Multi-Operator Differential Evolution Algorithm for Solving Real-World Constrained Optimization Problems

Karam M. Sallam *¶, Saber M. Elsayed†, Ripon K. Chakrabortty ‡, Michael J. Ryan§,
*§*Capability Systems Centre, School of Engineering & IT, University of New South Wales, Canberra, Australia,*
¶*The faculty of Computers and Information, Zagazig University, Egypt*
†‡*School of Engineering & IT, University of New South Wales, Canberra, Australia,*
*¶karam_sallam@zu.edu.eg, {†s.elsayed, ‡r.chakrabortty, §m.ryan}@unsw.edu.au

***Abstract*—Recently, many deferential evolution-based algorithms have been developed to solve constrained optimization problems. The performance of these methods outperforms the performance of single operator and/or algorithm-based ones. However, they do not perform consistently for all the problems tested in the literature. Also, the process of using the appropriate selection of algorithms and operators may be time-consuming since their designs are undertaken mainly through trial and error. In this paper, we propose an improved optimization algorithm that uses the benefits of multiple deferential evolution operators, with the best one is emphasized based on the quality and diversity of the population. The performance of the proposed algorithm is tested by solving 57 real-world constrained problems with different dimensions, number of equality and equality constraints, with its results showing a high success rate and that it outperformed different state-of-the-art algorithms.**

***Index Terms*—evolutionary algorithms, differential evolution, adaptive operator selection, constrained optimization.**

I. Introduction

The constrained optimization problems (COPs) are an important part of any problem in many fields including engineering and computer science [1]. The presence of additional constraints (inequality and equality) and their interrelationship with the objective function makes these problem more challenging than their unconstrained counterparts. These constraints functions are used to generate the feasible region that may be a well-shaped big space, a big irregular space, a tiny space, or a number of disjointed sets. Mathematically COPs can be represented by

$$\text{minimize } f(\overrightarrow{y})$$

$$\text{subject to: } g_k(\overrightarrow{y}) \leq 0,\ k = 1, 2, ..., K$$

$$h_e(\overrightarrow{y}) = 0,\ e = 1, 2, ..., E \quad (1)$$

$$L_j \leq y_j \leq U_j,\ j = 1, 2, ..., D \quad (2)$$

where s is the number of inequality constraints, $g_k(\overrightarrow{y})$, q equality constraints, $h_e(\overrightarrow{y})$, and each variable, y_j, has a lower and upper bound, L_j, U_j, respectively. The target of a COP, is to determine the values of all variables, y_1, y_2, ..., y_D, that minimize (or maximize) the objective function, $f(\overrightarrow{y})$, while satisfying all the constraints, including the boundary ones.

COPs posses different mathematical characteristics and proprieties, such as that their objective functions and constraints may be uni-modal or multi-modal, continuous or discontinuous, linear or nonlinear, and their variables can be discrete or real [2], [3]. The constraints functions are used to generate the feasible region that may be a well-shaped big space, a big irregular space, a tiny space, or a number of disjointed sets.These different characteristics make the process of locating the optimal solution challenging.

Computational intelligence (CI) based-methods, such as evolutionary algorithms (EAs), are widely used and have been successfully applied for COPs, since they have some essential advantages over traditional mathematical programming methods [4]. However, there is no guarantee that they will obtain optimum solutions and the quality of their solutions relies on the particular algorithm's design, the selection of its operators and its parameter settings. Among existing, differential evolution (DE), which is an population-based technique that use some sort of mutation, crossover and selection to generate new solutions and to guide them during the search to reach an optimal solution. Of them, DE has been extensively implemented in several fields, has gained popularity for solving problems in continuous domains and has proven its superiority over other well-known algorithms for solving complex optimization problems with different properties [5]–[7]. However, no single DE operator (or parameter) performs best for all types of test problems [1], [8]–[10]. This motivated researchers to introduce frameworks that use the strengths of different operators. Although they commonly enhance the optimization process, they still do not ensure reliable results for a wide range of test problems [11], [12] which reflects the need for better designs.

Considering the above-mentioned points, a multi-operator-based DE algorithm is introduced, that uses the strength of more than one DE operator, with more weight given to the better-one during the search process by dynamically updating the size of each sub-population based on two indicators, the quality of solutions and the diversity of each sub-populations. At the end of every generation, all sub-populations are gathered and then randomly redivided based on the new sub-population sizes. Indeed, to maintain the diversity of the population at the early stages of the optimization process and speed up convergence at later ones, a linear reduction of the population size is carried out. To handle constraints, instead of using all constraints to calculate the total violations, the algorithm starts with a subset of the most violated constrained are considered and gradually considers all of them. The performance of the proposed algorithm is judged by solving 57 real-world constrained problems taken from [13] competition on non-convex constrained optimization problems from the real-world. The results demonstrate that this algorithm statistically outperforms existing algorithms.

The rest of this paper is organized as follows: a brief literature review on related research studies is presented in Section II; the proposed approach and details of its components in Section III; the experimental results and analyses in Section IV; and, finally, the conclusions and future work in V.

II. Literature Review

Research on multi-operator differential algorithm (MODE) is still at its cuddle stage. Although there are opulence of works on MODE for unconstrained optimization problem, similar works on constrained optimization problem is still almost meagre. Among few, Elsayed et al. [3] proposed a self-adaptive MODE to solve different constraint optimization problems. Different success parameters such as constraint violation, feasibility ratio and solution quality were used to measure operator's performance, based on which number of solutions in each sub-population were dynamically updated. This self-adaptive MODE was further improved by Elsayed et al. [14] with integrating a co-variance matrix adaptation evolution strategy.

