{g-1}) = 0\right)$
    Then $x_i^g = min\{f(u_i^g), f(x_i^{g-1})\} \; and$
    $R = 1 \; and \; update \; Q - Table$
    Else If $\left(fF(u_i^g) = 1 \wedge fF(x_i^{g-1}) = 1\right)$
    Then $x_i^g = min\{\vartheta(u_i^g), \vartheta(x_i^{g-1})\}$
  i. Update x_{best} as in selection
Step 4. **Update Q-Table**
Step 5**. Update population size NP**
Step 6. **Update epsilon level $\epsilon(g)$**
Step 7. **Termination**: If the termination criterion is satisfied, report x_{best}. Otherwise, go to step 4.

Fig. 3. Outline of DE-QL algorithm

IV. Computational Results

The algorithms proposed are coded in Visual C++ and run on an Intel Core 2 Quad 2.66 GHz PC with 3.5GB memory. Population size is taken as 90. Jumping probability is taken as jP=0.1. The algorithm was tested using benchmark instances in "Problem Definitions, and Evaluation Criteria for the CEC 2020 Competition and Special Session on A Test-suite of Non-Convex Constrained Optimization Problems from the Real-World and Some Baseline Results"[1] and compared to those algorithms in [16]–[18]. In line with guidelines in [1], we carried out all 25 runs. Note that we tried to convert the Matlab Codes into C codes. We could be able to convert most of the problems except for RC34-RC44 as well as RC27, RC33.

V. Conclusion

Computational results are given in Table II to V. DE-QL algorithm seems to be quite competitive to those three best-performing algorithms from the literature. During the competition, DE-QL algorithms can be further improved by improving the Q-Table. In the current one, the state is taken as 1, and the same crossover and mutation rate, as well as mutation strategy, are applied to all individuals in the population. We can define the states as population size and draw all parameters (actions) from a complete Q_{ij} Table. During the competition, we will try to convert the Matlab codes into C codes.

References

[1] A. Kumar, G. Wu, M. Ali, R. Mallipeddi, P. N. Suganthan, S. Das "Problem Definitions and Evaluation Criteria for the CEC 2020 Competition and Special Session on A Test-suite of Non-Convex Constrained Optimization Problems from the Real-World and Some Baseline Results". 2020.

[2] R. Storn and K. Price, "Differential Evolution - A simple evolution strategy for fast optimization," *Dr. Dobb's J.*, vol. 22, no. 4, pp. 18–24, 1997.

[3] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[4] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution-An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, 2016.

[5] T. Ray, R. Sarker, and X. Li, "Preface," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9592, no. c, p. v, 2016.

[6] N. M. Hamza, D. L. Essam, and R. A. Sarker, "Constraint Consensus Mutation-Based Differential Evolution for Constrained Optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 447–459, 2016.

[7] A. Zamuda, J. D. Hernández Sosa, and L. Adler, "Constrained differential evolution optimization for underwater glider path planning in sub-mesoscale eddy sampling," *Appl. Soft Comput. J.*, vol. 42, pp. 93–118, 2016.

[8] S. M. Guo, J. S. H. Tsai, C. C. Yang, and P. H. Hsu, "A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set," *2015 IEEE Congr. Evol. Comput. CEC 2015 - Proc.*, pp. 1003–1010, 2015.

[9] G. Wu, W. Pedrycz, P. N. Suganthan, and R. Mallipeddi, "A variable reduction strategy for evolutionary algorithms handling equality constraints," *Appl. Soft Comput. J.*, vol. 37, pp. 774–786, 2015.

[10] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci. (Ny).*, vol. 329, pp. 329–345, 2016.

[11] Lixin Tang, Yun Dong, and Jiyin Liu, "Differential Evolution With an Individual-Dependent Mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, 2014.

[12] R. Tanabe and A. Fukunaga, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," *2013 IEEE Congr. Evol. Comput. CEC 2013*, no. 1, pp. 1952–1959, 2013.

[13] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, 2009.

