Two-phase Differential Evolution Framework for Solving Optimization Problems

Karam M. Sallam, Saber M. Elsayed, Ruhul A. Sarker and Daryl L. Essam

School of Engineering and Information Technology
University of New South Wales at Canberra
Australia
Emails: karam.sallam@student.adfa.edu.au;{s.elsayed, r.sarker, d.essam}@adfa.edu.au

***Abstract*—Over the last two decades, different differential evolution (DE) variants have been successfully used to solve different optimization problems. However, no single DE algorithm has consistently been the best for solving a wide range of them. In the literature, this drawback has been tackled by using multiple DE operators in a single framework. However, utilizing a problem's landscape in the design of an efficient selection mechanism to emphasize the best-performing DE variant has not yet been thoroughly explored. Motivated by this fact, in this paper, a new two-phase (exploration and exploitation) multi-operator DE algorithm is proposed. It starts with the exploration phase, dynamically placing emphasis on the best-performing DE based on two landscape indicators and its performance history, and then repeats this process during the exploitation phase. To judge the performance of this algorithm, a variety of real-world optimization problems taken from different disciplines are solved. According to the results obtained, this algorithm shows superior performance to those of state-of-the-art algorithms.**

***Index Terms*—differential evolution, multi-operator, landscape indicators, real-world problems**

I. Introduction

Many real-world problems in different fields, such as operations research, biology, data mining and engineering design, can be formulated as optimization problems with one or more objective functions subject to some constraints.

Due to the importance of solving these problems, researchers and practitioners have proposed exact methods. However, as they usually encounter many challenges, such as (1) requiring specific mathematical properties, such as convexity, continuity and differentiability, to be used; and/or (2) maybe needing the problem to be simplified by making various assumptions for the convenience of mathematical modeling [1], their use in the majority of real-world cases is unverified. For these reasons, many evolutionary algorithms (EAs) have been proposed and implemented in order to tackle real-world optimization problems. However, although they do not require specific mathematical properties to be satisfied, are flexible to dynamic changes, can handle evaluating each solution in parallel, have the capability for self-organization and are more broadly applicable in practice, there is no guarantee that they can reach an optimal solution and the quality of their solutions depends on their parameter settings. Indeed, selecting an appropriate algorithm for solving an optimization problem is not an easy task.

The family of EAs contains various algorithms, such as DE [2], genetic algorithms (GAs) [3] and evolution strategy (ES) [4], with the major difference among them the ways in which they produce new solutions. Of EAs, DE has gained popularity for solving continuous optimization problems [5][6]. However, there is no guarantee that a DE algorithm, which performs well for one or a certain class of problems will work well for another or a range of problems. One reason for this is the variability of the underlying mathematical properties of optimization problems.

To overcome this limitation, multi-methods and/or multi-operator algorithms, in which more than one algorithm and/or search operator are used in a single framework with a mechanism for emphasizing the best-performing one during the search process [7], [8], have been developed. In the literature, there are different categories of these methods, such as ensemble-based [9][10], hyper-heuristics [11], mutli-methods [12][13], multi-operators [7] and heterogeneous [14] approaches. They all consist of a pool of different algorithms and/or operators, and use a selection mechanism to emphasize the best-performing algorithm and/or operator during the search process based on different criteria, such as a re-enforcement learning mechanism [15][16], improvement in the solution quality and/or constraint violations and/or feasibility rate [8], and convergence differences and progress ratios [17].

However, determining a way of combining these operators and/or methods is still challenging. Also, utilizing a problem's landscape to design an efficient selection mechanism has not been comprehensively explored. Although it has been noted that the pool of operators is always fixed during the evolutionary process, this may lead to consuming too much computational effort because of poorly performing DE operators. Therefore, in this paper, we investigate using landscape metrics to obtain information about the characteristics of problems, such as modality, separability, etc., and place emphasis on better-performing DE operators, and changing the pool of DE operators based on the optimization phase.

Consequently, a novel two-phase multi-operator DE with a landscape selection mechanism, referred to as TP-MODE, is proposed in this paper. In TP-MODE the whole evolutionary process is divided into two phases based on the number of fitness evaluations. In the first, a group of DE mutation strategies which have the capability to maintain diversity are chosen to generate trial vectors and, in the latter, other DE mutation strategies characterized by their ability to converge fast are chosen to produce offspring. Simultaneously, a landscape metric is used to place more emphasis on the best-performing operator in each phase. TP-MODE is tested on 22 benchmark test functions taken from the IEEE CEC2011 competition for solving real-world applications [18]. The experimental results imply that the performance of TPMODE is better than those of four other state-of-the-art DE variants.

The rest of this paper is organized as follows: in Section II, a review of DE algorithms and operators, and some landscape measures is presented. Section III describes the proposed framework; the simulation results for the benchmark problems and values of the parameters are provided in Section IV; and finally, Section V discusses the conclusions drawn from this study and possible future research directions.