Another application of MODE can be found from the work of Li et al. [15], while they had generated three different trial vectors based on three different mutation strategies. Among those three mutation operators, two were used to increase the convergence rate of their proposed DE algorithm, and one operator was used to increase diversity. With successful application of their MODE approach, they showed the supremacy of MODE over traditional DEs in solving constrained optimization problems. Similarly, Yu et al. [16] also proposed an improved MODE algorithm, while they had utilized two different mutation operators to evolve the whole population. To solve constrained optimization problems, they had also proposed an advanced strategy to transform equality constraint into inequality constraints. Some notable extensions of MODE approach are: unified DE approach [17], improved unified DE approach [18], constrained optimization evolutionary algorithm [19], and constrained optimization composite DE algorithm [20]. While each of those advanced strategies had their own merits, however, the dynamic selection of optimal operator in a DE is still a challenging research context.

Considering the obvious necessity of having a well-established strategy to select best-performing evolutionary algorithm operators, Sallam et al. [21] proposed a landscape-based indicator to pick the best-suitable operator throughout the evolutionary process of DE. After solving many CEC constrained optimization test problems (both classical and real-application), they have proved the efficacy of their algorithm. Similar study was also conducted by in [1], while they proposed a MODE framework after considering objective functionâs landscape information in addition of operators performance history. Recently, Kumar et al. [22] proposed an enhanced version of MODE algorithm, in which they had applied an exponential population size reduction technique to reduce the size of the population for any subsequent iterations. After solving many CEC19 test problems, they have proved the supremacy of their proposed algorithm. In order to accelerate the searching capability of DE, Elsayed et al. [23] proposed a new mechanism to automatically select the best-fitted operators, such as amplification factor, crossover rate and population size. Different constrained optimization test problems were also employed to assess their algorithm.

Based on these relevant literature survey, it can be claimed that the search of finding appropriate selection of best-performing combination of operators and parameters for any evolutionary algorithm (say DE) is still considered as a tedious task, despite of a few earlier contributions from many researchers. Some obvious directions stemming from this research gap can be: (i) how to dynamically reduce the size of sub-populations based on the quality of solutions and the diversity of sub-populations, (ii) how to delve different operator's performance during the evolutionary process of a DE. Considering these as a challenging research context, this paper proposes an enhanced MODE (EnMODE) algorithm, which has the capacity to update population size dynamically based on solution quality and diversity of sub-populations.

III. Proposed Algorithm

As previously stated, the relative performance of DE operator may vary during the optimization stages, in other words, the performance of one DE operator may be good at the early phases of the optimization process and bad at the later ones, or vice versa [3], [9], [24]. Also, an operator may perform well for certain types of problem but its performance may be bad for another. As a result, the use of multi-operator DE is required with placing more weight to the better-performing one at each evolutionary stage. Algorithm 1 presents the main steps of the proposed EnMODE.

Firstly, a random initial population of size NP, i.e., $Y = \{\overrightarrow{y}_1, \overrightarrow{y}_2, ..., \overrightarrow{y}_{NP}\}$ is generated. Then, the fitness function ($f(\overrightarrow{y})$) and total constrained valuation ($\psi(\overrightarrow{y})$) for each solution is calculated. Then, the whole population is splitted into n_{op} sub-populations of equal size PS_{op}, each of which is evolved by a DE operator, i.e., new solutions u_i are generated using the assigned DE operators. Evaluate the new generated population and compute the fitness function and the total constraint violation for every new solution. Then, a comparison between each solution in the parent population and its corresponding one in the offspring population is carried out as discussed in subsection III-D, with the winner enters the next generation. Subsequently, at the end of each iteration, the improvement index is calculated as discussed at Section III-B, based on which the number of solutions in each sub-population is updated with a minimum sub-population size to each operator was set. Also, a linear population size reduction is conducted at the end of each iteration by removing the worst individual [25], as in Equation 3. This is done to preserve the diversity at the early generations and boosting the convergence in the later ones.

$$NP_{t+1} = round[(\frac{NP^{min} - NP^{init}}{MAX_{FES}}) \times FES + NP^{init}] \quad (3)$$

where NP^{min} is the minimum number of individuals the algorithm can use, FES the current number of function evaluations (FES), MAX_{FES} the largest number of FES.

Algorithm 1 EnMODE algorithm

```
1:  **Define** $n_{op}$, $MAX_{FES}$, $NP$, $t \leftarrow 1$ and $FES \leftarrow 0$;
2:  Generate an initial random population $(Y)$ of size $NP$, with the variables of each solution $(\overrightarrow{y}_i)$ must be within their boundaries;
3:  Evaluate $f(X)$ and calculate the constraints violations of $(\overrightarrow{y}_i)$ as discussed in Subsection III-D;
4:  Update number of fitness evaluations $FES \leftarrow FES + NP$;
5:  **while** $FES \leq MAX_{FES}$ **do**
6:      Distribute $Y$ randomly over $Y_{op}, \forall op = 1, 2, ..., n_{op}$, where each $Y_{op}$ is of size $PS_{op}$;
7:      Generate new population using the assigned DE operators, i.e., each operator op evolves its assigned number of individuals $NP_{op}$;
8:      **for** $op = 1 : n_{op}$ **do**
9:          Generate new solutions using the $op^{th}$ DE operators;
10:         Evaluate new generated solutions and calculate their constraints violations as discussed in Subsection III-D;
11:         Compute the improvement index value $(IIV_{op})$ as discussed in Section III-B;
12:         Update the values of $F$ and $Cr$ as in Section III-C;
13:     **end for**
14:     Update $FES$, $FES \leftarrow FES + NP$;
15:     Update the number of solutions $(PS_{op})$ each DE operator evolves (Section III-B);
16:     Update $NP$ (equation 3);
17:     $t \leftarrow t + 1$ ; and go to **step 5**;
18: **end while**
```

A. DE mutation strategies

In our algorithm, we use the following two DE mutation operators evolve the entire population as they perform well in solving unconstrained and constrained optimization problems [1], [26].