[14] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," *Proc. 2014 IEEE Congr. Evol. Comput. CEC 2014*, pp. 1658–1665, 2014.

[15] J. Lampinen, "A constraint handling approach for the differential evolution algorithm," *Proc. 2002 Congr. Evol. Comput. CEC 2002*, vol. 2, pp. 1468–1473, 2002.

[16] A. Trivedi, D. Srinivasan, N. Biswas, An improved unified differential evolution algorithm for constrained optimization problems, in: 2018 521 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1–10..

[17] M. Hellwig, H.-G. Beyer, A matrix adaptation evolution strategy for constrained real-parameter optimization, in: 2018 IEEE Congress on 523 Evolutionary Computation (CEC), IEEE, 2018, pp. 1–8.

[18] Z. Fan, Y. Fang, W. Li, Y. Yuan, Z. Wang, X. Bian, Lshade44 with an improved constraint-handling method for solving constrained 525 single-objective optimization problems, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1–8.

[19] A. Zamuda, "Adaptive constraint handling and Success History Differential Evolution for CEC 2017 Constrained Real-Parameter Optimization," *2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc.*, no. 1, pp. 2443–2450, 2017.

[20] R. Polakova, "L-SHADE with competing strategies applied to constrained optimization," *2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc.*, no. 2, pp. 1683–1689, 2017.

[21] J. Tvrdík and R. Poláková, "A simple framework for constrained problems with application of L-SHADE44 and IDE," *2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc.*, pp. 1436–1443, 2017.

[22] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," *2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc.*, pp. 1231–1238, 2017.

[23] P. Hansen, N. Mladenović, and D. Urošević, "Variable neighborhood search and local branching," *Comput. Oper. Res.*, vol. 33, no. 10, pp. 3034–3045, 2006.

[24] K. Deb, *Deb, K.: An Efficient Constraint Handling Method for Genetic Algorithm. Computer Methods in Applied Mechanics and Engineering 186, 311-338*, vol. 186. 2000.

[25] T. Takahama and S. Sakai, *Efficient Constrained Optimization by the ε Constrained Rank-Based Differential Evolution*. 2010.

[26] S.S. Choong, L-P. Wong, and C.P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," *Information Sciences*, vol. 436-437, pp. 89-107, 2018.

[27] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: Proceedings of International Conference on Computational Intelligence for Modeling Control and Automation, 2005, pp. 695–701.

[28] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution algorithms, in: Proceedings of IEEE Congress on Evolutionary Computation, 2006, pp. 2010–2017.

[29] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution for optimization of noisy problems, in: Proceedings of IEEE Congress on Evolutionary Computation, 2006, pp. 1865–1872.

[30] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution, IEEE Trans. Evol. Comput. 12 (1) (2008) 64–79.

[31] H. Wang, Z.J. Wu, S. Rahnamayan, Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems, Soft Comput. 15 (11) (2011) 2127–2140.

[32] H. Wang, Z.J. Wu, S. Rahnamayan, L.S. Kang, A scalability test for accelerated DE using generalized opposition-based learning, in: Proceedings of International Conference on Intelligent System Design and Applications, 2009,pp. 1090–1095.

[33] H. Wang, Z.J. Wu, S. Rahnamayan, Y. Liu, M. Ventresca, Enhancing particle swarm optimization using generalized opposition-based learning, Inform. Sci.181 (20) (2011) 4699–4714.

[34] P. Hansen, N. Mladenović, and D. Urošević, "Variable neighborhood search and local branching," *Comput. Oper. Res.*, vol. 33, no. 10, pp. 3034–3045, 2006.