II. DE and the Related Work

In this section, a literature review of DE and the concept of landscape analysis are discussed.

A. DE algorithm

Generally speaking, DE algorithm has shown good performance for solving optimization problems. It has three main operations (mutation, crossover, and selection) to evolve a population of individuals during the search process. DE starts with a uniform random population, such that

$$\overrightarrow{X}_{i,G} = x_{i,G}^{min} + (x_{i,G}^{max} - x_{i,G}^{min}) \times rand(1, D) \qquad (1)$$

where $i \in NP$, $j = 1, 2, ..., D$, NP is the number of individuals, D the dimension of the problem and $rand$ a random real number uniformly generated between 0 and 1, and G the generation number.

1) mutation: Before the mutation operator is applied, every vector $\overrightarrow{X}_{i,G}$ in the population is considered a parent vector (target vector), and corresponding to each parent vector, a mutant vector $\overrightarrow{V}_{i,G} = (v_{i,G}^1, v_{i,G}^2, ..., v_{i,G}^D)$ is produced by the weighted difference between two random vectors to a third vector (the base vector) from the current population (D is the number of decision variables), such that

$$\overrightarrow{V}_{i,G} = \overrightarrow{X}_{r_1^i,G} + F \times (\overrightarrow{X}_{r_2^i,G} - \overrightarrow{X}_{r_3^i,G}) \qquad (2)$$

where $r_1^i \neq r_2^i \neq r_3^i$ are random integer numbers selected from the range$[1, NP]$, which are all different from the index i, the scaling factor F is the control parameter for scaling the difference vectors and G the generation number.

2) crossover: Following the mutation operation, a trial vector $\overrightarrow{U}_{i,G} = \{u_{i,G}^1, u_{i,G}^2, ..., u_{i,G}^D\}$ is produced by crossing a parent vector and its corresponding trial vecto, such that

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & \text{if } (rand(0,1) \leq CR) \text{ or } (j = j_{rand}) \\ x_{i,G}^j & \text{otherwise} \end{cases} \qquad (3)$$

where $rand(0,1)$ a random real number uniformly generated between 0 and 1, CR the crossover control parameter and $j_{rand} \in [1, 2, ..., D]$ a randomly chosen index which ensures that the trail vector, $\overrightarrow{U}_{i,G}$ obtains at least one element from the mutant vector, $\overrightarrow{V}_{i,G}$.

3) selection: To decide whether parent $\overrightarrow{X}_{i,G}$ or child $\overrightarrow{U}_{i,G}$ vectors survive to the next generation a selection operation is employed. This operation is performed using

$$\overrightarrow{X}_{i,G+1} = \begin{cases} \overrightarrow{U}_{i,G} & \text{if } f(\overrightarrow{U}_{i,G}) \leq f(\overrightarrow{X}_{i,G}) \\ \overrightarrow{X}_{i,G} & \text{otherwise} \end{cases} \qquad (4)$$

where $f(\overrightarrow{U}_{i,G})$ and $f(\overrightarrow{X}_{i,G})$ are the objective functions of the child and parent, respectively. Similar to other EAs algorithms, DE repeats the aforementioned steps, generation after generation, to evolve its population of solutions until some specific termination condition is met.

B. Improved DE Algorithms

In this section, commonly used improved variants of DE are discussed.

As mentioned earlier, in DE the quality of the obtained solutions depends on its parameter (population size NP, crossover rate CR, and scaling factor F) settings. Thus, in the literature there are some different adaptation schemes that have been proposed, and we will review some of them in this section.

A fuzzy adaptive DE algorithm (FADE), in which fuzzy logic controllers were used to adapt the parameters F and CR, was proposed by Liu and Lampinen [19]. An adaptive DE algorithm (ADE) was proposed by Zaharie and Daniela [20], in which F and CR were adapted with regard to population diversity. A self-adaptive (jDE) strategy to adapt the F and CR values of the DE algorithm, was proposed by Breast et al. [21]. In jDE, both F and CR are also encoded into the individuals and their values survive to the next generation with a particular probability, τ_1 and τ_2, and otherwise, F and CR values are randomly re-initialized to new values within the given range for the next generation [22]. An adaptive DE algorithm with an optional external memory (JADE) was proposed by Zhang et al. [23], in which the control parameters CR and F of each individual were automatically updated according to previously successful experiences. Also, it implements a "DE/current-to-*p*best" mutation strategy, with an external archive, which some of the good solutions are stored on it to avoid premature convergence and also diversify the population. Tanabe and Fukunaga [24] introduced success-history based parameter adaptation for differential evolution (SHADE), which is an improved version of JADE, uses a history based parameter adaptation method. The L-SHADE [25] algorithm is a SHADE algorithm that uses linear population size reduction (LPSR) to dynamically re-size the population during a run. LPSR reduces the population linearly as the number of fitness evaluations increases. L-SHADE showed good performance, in comparison with other algorithm over a set of unconstrained optimization problems. A new Sinusoidal DE (SinDE), in which new sinusoidal formulas was used to automatically adjust the values of F and CR, was proposed by Draa et al. [26]. It was claimed by the author that, SinDE especially gives good results for multi-modal and composition functions. To dynamically select the a best compromise of parameters (i.e. F, CR and NP) for solving a specific problem in each run, Sarker et al. [27] designed a new mechanism to do it. The experimental results demonstrated its superior performance over the state-of-the-art DE variants.