- DE/current-to-ϕbest/1/bin with archive

$$u_{i,j} = \begin{cases} x_{i,j} + F_i.(x_{\phi,j} - x_{i,j} + x_{r1,j} - x_{r2,j}) \\ \qquad \text{if } (rand \leq Cr_i \text{ or } j = j_{rand}) \\ x_{i,j} \qquad \qquad \text{otherwise} \end{cases} \quad (4)$$

- DE/rand-to-ϕbest/1/bin with archive

$$u_{i,j} = \begin{cases} x_{i,j} + F_i.(x_{\phi,j} - x_{i,j} + x_{r1,j} - x_{r2,j}) \\ \qquad \text{if } (rand \leq Cr_i \text{ or } j = j_{rand}) \\ x_{i,j} \qquad \qquad \text{otherwise} \end{cases} \quad (5)$$

where $r_1 \neq r_2 \neq r_3 \neq i$ are random integer numbers, $\overrightarrow{x}_{r1}$ and $\overrightarrow{x}_{r3}$ randomly selected from the whole population, with $x_{\phi,j}$ chosen from the best 10% of solutions in the whole populations and $x_{r2,j}$ from the union of the whole population and archive. Also, an archive is used to maintain the diversity of the population, with new solutions worse than their parent solutions putted into the archive [27]. To make space for the newly generated individuals, once the archive size is greater than its default size, the worst individuals are removed from it.

Note that the proposed framework is flexible to adopt more operators.

B. Updating number of individuals evolved by operator op (NP_{op})

As previously stated, the solutions' quality and sub-population's diversity are used to update the number of solutions in each sub-population.

For the solutions' quality, the best individual at the end of each generation in each sub-population is used, based on which the quality rate is calculated as

$$Qual_{op} = \frac{f_{t,op}^{best}}{\sum_{op=1}^{n_{op}} f_{t,op}^{best}}, \forall op = 1, 2, ..., n_{op} \quad (6)$$

where $f_{t,op}^{best}$ is the best objective function value obtained by operator *op* at generation t. Note, as we deal with constrained optimization problem, $f_{t,op}^{best}$ is determined based on both the fitness function values and the total constraint violations. This is done by sorting the solutions based on both fitness function values and total constraint violations.

Similarly, for the diversity obtained from each DE search operator is the mean deviation of each solution obtained from *op* from the best solution, i.e.,

$$Div_{op} = \frac{1}{NP_{op}} \left(\sum_{i=1}^{NP_{op}} dis(\overrightarrow{y}_{op,i} - \overrightarrow{y}_{op}^{best}) \right), \quad \forall op = 1, 2, ..., n_{op} \quad (7)$$

where $dis(\overrightarrow{y}_{op,i} - \overrightarrow{y}_{op}^{best})$ is the Euclidean distance between the i^{th} solution and best one obtained by each operator *op*. Also the diversity index is calculated by

$$DI_{op} = \frac{Div_{op}}{\sum_{op=1}^{n_{op}} Div_{op}}, \forall op = 1, 2, ..., n_{op} \quad (8)$$

Based on the above equations, the improvement index value (IIV_{op}) is calculated as

$$IIV_{op} = (1 - Qual_{op}) + DI_{op}, \forall op = 1, 2, ..., n_{op} \quad (9)$$

Note: to satisfy the aim of maximizing the IIV_{op}, we subtracted $Qual_{op}$ from one.

Finally, the number of individuals that each DE operator (NP_{op}) evolves is calculated by

$$PS_{op} = max\left(0.1, min\left(0.9, \frac{IIV_{op}}{\sum_{op=1}^{n_{op}} IIV_{op}}\right)\right) \times NP, \quad \forall op = 1, 2, ..., n_{op} \quad (10)$$

To avoid a sub-population of certain operator to have zero solutions a minimum value is used, that is equal to $0.1 * NP$.

Note: the summation of PS_{op} must equal the whole population size.

C. Managing F and Cr

The performance of the DE algorithm mainly depends on its search operators (mutation and crossover) and its control parameters (NP, F and Cr). However, choosing their values is challenging. Thus, in this paper, we used a self-adaptive technique to set the values of F and Cr [21], [26].

A historical memory of length H for both F and Cr is used. The values in these memories are expressed as and and their initial values were set to 0.5 and 0.2, respectively. Each individual ($\overrightarrow{x}_i$) is associated with its own (F_i) and (Cr_i), and their values are calculated using the following equations:

$$Cr_i = randni(\mu_{Cr,r_i}, 0.1) \tag{11}$$

$$F_i = randci(\mu_{F,r_i}, 0.1) \tag{12}$$

where r_i is randomly chosen from $[1, H]$, $randni$ and $randci$ are two functions used to generate random numbers from Normal and Cauchy distributions with mean μ_{Cr,r_i} and μ_{F,r_i}, respectively, with variance 0.1. If the Cr_i values are outside [0, 1], they are replaced by the limit value (0 or 1) nearest to the generated value. If $F_i > 1$, its value is replaced by 1, and if $F_i \leq 0$, Equation 12 is repeatedly executed until a valid value is reached.