Table II. Results of Industrial Chemical Processes problems (RC01 -RC07)

		RC01	RC02	RC03	RC04	RC05	RC06	RC07
Best	*f*	1.893116E+02	7.049037E+03	-4.529120E+03	-3.882604E-01	-4.000026E+02	1.133967E+00	1.634482E+00
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	6.727716E-02	8.747879E-02
Median	*f*	1.893116E+02	7.049037E+03	-4.529120E+03	-3.745107E-01	0.000000E+00	1.084778E+00	1.329929E+00
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.343533E-01	6.879491E-01
Mean	*f*	1.893413E+02	7.049037E+03	-4.524889E+03	-2.892402E-01	-8.000528E+00	1.110147E+00	1.248564E+00
	v	0.000000E+00	0.000000E+00	2.506583E-03	0.000000E+00	0.000000E+00	2.131301E-01	8.380490E-01
Worst	*f*	1.896505E+02	7.049037E+03	-4.494899E+03	-9.999996E-05	1.000000E+02	1.093033E+00	9.978200E-01
	v	0.000000E+00	0.000000E+00	2.088819E-02	0.000000E+00	0.000000E+00	7.361013E-01	2.394716E+00
Std	*f*	8.718300E-02	1.087359E-08	1.075976E+01	1.492452E-01	8.621718E+01	4.298594E-02	3.398562E-01
	v	0.000000E+00	0.000000E+00	6.927829E-03	0.000000E+00	0.000000E+00	2.115046E-01	8.362716E-01
FR		1.000000E+00	1.000000E+00	8.800000E-01	1.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
c		0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	1, 4, 5	5, 7, 0

Table III. Results of Process Synthesis and Design Problems (RC08 -RC14)

		RC08	RC09	RC10	RC11	RC12	RC13	RC14
Best	*f*	2.000000E+00	2.557655E+00	1.076543E+00	9.923846E+01	2.924831E+00	2.688742E+04	5.363894E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Median	*f*	2.000000E+00	2.557655E+00	1.076543E+00	9.924496E+01	2.924831E+00	2.688742E+04	5.363896E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	6.033899E-26	0.000000E+00	0.000000E+00	0.000000E+00
Mean	*f*	2.000000E+00	2.557655E+00	1.076543E+00	1.015512E+02	2.935533E+00	2.688742E+04	5.386137E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	2.414409E-27	0.000000E+00	0.000000E+00	0.000000E+00
Worst	*f*	2.000000E+00	2.557655E+00	1.076543E+00	1.078925E+02	3.081732E+00	2.688742E+04	5.919122E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Std	*f*	0.000000E+00	1.359740E-15	4.532467E-16	3.755235E+00	3.175656E-02	1.113899E-11	1.110386E+03
	v	0.000000E+00	0.000000E+00	0.000000E+00	1.206762E-26	0.000000E+00	0.000000E+00	0.000000E+00
FR		1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
c		0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 1	0, 0, 0	0, 0, 0	0, 0, 0

Table IV. Results of Mechanical Engineering Problems (RC15 -RC23)

		RC15	RC16	RC17	RC18	RC19	RC20	RC21	RC22	RC23
Best	*f*	2.994424E+03	3.221300E-02	1.266523E-02	8.077438E+03	1.670218E+00	2.638958E+02	2.352425E-01	-4.740975E-02	1.606987E+01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Median	*f*	2.994424E+03	3.221300E-02	1.266524E-02	1.093501E+04	1.670218E+00	2.638958E+02	2.352425E-01	-4.740975E-02	1.606987E+01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Mean	*f*	2.994424E+03	3.221788E-02	1.266581E-02	1.085138E+04	1.670218E+00	2.638958E+02	2.352425E-01	-4.504460E-02	1.606987E+01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Worst	*f*	2.994424E+03	3.233488E-02	1.267668E-02	1.355810E+04	1.670218E+00	2.638958E+02	2.352425E-01	-3.307692E-02	1.606987E+01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Std	*f*	4.641246E-13	2.437644E-05	2.275941E-06	1.459232E+03	1.868784E-16	1.640928E-14	1.133117E-16	4.862111E-03	2.346719E-08
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
FR		1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
c		0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0

Table V. Results of Mechanical Engineering Problems (RC24 -RC32)