To follow our discussion about DE algorithms and as mentioned earlier that no single DE algorithm was able to solve all kind of optimization problems, a considerable number of DE algorithms, which uses more than one mutation strategy or incorporated with other algorithms, have been proposed to solve this drawback and here we will review some of them.

Sallam et. al [5] proposed a neuro-dynamic differential evolution algorithm for solving CEC2015 single objective optimization problems. An adaptive mechanism was proposed for the appropriate use of L-SHADE and neuro-dynamic during the search process. A self-adaptive DE (SaDE) was proposed by Qin et al. [28] for solving unconstrained real-parameter optimization. In SaDE, both the trial vector generation strategy and its associated control parameter values, were gradually self-adapted by learning from their previous search experience. A composite DE (CoDE) was proposed by Wang et al. [29] for solving optimization problems. It uses three mutation strategies randomly with three fixed control parameter settings for generating a new trial vector at each generation. To generate a new solution, three vectors were generated, then the best one among them was selected to enter the next generation. From the experimental results, it was concluded that CoDE is a promising optimization algorithm for solving optimization problems. A self adaptive multi-operator differential evolution (SAMO-DE), for solving constrained optimization problems, was proposed by Elsayed et al. [8]. In SAMO-DE, each operator had its own sub-population which was evolved by a different DE operator. Based on an improvement measure in which the solution quality, constraint violation and feasibility ratio were used to calculate the success of each operator, the number of individuals in each sub-population was adaptively updated and more emphasis was given to the operator with the highest success. The results showed that SAMO-DE performed better than other-state-of-the-art algorithms. Using a mix of four different DE mutation strategies, two crossover operators and two constraint handling techniques for constrained optimization problems was proposed by Elsayed et al. [30]. The experimental results show that the proposed algorithm is better than other state-of-the-art algorithms. Mallipeddi et al. [9] designed the ensemble differential evolution algorithms (EPSDE). In these algorithms, a pool of distinct DE mutation strategies, crossover strategies and a pool of associated values for the control parameters not only coexist throughout the evolutionary process, but also compete with each other to produce offspring. All of the above mentioned methods did not incorporate any landscape information in the selection phase. For more inforatiomation regarding DE algorithms, readers are refered to [31] [32].

In the recent past, researchers and practitioners have used fitness landscape to determine and select an appropriate algorithm or operator for solving optimization problems. In [33], the authors suggested using a cost sensitive learning model to select the best algorithm amongst a set of four algorithms. The four algorithms were selected manually from the algorithms solving Black-Box Optimization Benchmarking (BBOB) in 2009 and 2010 [34][35]. The functions in $10D$ were characterized using 19 measurements extracted by the exploratory landscape analysis (ELA) technique. In their work, low-level features [36] were obtained by systematic sampling of the functions on the search space. Then separability, modality, and global structures of the optimization problem were measured as the first step to characterize the landscape (this step was done offline). Next, a machine learning model was constructed to select the best algorithm from the portfolio. Based on two different cross validation schemes, the model was validated. However, the results may not be general for a knowledge base with problems of different dimensionality such as $2D$, $3D$, $5D$, and $20D$. Also, because the low level features were extracted in a different step, the authors did not add the computational cost for calculating the features to the function evaluations. Further, as the selection of the four algorithms was done manually, this weakened its validation on unobserved problems. In [37], decision trees were employed to predict the failure of seven different particle swarm optimization algorithms (PSO), by using a number of fitness landscape metrics. In [38], an adaptive operator selection mechanism, based on a set of four fitness landscape analysis techniques, was used to train an online regression learning model (dynamic weighted majority), which was used to predict the weight of each operator in each generation. Their proposed mechanism was used to determine the most suitable crossover operator, among four crossover operators, to solve a set of Capacitated Arc Routing Problem (CARP) instances. The authors used instantaneous reward, in which the reward was considered as the value computed at the last evaluation. In comparison with some of the-state-of-the-art algorithms, the algorithm did not show significant benefit.

III. Two-phase multi-operator DE

In this section, the two-phase multi-operator DE with a landscape selection mechanism is presented.