At the end of each generation (t), the F_i and Cr_i used by the successful individuals are inserted in S_F and S_{Cr}, then the values inside the historical memories are updated by

$$\mu_{Cr,h,t+1} = \begin{cases} meanw_L(S_{Cr}) & \text{if } S_{Cr} \neq \phi \\ \mu_{Cr,h,t} & \text{otherwise} \end{cases} \tag{13}$$

$$\mu_{F,h,t+1} = \begin{cases} meanw_L(S_F) & \text{if } S_F \neq \phi \\ \mu_{F,h,t} & \text{otherwise} \end{cases} \tag{14}$$

where $1 \leq h \leq H$ is the location of the historical memory to update. The value of h is set initially to 1 and is consequently increased whenever a new element is inserted into the memories. In case of $h > H$, h is reset to 1 and $meanw_L(S_{Cr})$, the Lehmer mean, is calculated by

$$meanw_L(S_{Cr}) = \frac{\sum_{h=1}^{|S_{Cr}|} \omega_h . S_{Cr_h}^2}{\sum_{h=1}^{|S_{Cr}|} \omega_h . S_{Cr_h}} \tag{15}$$

where ω_h is the weight calculated by

$$\omega_h = \frac{\gamma_h}{\sum_{h=1}^{|S_{Cr}|} \gamma_h} \tag{16}$$

The values of γ_h is calculated based on the following scenarios:

1) Feasible to feasible: the best solution feasible at both generations $t-1$ and t;
2) Infeasible to feasible: the best solution in the population is infeasible at $t-1$ and becomes feasible at t; and
3) Infeasible to infeasible: the best solution in the population is infeasible in both generations $t-1$ and t.

Firstly, for each successful solution ($h \in 1, 2, ..., |S_{Cr}|$ which falls in case 1, its γ_h is calculated as:

$$\gamma_h = \beta_{h=} max\left(0, \frac{\psi_{h,t-1} - \psi_{h,t}}{\psi_{h,t-1}}\right) + max\left(0, \frac{f_{h,t-1} - f_{h,t}}{f_{h,t-1}}\right) \tag{17}$$

Then, for each successful solutions which exists in scenarios 2 or 3, its γ_h is computed as:

$$\gamma_h = max(0, \beta_h) + \frac{\psi_{h,t-1} - \psi_{h,t}}{\psi_{h,t-1}} + max\left(0, \frac{f_{h,t-1} - f_{h,t}}{f_{h,t-1}}\right) \tag{18}$$

D. Constraints handling

In this paper, we used the constrained handling technique proposed in [23]. It starts by sorting the constraints according to the sum of the constraints violations of all solutions in the initial population from the least violated to the most violated constraint or vice versa. It is well known that the more number of constraints exist in any problem, the more complex will be that problem. Instead of using all constraints, the algorithm starts with a subset of the constraints (Con_n) for a predefined number of generations (W). After W generations, a new subset is then added to the current one, and the method tries to attain the feasible region of both the new and previous sunsets of constraints. The process lasts until all the constraints are handled and the final feasible region is reached.

To select between any solution and its parent, the method developed by Deb [28] is adopted and it has three scenarios:

1) from two feasible individuals, the one with the best function value is selected;
2) from two infeasible individuals, the one with the smallest sum of constraint violations (ψ) is chosen, where ψ is computed using Equation 19; and
3) a feasible solution is always better than an infeasible one.

$$\psi(\overrightarrow{x}_i) = \sum_{k=1}^{K} max(0, g_k(\overrightarrow{x}_i)) + \sum_{e=1}^{E} max(0, |h_e(\overrightarrow{x}_i)| - \delta_e) \tag{19}$$

where $g_k(\overrightarrow{x}_i)$ and $h_e(\overrightarrow{x}_i)$ are the k^{th} inequality and e^{th} equality constraints, respectively. For each equality constraint h_e, δ_e was set to a value of 0.0001.

IV. Experimental Results and analysis

To judge the performance of the proposed EnMODE algorithm, several experiments were conducted on 57 constrained optimization problems with different dimensions ranges from 2 variables to 158 variables and different number of equality and inequality constraints ranges from 2 constraints to 148 constraints. Details of these problems can be found in [13]. These problems are categorized to six sets:

- Industrial Chemical Processes contains seven test problems (RC01 - RC07);
- Process Synthesis and Design Problems has seven test problems (RC08 - RC14);
- Mechanical Engineering Problem includes 19 test problems (RC15 - RC33);
- Power System Problems has 11 test problem (RC34 - RC44);

- Power Electronic Problems contains six test problems (RC45 - RC50); and
- Livestock Feed Ration Optimization has seven test problems (RC51 - RC57).

The proposed EnMODE was coded in Matlab R2018b and run on a PC with 3.4 GHz Core I7 processor, 16 GB RAM and Windows 10. For all the other comparative algorithms, the values of their parameters were obtained from relevant articles and, to ensure fair comparisons, they all used the same seed. According to the benchmark rules, all the algorithms were run 25 times for MAX_{FES} calculated by Equation 20, or $|f(\overrightarrow{x}_{best} - f(\overrightarrow{x^*})| = 0$, where x^* is the global optimal solution and x_{best} the best solution obtained by the proposed algorithm, with the average and standard deviation results recorded.

$$MAX_{FES} = \begin{cases} 1 \times 10^5 & \text{if } D \leq 10 \\ 2 \times 10^5 & \text{if } 10 < D \leq 30 \\ 4 \times 10^5 & \text{if } 30 < D \leq 50 \\ 8 \times 10^5 & \text{if } 50 < D \leq 150 \\ 10^6 & \text{if } 150 < D \end{cases} \quad (20)$$

For statistical comparisons of the algorithms, we conducted two non-parametric tests (the Wilcoxon signed-rank and Friedman ranking tests [29]). The proposed algorithm's performances were also graphically judged by plotting their performance profiles [30] which is a tool used to compare the performance of several methods (M) using several test functions (P) and a comparison goal (i.e., the average computational time and number of FES) to attain certain level of the performance criteria (i.e., optimal fitness function value). For a method (A), the performance profile Rho_A is calculated as

$$Rho_A(\tau) = \frac{1}{n_p} \times |p \in P : r_{p,A} \leq \tau| \quad (21)$$

where $Rho_A(\tau)$ is the percentage of $A \in M$ that the performance ratio $r_{p,m}$ is within a factor $\tau \in R$ for the best possible probability and Rho_A a function that returns the cumulative distribution for the $r_{p,A}$.