		RC24	RC25	RC26	RC27	RC28	RC29	RC30	RC31	RC32
Best	*f*	-1.934170E+02	1.616122E+03	3.536096E+01	NA	1.461414E+04	2.964895E+06	2.658559E+00	1.232595E-32	-3.066554E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	NA	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Median	*f*	-1.803795E+02	1.661023E+03	3.726244E+01	NA	1.461414E+04	2.964895E+06	2.658559E+00	1.805425E-15	-3.066554E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	NA	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Mean	*f*	-1.812541E+02	1.679273E+03	3.734397E+01	NA	1.461414E+04	2.964895E+06	2.658559E+00	3.662051E-14	-3.066554E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	NA	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Worst	*f*	-1.762534E+02	1.918258E+03	4.166728E+01	NA	1.461414E+04	2.964895E+06	2.658559E+00	3.968012E-13	-3.066554E+04
	v	0.000000E+00	0.000000E+00	0.000000E+00	NA	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
Std	*f*	3.645657E+00	8.252875E+01	1.447422E+00	NA	5.569495E-12	1.400214E-09	4.532467E-16	8.680683E-14	3.236915E-12
	v	0.000000E+00	0.000000E+00	0.000000E+00	NA	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
FR		1.000000E+00	1.000000E+00	1.000000E+00	NA	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
c		0, 0, 0	0, 0, 0	0, 0, 0	NA	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0

Table VI. Results of Power Electronic Problems (RC45 -RC50)

		RC45	RC46	RC47	RC48	RC49	RC50
Best	*f*	8.817609E-02	1.239425E-01	4.567615E-02	5.721066E-01	5.895130E-02	2.865583E-01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.564735E-02
Median	*f*	8.818517E-02	1.240671E-01	4.665223E-02	5.721078E-01	5.895163E-02	2.865583E-01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.564735E-02
Mean	*f*	9.050944E-02	1.534037E-01	5.177782E-02	5.722402E-01	6.142385E-02	2.865602E-01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.564760E-02
Worst	*f*	1.158621E-01	2.989266E-01	8.568923E-02	5.739832E-01	9.273689E-02	2.866038E-01
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.565338E-02
Std	*f*	6.595029E-03	4.987410E-02	1.088397E-02	3.939256E-04	7.320693E-03	9.096783E-06
	v	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.205439E-06
FR		1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	0.000000E+00
c		0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0

Table VII. Results of Livestock Feed Ration Optimization Problems (RC51 -RC57)

		RC51	RC52	RC53	RC54	RC55	RC56	RC57
Best	*f*	4.548820E+03	3.510740E+03	4.997607E+03	4.240544E+03	1.918801E+03	1.047739E+04	2.289147E+03
	v	1.053822E-04	0.000000E+00	0.000000E+00	0.000000E+00	1.030933E-02	1.855236E-02	7.110005E-03
Median	*f*	4.550732E+03	3.820756E+03	5.050062E+03	4.240544E+03	2.114568E+03	1.149086E+04	2.812103E+03
	v	3.725959E-06	0.000000E+00	0.000000E+00	0.000000E+00	9.744082E-03	8.367024E-03	3.312377E-03
Mean	*f*	4.553275E+03	3.798519E+03	5.047819E+03	4.240568E+03	3.653526E+03	1.223436E+04	2.791247E+03
	v	1.574014E-05	0.000000E+00	0.000000E+00	8.715626E-07	6.891171E-03	7.714438E-03	3.019498E-03
Worst	*f*	4.566131E+03	4.109108E+03	5.173172E+03	4.237968E+03	6.660606E+03	1.368937E+04	2.874921E+03
	v	1.457714E-05	0.000000E+00	0.000000E+00	2.178907E-05	1.550304E-04	3.844068E-03	2.110542E-03
Std	*f*	5.782437E+00	1.808565E+02	5.398990E+01	8.160676E-01	2.132424E+03	1.189013E+03	1.200171E+02
	v	2.839097E-05	0.000000E+00	0.000000E+00	4.357813E-06	4.773834E-03	4.112392E-03	1.118437E-03
FR		0.000000E+00	1.000000E+00	1.000000E+00	9.600000E-01	0.000000E+00	0.000000E+00	0.000000E+00
c		0, 0, 3	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 1	0, 0, 0	0, 0, 1