A. *TP-MODE*

In designing the proposed algorithm, two considerations taken into account are that, (1) as no single mutation strategy might be able to solve all kinds of problems, it is beneficial to use a multi-operator [8], [13], and (2) the new algorithm should exhibit an exploration capability in the early phase of the evolution and fast convergence in a later one.

The framework of TP-MODE is presented in Algorithm 1.

Generally speaking, the entire evolutionary process is divided into two phases, exploration and exploitation. In the first, TP-MODE focuses on improving its exploration by using two DE mutation strategies which maintain diversity and, in the second, two other DE strategies which can converge quickly are used. Initially, NP individuals are randomly generated by equation 15. Subsequently, based on the type of phase, two mutation strategies, in which each operator is randomly assigned the same number of individuals (n_{op}, $\forall op = \{1, 2\}$), are implemented. Then, a new solution is generated using its assigned DE variant and evaluated according to the fitness function value. If it is better than its parent, it survives to the next generation; otherwise its parent is retained for the next generation. At the same time, two landscape metrics (the information landscape negative searchability metric/ searchability indicator (SI), fitness distance correlation (FDC)) and one performance measure (success rate (SR)) are calculated for each operator, with the overall normalized measure computed from these values using equation (13) based on which, n_{op}, $\forall op = \{1, 2\}$ is updated. As this process may abandon certain operators which could be useful in later stages of the evolutionary process, we set a minimum number of individuals for each operator. The above process is repeated until the maximum number of fitness function evaluations (MAX_{FES}) is reached. In this paper, the algorithm switches from the first to second phase based on a predefined number of FEs ($\frac{MAX_{FES}}{3}$).

In the following subsections, TP-MODE's components are discussed in more detail.

1) Two-phase mutation strategies:: As, in the first phase, the main objective is to maintain exploration capability of TP-MODE to avoid premature convergence, DE/rand/1/bin is chosen to be placed in its pool while the second variant is DE/ϕbest/1, where ϕ is set to $[1, 0.6 * NP]$, as described in [39].

On the other hand as, in the second phase, the aim is to speed up convergence; information from the best solution is of great interest. Therefore, DE/current-to-ϕbest/1/bin/with archive and DE/ϕbest/1 are used, withϕ is equal to 0.1 for both operators.

2) Selection mechanism: As previously mentioned, in each phase, two operators are used, with TP-MODE placing emphasis on the best-performing one based on three indicators: (1) the SR; (2) SI; and (3) FDC.

- SR
  The SR of each operator (SR_{op}) is defined as the number of successful offspring generated by a search operator (op), divided by the number of individuals assigned to op as :

$$SR_{op} = \frac{\text{Number of successful offsprings}}{n_{op}}, \forall op = \{1,\ 2\} \tag{5}$$

Then, the normalized value for the SR is calculated using:

$$NM_{SR_{op}} = \frac{\overline{SR}_{op}}{\sum_{OP=1}^{m} \overline{SR}_{op}} \tag{6}$$

where the $\overline{SR}_{op}$ is the mean value of the SR of operator op.

- SI
  The searchability of the algorithm, which is the capability of its search operator to move to a region of the search space with a better fitness value, is measured by computing an information landscape metric based on the difference between the information landscape vector of the problem to be solved and a reference landscape vector. The reference landscape is the landscape of a function tthat can be easily optimized by any optimization algorithm in any dimension [40] and the reference function $f_{ref}(\overrightarrow{x})$ is determined using:

$$f_{ref}(\overrightarrow{x}) = \sum_{j=1}^{D} (x_j - x_j^*)^2 \tag{7}$$

where $\overrightarrow{x}_i^*$ is the best individual in the sample.
An information matrix for a minimization problem $M = [a_{i,j}]$ is constructed using:

$$a_{i,j} = \begin{cases} 1 & \text{if } f(x_i) < f(x_j) \\ 0.5 & \text{if } f(x_i) = f(x_j) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Not all the entries in the information landscape are required to define it [40] [41]. There are duplicates in the entries due to symmetry (the lower triangle should be omitted), the entries on the diagonal are always 0.5 (and should be omitted), and the row and column of the optimum solution should also be omitted. Therefore, the information matrix can be reduced to a vector $LS = (ls1, ls_2, ..., ls_{|LS|})$, where the number of elements in LS is: $|LS| = \frac{(NP-1)\times(NP-2)}{2}$.
After constructing the landscape vector of the problem to be optimized (LS_f) and the vector landscape of the reference function (LS_{ref}), the SI is computed as:

$$LD_{op} = \frac{1}{|LS_f|} \times \sum_{i=1}^{|LS_f|} |ls_{1i} - ls_{2i}| \tag{9}$$

When LD is close to 0, the problem is considered easy and difficult when LD is close to 1.
The normalized value for the LD is then calculated as:

$$NM_{LD_{op}} = \frac{(1 - M_{LD_{OP}})}{\sum_{OP=1}^{m}(1 - M_{LD_{OP}})} \tag{10}$$

where M_{LD} is the mean value of information landscape.