A. Parameter Setting

In terms of the algorithms' parameters: NP^{init} was set to a value of 200 solutions, NP^{min} to 4, the window size W to 50 generations, Con_n to $(E + K)/2$, the archive rate (A) to 1.4 and memory size (H) to 5.

B. Detailed results obtained from EnMODE

The detailed results obtained from the proposed EnMODE are presented at this section. The best, median, average, worst and the standard deviation of each problem associated with their violation values are presented at Tables I - VI .

The proposed algorithm was able to obtain the feasible solutions for 40 problems out of 57 problems. However, the violations for the problems which the algorithm was not able to obtain feasible solutions were very small.

EnMODE was able to obtain the optimal solutions for 5 test problems in the Industrial Chemical Processes test problems set (RC01-RC05). It was also able to achieve the optimal solutions for five test problems in the Process Synthesis and Design Problems test problems set (RC08, RC09, RC10, RC12 and RC13). The proposed algorithm was able to obtain the optimal solutions for all test problems in the Mechanical Engineering Problem test problem set (RC15 - RC33). The performance of the proposed EnMODE was deteriorated when solving the power system problems, as it was able to obtain the optimal solution for only one test problems and the same behaviour happened for the Livestock Feed Ration Optimization problem set. For the Power Electronic Problems, the proposed algorithm was able to obtain solutions that were very close to the optimal.

The performance of EnMODE was compared with those obtained from (1) IUDE: Improved variant of Unified Differential Evolution [18]; (2) ϵMAgES: Matrix Adaptation Evolution Strategy with $\epsilon-$constraint; and [31]; (3) LSHADE44: LSHADE with IEpsilon [32].

Firstly, the proposed algorithm was able to reach 70.18% FRs for all test problems. IUDE achieved 54.39% FRs, while the FRs for ϵMAgES and LSHADE44 were 49.12% and 59.65%, respectively.

Regarding the quality of solutions and based on the best results, as shown in Table VIII, EnMODE was superior to IUDE, ϵMAgES and LSHADE44 for 40, 35 and 41 test problems, respectively, similar for 3, 3 and 2 test problems, respectively, and inferior for 14, 19 and 14 test problems, respectively. Considering the median obtained results, EnMODE was superior to IUDE, ϵMAgES and LSHADE44 for 38, 39 and 46 test problems, respectively, equal in 1, 1 and 1 test problems, respectively, and worse for 18, 17 and 10 test problems, respectively. Regarding the average results, the proposed EnMODE was better than IUDE, ϵMAgES and LSHADE44 for 36, 35 and 48 test problems respectively, worse for 20, 22 and 8 test problems, respectively, and similar for 1, 0 and 1 test problems, respectively.

The Wilcoxon test was carried out to see whether the proposed algorithm was statistically better than the rival algorithms, with the obtained results shown in Table VIII. It was clear that EnMODE was statistically better than all the other algorithms for the obtained best, median and average results. Also, the Friedman test results depicted in Table IX demonstrate that EnMODE was ranked first for the best, median and average results.

For further analysis, a graph of the performance profiles for the test problems plotted to compare all the algorithms for both mean and median results is presented in Figure 1. It indicates that consistent results were obtained from the Friedman and Wilcoxon tests as EnMODE reaches a probability of 1.0 first at $\tau \approx 16$ and $\tau \approx 1900$ for mean and median results, respectively.

V. Conclusion and Future Work

A multi-operator DE algorithm is proposed to solve real-world constrained optimization problems. The better-performing operator is given more weight based on two criteria, the solutions quality and population diversity. To handle constraints, instead of considering all constraints, first, the proposed algorithm considers a subset of the most violated constraints and gradually incorporates all of them to

TABLE I: Outcome at FES=MAX_{FES} for Problems RC01-RC08

Criteria		**RC01**	**RC02**	**RC03**	**RC04**	**RC05**	**RC06**	**RC07**	**RC08**
Best	f	1.8931E+02	7.0490E+03	-4.5291E+03	-3.8826E-01	-4.0001E+02	1.0835E+00	9.5104E-01	2.0000E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.3325E-01	1.0067E-01	0.0000E+00
Median	f	1.8931E+02	7.0490E+03	-4.5291E+03	-3.7440E-01	-3.9707E+02	1.0843E+00	1.0901E+00	2.0000E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	9.3738E-02	5.1155E-01	0.0000E+00
Mean	f	1.8931E+02	7.0490E+03	-4.3537E+03	-3.7574E-01	-3.3781E+02	1.0870E+00	1.1617E+00	2.0000E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.5174E-01	1.8113E-01	0.0000E+00
Worst	f	1.8931E+02	7.0490E+03	-1.4272E+02	-3.6918E-01	0.0000E+00	1.1121E+00	1.5976E+00	2.0000E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.1157E-01	1.6751E-01	0.0000E+00
Std.	f	2.1707E-14	0.0000E+00	8.7728E+02	4.6905E-03	1.3137E+02	7.4308E-03	1.9498E-01	0.0000E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	1.2371E-01	1.9189E-01	0.0000E+00
FR		1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	0.0000E+00	0.0000E+00	1.0000E+00
c		[000]	[000]	[000]	[000]	[000]	[010]	[010]	[000]