- FDC
  The FDC, which was proposed by Jones and Forrest [42] and measures the correlation between the objective value

and distance to the nearest optimum in the search domain is another method to determine a problem's difficulty [43].
Given a set of solutions $X = \{\overrightarrow{x}_1, \overrightarrow{x}_2, ..., \overrightarrow{x}_{NP}\}$ and their objective function values $F = \{f_1, f_2, ..., f_{NP}\}$, the FDC value is computed by:

$$FDC_{op} = \frac{\sum_{i=1}^{NP}(f_i - \bar{f})_{op} \times (d_i - \bar{d})_{op}}{n \times (\sigma_f \times \sigma_d)_{op}} \tag{11}$$

where d_i is the distance between the i^{th} individual and the best solution in the population, $\bar{f}$, $\bar{d}$, σ_f, and σ_d are the mean and standard deviations of fitness function and distance, respectively.
For a minimization problem, a value of FDC_{op} closes to 1 means that the problem is relatively easy, and one near 0 that it is difficult.
The normalized FDC (NM_{FDC}) is calculated using equation by:

$$NM_{FDC_{op}} = \frac{\overline{FDC}_{op}}{\sum_{OP=1}^{m} \overline{FDC}_{op}} \tag{12}$$

where the $\overline{FDC}_{op}$ is the mean value of FDC of operator *op*.
Subsequently, the overall normalized value for each operator is computed using:

$$ONM_{op,G} = \frac{NM_{SR_{op}} + NM_{LD_{op}} + NM_{FDC_{op}}}{\sum_{op=1}^{m}(NM_{SR_{op}} + NM_{LD_{op}} + NM_{FDC_{op}})}, \tag{13}$$

After determining the overall normalized value for each operator, the number of individuals for each operator in each generation, is calculated by:

$$n_{op,G} = MSS + \frac{ONM_{op,G}}{\sum_{op=1}^{m} ONM_{op,G}} \times (NP - MSS \times m) \tag{14}$$

where $n_{op,G}$ is the number of individuals operator *op* will evolve at generation G, MSS is the minimum number of individuals for operator *op* at generation G.

3) Population Initialization and Updating Method : In the initialization phase, the Latin Hypercube Design, which is a type of stratified sampling, is used to generate the initial population because of its capability to generate a sample of points thatmore efficiently covers the whole search region .

The initialization is performed using:

$$\begin{gathered} x_{i,j} = x_i^{min} + (x_i^{max} - x_i^{min}) \times lhd(1, NP) \\ i \in NP \, and \, j = 1, 2, ..., D \end{gathered} \tag{15}$$

where lhd is a function that generates random numbers using the Latin Hypercube Design.

Moreover, a linear population size reduction schema is used to adaptively re-size the NP during the evolutionary process [25], as follows:

$$NP_{t+1} = round[(\frac{NP^{min} - NP^{init}}{MAX_{FES}}) \times FES + NP^{init}] \tag{16}$$

where NP^{min} is the smallest number of individuals that the proposed algorithm can use. FES is the current number of fitness evaluations, MAX_{FES} is the maximum number of fitness evaluations.

4) Managing F and CR: As it is a fact that the performance of the DE algorithm is affected by the values of the control parameters (F and CR) in each phase [44] [45], we use a different adaptation mechanism for their values. In the first, the mechanism proposed by Elsayed et al. [46] is adopted as it has shown its capability to maintain diversity.

Initially, for each individual in the population, two sets of control parameters, $F \in N(0.5, 15)$ and $CR \in (0.5, 0.15)$, are generated using a Gaussian distribution with mean and standard deviation values of 0.5 and 0.15, respectively. Then, as in [7], to generate new offspring, F and CR are calculated, respectively,:

$$\begin{gathered} F = F_{r1} + N(0, 0.5) \times (F_{r2} - F_{r3}) + N(0, 0.5) \\ (F_{r4} - F_{r5}) + N(0, 0.5) \times (F_{r6} - F_{r7}) \end{gathered} \tag{17}$$

$$\begin{gathered} CR = CR_{r1} + N(0, 0.5) \times (CR_{r2} - CR_{r3}) + N(0, 0.5) \\ (CR_{r4} - CR_{r5}) + N(0, 0.5) \times (CR_{r6} - CR_{r7}) \end{gathered} \tag{18}$$

where $r_1^i \neq r_2^i \neq r_3^i \neq r_4^i \neq r_5^i \neq r_6^i \neq r_7^i$ are random integer numbers selected from the range$[1, NP]$.

If the value of F is less than 0.1 or larger than 1, it is truncated to 0.1 and 1, respectively, while if the value of CR is less than 0.01 or larger than 1, it is truncated to 0.01 and 1, respectively.

In the second phase, inspired by [47], a scaling factor pool (i.e., F_{pool} = [0.6, 0.8, 1.0]) and a crossover control parameter pool (i.e., CR_{pool} = [0.1, 0.2, 1.0]) are established in DE. Then, at each generation, F and CR are randomly chosen from F_{pool} and CR_{pool}, respectively.

IV. Experimental results

In this section, the performances of the proposed algorithm for solving a benchmark test set of 22 problems taken from the CEC2011 competition considering real-world applications is discussed [18]. The proposed algorithm was run following the guidelines of the competition that required 25 independent runs for each test problem with up to $FES_{MAX} = 150,000$ fitness evaluations. It was coded using Matlab R2014a and run on a PC with a 3.4 GHz Core I7, 16 GB RAM, and windows 7.

Algorithm 1 Proposed framework

```
1:  Generate an initial population $(X)$ of size $NP$ using Latin Hypercube Design; $FES \leftarrow 0$;
2:  Evaluae $X$;
3:  $FES \leftarrow FES + NP$;
4:  $n_{op_1}$=$n_{op_2}$=$\frac{NP}{2}$;
5:  while $FES \leq MAX_{FES}$ do
6:      if $FES \leq limit$ then
7:          {Phase1}
8:          Set operators $op_1 = DE/\phi best/1$; $op_2 = DE/rand/2$;
9:          Calculate $F$ and $CR$ using equations (17), and (18);
10:     else
11:         {Phase2}
12:         Set operators $op_1 = DE/current - to - \phi best/1/Archive$; $op_2 = DE/\phi best/1$;
13:         Calculate control parameters $F$ and $CR$ as in the last paragraph of Sec III-A4 ;
14:     end if
15:     Evolve individuals using its assigned operators $op$;
16:     Compute $NM_{SR_{op}}$, $NM_{LD_{op}}$, and $NM_{FDC_{op}}$ using equations (6), (10), and (11);
17:     Update $n_{op_1}$ and $n_{op_2}$ using equation (14);
18:     $FES \leftarrow FES + NP$;
19:     if $FES == limit$ then
20:         $n_{op_1}$=$n_{op_2}$=$\frac{NP}{2}$;
21:     end if
22:     Update the $NP$ using equation (16);
23: end while
```

A. Algorithm Parameters and Operators

The default values of NP^{init} of 200 and NP^{min} of 10 were set based on our empirical analysis, with φ set at a value of 0.6 for DE/φbest/1 to maintain diversity, and 0.1 for the other two variants, to speed up the convergence rate. For DE/current-to-ϕbest/1/bin/with archive the archive rate (A) was set at a value of 1.4, $limit$ the maximum limit for running phase 1, after which phase two started, and set at a value of $\frac{FES_{MAX}}{3}$ was set based on our empirical analysis and this parameter was fixed for all problems.

B. Detailed Results of proposed algorithm

The detailed results (best, worst, median, average, and standard deviation (Std.)) are shown in Table I. The performance of TP-MODE was compared with those of three algorithms, GA-MPC [48] (the winner of the CEC2011 competition), SAMO-DE [7], an ensemble DE algorithm (EPSDE) [9], that uses different strategies, and SHADE [24] . All Algorithms used the same stopping criteria; i.e., $150,000$ fitness function evaluations.

Table II presents the best, mean and standard deviation values of these algorithms. Based on the best fitness values obtained, TP-MODE was better than GA-MPC, SAMO-DE and EPSDE for 7, 12 and 14 problems, respectively, and the same for 9, 7 and 3, respectively, while it was inferior for 6, 3 and 5, respectively. Regarding the average fitness values, TP-MODE was superior to GA-MPC, SAMO-DE and EPSDE for 10, 15 and 15 problems, respectively, obtained the same mean results for only 4, 3 and 2, respectively, and was inferior for 8, 4 and 5, respectively.

As it is also possible to study the statistical difference between any two algorithms using a non-parametric test, the Wilcoxon Signed Rank Test [49] was performed, with the results regarding the best and average fitness values presented in Table III. As a null hypothesis, it was assumed that there was no significant difference between the best and/or mean values of two samples while an/the alternative hypothesis was that there was a significant difference at a 10% significance level. Based on the test results, we assigned one of three signs ($+$, $-$, and $\approx$) for the comparison of any two algorithms (shown in the last column), where the "$+$" sign means the first algorithm was significantly better than the second, the "$-$" sign means that the first algorithm was significantly worse, and the "$\approx$" sign means that there was no significant difference between thim. It is clear that, from Table III that TP-MODE was superior to GA-MPC, SAMO-DE, EPSDE, and SHADE in terms of the average results and very competitive with them regarding the best.