TABLE II: Outcome at FES=MAX_{FES} for Problems RC09-RC16

Criteria		**RC09**	**RC10**	**RC11**	**RC12**	**RC13**	**RC14**	**RC15**	**RC16**
Best	f	2.5577E+00	1.0765E+00	9.9239E+01	2.9248E+00	2.6887E+04	5.8505E+04	2.9944E+03	3.2213E-02
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Median	f	2.5577E+00	1.0765E+00	1.0738E+02	2.9248E+00	2.6887E+04	5.8505E+04	2.9944E+03	3.2213E-02
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Mean	f	2.5577E+00	1.1529E+00	1.0510E+02	2.9248E+00	2.6887E+04	5.8505E+04	2.9944E+03	3.2213E-02
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Worst	f	2.5577E+00	1.2500E+00	1.0738E+02	2.9248E+00	2.6887E+04	5.8505E+04	2.9944E+03	3.2213E-02
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Std.	f	1.3597E-15	8.7877E-02	3.7287E+00	4.5325E-16	1.1139E-11	8.0620E-09	4.6412E-13	3.1672E-18
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
FR		1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00
c		[000]	[000]	[000]	[000]	[000]	[000]	[000]	[000]

TABLE III: Outcome at FES=MAX_{FES} for Problems RC17-RC24

Criteria		**RC17**	**RC18**	**RC19**	**RC20**	**RC21**	**RC22**	**RC23**	**RC24**
Best	f	1.2665E-02	6.0597E+03	1.6702E+00	2.6390E+02	2.3524E-01	5.2577E-01	1.6070E+01	2.5438E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Median	f	1.2719E-02	6.0597E+03	1.6702E+00	2.6390E+02	2.3524E-01	5.2628E-01	1.6070E+01	2.5438E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Mean	f	1.2710E-02	6.0597E+03	1.6702E+00	2.6390E+02	2.3524E-01	5.2691E-01	1.6070E+01	2.5438E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Worst	f	1.2719E-02	6.0597E+03	1.6702E+00	2.6390E+02	2.3524E-01	5.3121E-01	1.6070E+01	2.5438E+00
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Std.	f	2.0138E-05	9.2825E-13	0.0000E+00	0.0000E+00	1.1331E-16	1.4402E-03	3.3335E-14	1.3501E-12
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
FR		1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00
c		[000]	[000]	[000]	[000]	[000]	[000]	[000]	[000]

TABLE IV: Outcome at FES=MAX_{FES} for Problems RC25-RC32

Criteria		**RC25**	**RC26**	**RC27**	**RC28**	**RC29**	**RC30**	**RC31**	**RC32**
Best	f	1.6161E+03	3.5359E+01	5.2445E+02	1.6958E+04	2.9649E+06	2.6586E+00	0.0000E+00	-3.0666E+04
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Median	f	1.6161E+03	3.5364E+01	5.2445E+02	1.6958E+04	2.9649E+06	2.6586E+00	0.0000E+00	-3.0666E+04
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Mean	f	1.6161E+03	3.5728E+01	5.2445E+02	1.6958E+04	2.9649E+06	2.8149E+00	0.0000E+00	-3.0666E+04
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Worst	f	1.6161E+03	3.7268E+01	5.2445E+02	1.6958E+04	2.9649E+06	3.6359E+00	0.0000E+00	-3.0666E+04
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Std.	f	1.7751E-11	5.9911E-01	3.7602E-07	3.7130E-12	1.4258E-09	3.6570E-01	0.0000E+00	3.7130E-12
	v	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
FR		1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00
c		[000]	[000]	[000]	[000]	[000]	[000]	[000]	[000]

TABLE V: Outcome at FES=MAX_{FES} for Problems RC33-RC40

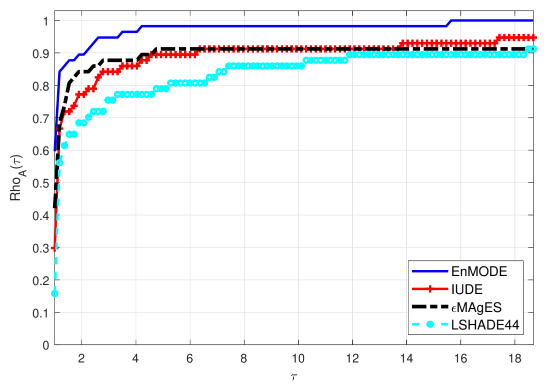
Criteria		RC33	RC34	RC35	RC36	RC37	RC38	RC39	RC40
Best	f	2.6393E+00	1.8763E-08	-5.5866E+01	-7.9982E+01	-1.3614E+01	-2.9661E+00	-7.7307E+00	8.7659E+00
	v	0.0000E+00	1.4935E-01	2.2549E+00	4.6087E+00	2.8689E-01	1.1281E-01	1.4295E-01	1.0120E+00
Median	f	2.6393E+00	1.8453E+00	9.3738E+01	6.0832E+01	1.7859E+00	8.3966E-01	-7.3630E-01	2.3980E+01
	v	0.0000E+00	1.3138E-01	8.4722E-01	1.5698E+00	1.6146E-01	9.6510E-02	1.2017E-01	1.1998E+00
Mean	f	2.6393E+00	2.9323E+00	7.3181E+01	4.8808E+01	1.1398E+00	1.0646E+00	-8.1795E-01	2.5588E+01
	v	0.0000E+00	1.2040E-01	1.2654E+00	2.2869E+00	1.3024E-01	1.0150E-01	1.0001E-01	1.2460E+00
Worst	f	2.6393E+00	1.1456E+01	1.4570E+02	1.4590E+02	4.5562E+00	5.7313E+00	3.0522E+00	5.1304E+01
	v	0.0000E+00	5.7930E-02	7.7016E-01	1.0586E+00	7.0381E-02	7.4837E-02	6.5590E-02	1.3763E+00
Std.	f	1.0175E-15	3.5085E+00	6.4342E+01	6.7606E+01	3.5383E+00	1.8288E+00	2.3655E+00	1.0875E+01
	v	0.0000E+00	2.8552E-02	6.0842E-01	1.1642E+00	4.4721E-02	1.6835E-02	1.6211E-02	3.1683E-01
FR		1.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
c		[000]	[010]	[010]	[100]	[010]	[010]	[010]	[100]