Finally, the Friedman Test was conducted to rank all algorithms, with the results shown in Table IV in which it is clear that TP-MODE was superior to the other three algorithms regarding both the best and average fitness results, followed by GA-MPC, SHADE, EPSDE and SAMODE, respectively.

V. Conclusion and future work

In this study, a new multi-operator DE algorithm for solving real-world application problems was proposed. It focused on diversity without losing convergence through dividing the whole search process into two phases, with two DE mutation operators used in each phase to achieve the aim of the corresponding phase. Furthermore, a new measure based on the quality of solutions and characteristics of the landscape was used to emphasize the best-performing operator in each phase. The performance of the proposed algorithm was tested on 22 real-world application problems taken from the CEC2011 competition, with the results demonstrating its superiority over three other state-of-the-art algorithms.

Possible extensions of this work include obtaining a dynamic balance between convergence and diversity, and analyzing each component in the algorithm's design. Also, using more than one EA may be of great interest.

References

[1] R. Sarker, J. Kamruzzaman, and C. Newton, "Evolutionary optimization (evopt): a brief review and analysis," *International Journal of Computational Intelligence and Applications*, vol. 3, no. 04, pp. 311–330, 2003.

[2] R. Storn and K. Price, "Differential evolution a simple and efficient adaptive scheme for global optimization over continuous spaces, international computer science institute, berkeley," *Berkeley, CA*, 1995.

[3] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[4] I. Rechenberg, "Evolution strategy," *Computational Intelligence: Imitating Life*, vol. 1, 1994.

Table I

THE FUNCTION VALUES OBTAINED BY TP-MODE OVER 25 RUNS AND 150000 FITNESS FUNCTION EVALUATIONS

Problems	Best	Worst	Median	Mean	Std.
P01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
P02	-2.842253E+01	-2.163764E+01	-2.589359E+01	-2.573171E+01	1.594532E+00
P03	1.151489E-05	1.151489E-05	1.151489E-05	1.151489E-05	0.000000E+00
P04	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
P05.1	-3.684537E+01	-3.163659E+01	-3.479658E+01	-3.465468E+01	1.119970E+00
P05.2	-2.916612E+01	-2.300456E+01	-2.916612E+01	-2.822079E+01	1.385154E+00
P06	5.000000E-01	1.096765E+00	7.510147E-01	7.603013E-01	1.655763E-01
P07	2.200000E+02	2.200000E+02	2.200000E+02	2.200000E+02	0.000000E+00
P08	1.628873E+03	3.059153E+03	2.442968E+03	2.453615E+03	3.755708E+02
P09	-2.184235E+01	-2.164293E+01	-2.164414E+01	-2.165958E+01	5.411104E-02
P10.1	5.116130E+04	5.247562E+04	5.175826E+04	5.180184E+04	3.431911E+02
P10.2	1.068335E+06	1.076211E+06	1.072737E+06	1.072583E+06	1.572728E+03
P11.1	1.544419E+04	1.544419E+04	1.544419E+04	1.544419E+04	1.897504E-05
P11.2	1.812350E+04	1.867086E+04	1.833903E+04	1.835748E+04	1.507295E+02
P11.3	3.271559E+04	3.278829E+04	3.276597E+04	3.276269E+04	1.931411E+01
P11.4	1.259621E+05	1.295006E+05	1.279397E+05	1.278757E+05	1.111803E+03
P11.5	1.881031E+06	1.941075E+06	1.913161E+06	1.913606E+06	1.427120E+04
P12.1	9.362562E+05	9.420340E+05	9.392626E+05	9.392934E+05	1.544130E+03
P12.2	9.400879E+05	1.068943E+06	9.697023E+05	9.788815E+05	3.926205E+04
P12.3	9.362562E+05	9.420340E+05	9.392626E+05	9.392934E+05	1.544130E+03
P13	9.875190E+00	1.797719E+01	1.409832E+01	1.412068E+01	1.616015E+00
P14	8.593443E+00	1.877439E+01	9.827477E+00	1.147226E+01	3.270544E+00