TABLE VI: Outcome at FES=MAX_{FES} for Problems RC41-RC48

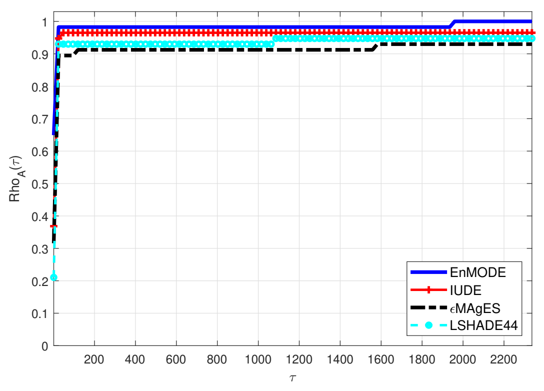
Criteria		RC41	RC42	RC43	RC44	RC45	RC46	RC47	RC48
Best	f	3.3908E-02	-1.0930E+01	-1.8864E+02	-6.1869E+03	7.4334E-02	5.1508E-02	3.8154E-02	4.0363E-02
	v	1.7279E-01	2.4691E+00	5.1686E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Median	f	2.9766E+00	-1.2349E+00	1.4642E+01	-6.0817E+03	1.3718E-01	6.3260E-02	6.7741E-02	4.7687E-02
	v	4.7309E-01	1.3898E+00	1.3169E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Mean	f	1.4269E+03	-2.1680E+00	-1.3935E+00	-6.0838E+03	1.4324E-01	6.3581E-02	6.4366E-02	7.0063E-02
	v	2.7833E+00	2.2045E+00	2.5476E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Worst	f	2.9304E+04	-8.2359E-01	3.9665E+01	-6.0048E+03	3.0063E-01	7.4371E-02	9.8325E-02	5.5003E-01
	v	9.3334E+00	2.3335E+00	2.2846E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
Std.	f	5.8475E+03	2.2339E+00	4.9951E+01	5.3493E+01	5.4304E-02	5.1785E-03	1.6878E-02	1.0023E-01
	v	3.4535E+00	5.2961E-01	8.9008E-01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
FR		0.0000E+00	0.0000E+00	0.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00	1.0000E+00
c		[010]	[100]	[100]	[000]	[000]	[000]	[000]	[000]

TABLE VII: Outcome at FES=MAX_{FES} for Problems RC49-RC57

Criteria		RC49	RC50	RC51	RC52	RC53	RC54	RC55	RC56	RC57
Best	f	3.7313E-02	1.3654E-01	4.4582E+03	3.3497E+03	4.3452E+03	3.1921E+03	1.6943E+03	9.0491E+03	1.9627E+03
	v	0.0000E+00	2.0665E-02	1.0804E-03	0.0000E+00	4.5582E-03	4.0751E-03	9.7894E-03	1.0635E-02	1.9552E-03
Median	f	9.8783E-02	3.5610E-01	4.4986E+03	3.3633E+03	4.3633E+03	3.3416E+03	5.6877E+03	1.1976E+04	2.4707E+03
	v	0.0000E+00	1.2873E-03	3.8898E-04	0.0000E+00	5.2830E-04	1.3538E-03	9.1415E-03	7.7943E-03	4.9139E-04
Mean	f	9.4150E-02	3.2722E-01	4.5030E+03	3.3682E+03	4.6761E+03	3.3347E+03	4.9375E+03	1.1419E+04	2.4686E+03
	v	0.0000E+00	1.0732E-02	3.4746E-04	0.0000E+00	2.8314E-03	1.4806E-03	3.1987E-03	5.8596E-03	1.0306E-03
Worst	f	1.9977E-01	4.0885E-01	4.5451E+03	3.4021E+03	5.6477E+03	3.3417E+03	6.7116E+03	1.2962E+04	3.5415E+03
	v	0.0000E+00	1.5889E-02	4.0282E-05	0.0000E+00	0.0000E+00	1.3536E-03	0.0000E+00	2.6854E-03	6.4600E-05
Std.	f	4.8095E-02	7.0787E-02	1.8198E+01	1.5193E+01	4.3181E+02	3.0073E+01	1.6876E+03	1.2173E+03	3.7414E+02
	v	0.0000E+00	4.0623E-03	1.9144E-04	0.0000E+00	1.9939E-03	5.4792E-04	3.4431E-03	2.7461E-03	5.4227E-04
FR		1.0000E+00	0.0000E+00	0.0000E+00	1.0000E+00	1.2000E-01	0.0000E+00	4.0000E-02	0.0000E+00	0.0000E+00
c		[000]	[001]	[001]	[000]	[001]	[001]	[001]	[001]	[001]



(a)



(b)

Fig. 1: Performance profiles graphs comparing the performance of EnMODE, IUDE, ϵMAgES and LSHADE44 based (a) average results and (b) median results

TABLE VIII: Summary of comparisons of performances of EnMODE, IUDE, ϵMAgES, and LSHADE44

Algoeithms	Criteria	Better	Similar	Worse	(p, Dec.)
EnMODE vs. IUDE	Best	40	3	14	(0.000, +)
	Median	38	1	18	(0.004, +)
	Mean	36	1	20	(0.003, +)
EnMODE vs. ϵMAgES	Best	35	3	19	(0.001, +)
	Median	39	1	17	(0.000, $\approx$)
	Mean	35	0	22	(0.013, +)
EnMODE vs. LSHADE44	Best	41	2	14	(0.000, $\approx$)
	Median	46	1	10	(0.000, $\approx$)
	Mean	48	1	8	(0.000, $\approx$)

TABLE IX: Ranking of all algorithms EnMODE, IUDE, ϵMAgES and LSHADE44 obtained by Friedman test

Algorithms	**Best**	**Median**	**Mean**
EnMODE	**1.89**	**1.89**	**1.82**
IUDE	2.61	2.36	2.38
ϵMAgES	2.42	2.52	2.66
LSHADE44	3.07	3.23	3.15

calculate the total constraint violations. The performance of the proposed algorithm was judged by solving 57 real-world constrained optimization problems with different dimensions vary from 2 to 157, number of equality constraints vary from 0 to 148, and number of equality constraints vary from 0 to 105. The computational results showed that it was 100% statistically better than or similar to the rival algorithms.

REFERENCES

[1] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-based adaptive operator selection mechanism for differential evolution," *Information Sciences*, vol. 418, pp. 383–404, 2017.

[2] K. M. Sallam, R. A. Sarker, and D. L. Essam, "Reduced search space mechanism for solving constrained optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 65, pp. 147–158, 2017.

[3] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization problems," *Computers & operations research*, vol. 38, no. 12, pp. 1877–1896, 2011.

[4] K. Deb, *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[5] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.

[6] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[7] K. M. Sallam, R. A. Sarker, D. L. Essam, and S. M. Elsayed, "Neurodynamic differential evolution algorithm and solving cec2015 competition problems," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 1033–1040.

[8] S. M. Elsayed, "Evolutionary approach for constrained optimization," PhD dissertation, University of New South Wales, Canberra, Australia, 2012.

[9] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Improved united multi-operator algorithm for solving optimization problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[10] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-method based orthogonal experimental design algorithm for solving cec2017 competition problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 1350–1357.

[11] S. Elsayed, R. Sarker, and C. A. C. Coello, "Fuzzy rule-based design of evolutionary algorithm for optimization," *IEEE transactions on cybernetics*, vol. 49, no. 1, pp. 301–314, 2017.

[12] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Two-phase differential evolution framework for solving optimization problems," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–8.

[13] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, "Test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *submitted to: Swarm and Evolutionary Computation*, 2019.

[14] S. M. Elsayed, R. A. Sarker, and T. Ray, "Differential evolution with automatic parameter configuration for solving the cec2013 competition on real-parameter optimization," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1932–1937.

[15] W. Li, S. Li, Z. Chen, L. Zhong, and C. Ouyang, "Self-feedback differential evolution adapting to fitness landscape characteristics," *Soft Computing*, vol. 23, no. 4, pp. 1151–1163, 2019.

[16] X. Yu, Y. Lu, X. Wang, X. Luo, and M. Cai, "An effective improved differential evolution algorithm to solve constrained optimization problems," *Soft Computing*, vol. 23, no. 7, pp. 2409–2427, 2019.

[17] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 1231–1238.

[18] A. Trivedi, D. Srinivasan, and N. Biswas, "An improved unified differential evolution algorithm for constrained optimization problems," in *Proceedings of 2018 IEEE Congress on Evolutionary Computation*. IEEE, 2018, pp. 1–10.

[19] B. Xu, X. Chen, and L. Tao, "Differential evolution with adaptive trial vector generation strategy and cluster-replacement-based feasibility rule for constrained optimization," *Information Sciences*, vol. 435, pp. 240–262, 2018.

[20] B.-C. Wang, H.-X. Li, J.-P. Li, and Y. Wang, "Composite differential evolution for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 7, pp. 1482–1495, 2018.

[21] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-assisted multi-operator differential evolution for solving constrained optimization problems," *Expert Systems with Applications*, p. 113033, 2019.

[22] A. Kumar, R. K. Misra, D. Singh, and S. Das, "Testing a multi-operator based differential evolution algorithm on the 100-digit challenge for single objective numerical optimization," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 34–40.

[23] S. Elsayed, R. Sarker, C. C. Coello, and T. Ray, "Adaptation of operators and continuous control parameters in differential evolution for constrained optimization," *Soft Computing*, vol. 22, no. 19, pp. 6595–6616, 2018.

[24] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Differential evolution with landscape-based operator selection for solving numerical optimization problems," in *Intelligent and evolutionary systems*. Springer, 2017, pp. 371–387.

[25] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 2014, pp. 1658–1665.

[26] S. Elsayed, R. Sarker, and C. C. Coello, "Enhanced multi-operator differential evolution for constrained optimization," in *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 4191–4198.

[27] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation*, vol. 13, no. 5, pp. 945–958, 2009.

[28] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[29] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

[30] H. J. Barbosa, H. S. Bernardino, and A. M. Barreto, "Using performance profiles for the analysis and design of benchmark experiments," in *Advances in Metaheuristics*. Springer, 2013, pp. 21–36.

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

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