Table II

THE FUNCTION VALUES ACHIVED BY TP-MODE, GA-MPC, SAMO-DE AND EPSDE OVER 25 RUNS AND 150000 FES

	TP-MODE			GA-MPC			SAMO-DE			EPSDE			SHADE		
Problems	Best	Mean	Std.	Best	Mean	Std.	Best	Mean	Std.	Best	Mean	Std.	Best	Mean	Std.
P01	0.00000E+00	0.000000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	1.21203E+00	3.37622E+00	0.00000E+00	1.78000E+00	4.18000E+00	2.73700E-02	7.59030E-01	2.29845E+00
P02	-2.84225E+01	-2.573171E+01	1.59453E+00	-2.84225E+01	-2.77007E+01	4.67305E-01	-2.84225E+01	-2.70698E+01	6.62481E-01	-2.14000E+01	-1.83000E+01	1.65000E+00	-2.44943E+01	-2.30316E+01	7.46050E-01
P03	1.15149E-05	1.151489E-05	0.00000E+00	1.15149E-05	1.15149E-05	0.00000E+00	1.15149E-05	1.15149E-05	0.00000E+00	1.15149E-05	1.15149E-05	1.52000E-19	1.15000E-05	1.15000E-05	5.19000E-21
P04	0.00000E+00	0.000000E+00	0.00000E+00	1.37708E+01	1.38154E+01	1.54600E-01	1.37708E+01	1.39407E+01	2.50222E-01	1.38000E+01	1.67000E+01	3.26000E+00	0.00000E+00	0.00000E+00	0.00000E+00
P05.1	-3.68454E+01	-3.465468E+01	1.11997E+00	-3.68454E+01	-3.50388E+01	8.32925E-01	-3.68439E+01	-3.35947E+01	1.57514E+00	-3.20000E+01	-2.90000E+01	1.84000E+00	-3.66566E+01	-3.60052E+01	6.38070E-01
P05.2	-2.91661E+01	-2.822079E+01	1.38515E+00	-2.91661E+01	-2.74881E+01	1.78214E+00	-2.91661E+01	-2.76347E+01	1.92353E+00	-1.99000E+01	-1.70000E+01	2.68000E+00	-2.91420E+01	-2.90486E+01	1.23890E-01
P06	5.00000E-01	7.603013E-01	1.65576E-01	5.00000E-01	7.48409E-01	1.24914E-01	5.00000E-01	8.16624E-01	1.19367E-01	1.28000E+00	1.42000E+00	7.38000E-02	9.06410E-01	1.12130E+00	8.34400E-02
P07	2.20000E+02	2.200000E+02	0.00000E+00	2.20000E+02	2.20000E+02	0.00000E+00	2.20000E+02	2.20000E+02	0.00000E+00	2.20000E+02	2.20000E+02	0.00000E+00	2.20000E+02	2.20000E+02	0.00000E+00
P08	1.62887E+03	2.453615E+03	3.75571E+02	4.66763E+02	1.22059E+03	3.61119E+02	9.44119E+02	2.26440E+03	8.54213E+02	7.84500E+02	2.52900E+03	1.32800E+03	1.14114E+03	2.22875E+03	7.14195E+02
P09	-2.18424E+01	-2.165958E+01	5.41110E-02	-2.18425E+01	-2.17022E+01	1.16347E-01	-2.18217E+01	-2.16589E+01	1.12958E-01	-2.18000E+01	-1.56000E+01	3.79000E+00	-2.18425E+01	-2.16289E+01	1.40740E-01
P10.1	5.11613E+04	5.180184E+04	3.43191E+02	5.09251E+04	5.20546E+04	4.49912E+02	5.12682E+04	5.23475E+04	5.78021E+02	5.11000E+04	5.22000E+04	7.24000E+02	5.15596E+04	5.32252E+04	3.80380E+03
P10.2	1.06834E+06	1.072583E+06	1.57273E+03	1.06955E+06	1.07338E+06	1.61759E+03	1.07021E+06	1.07313E+06	1.70325E+03	1.06000E+06	1.07000E+06	2.13000E+03	1.07000E+06	1.10000E+06	5.34206E+04
P11.1	1.54442E+04	1.544419E+04	1.89750E-05	1.54442E+04	1.54442E+04	1.75112E-07	1.54442E+04	1.54442E+04	7.24128E-04	1.54000E+04	1.55000E+04	1.55000E+01	1.54442E+04	1.54442E+04	5.57000E-12
P11.2	1.81235E+04	1.835748E+04	1.50730E+02	1.81006E+04	1.82610E+04	6.98163E+01	1.82843E+04	1.85249E+04	1.34481E+02	1.81000E+04	1.81000E+04	4.39000E+01	1.80286E+04	1.81307E+04	5.17471E+01
P11.3	3.27156E+04	3.276269E+04	1.93141E+01	3.27238E+04	3.27698E+04	2.68249E+01	3.27840E+04	3.28462E+04	3.45006E+01	3.26000E+04	3.27000E+04	3.59000E+01	3.27429E+04	3.27600E+04	2.80246E+01
P11.4	1.25962E+05	1.278757E+05	1.11180E+03	1.29213E+05	1.33230E+05	1.87880E+03	1.30037E+05	1.32707E+05	1.14873E+03	1.28000E+05	1.31000E+05	2.47000E+03	1.26951E+05	1.29231E+05	1.58563E+03
P11.5	1.88103E+06	1.913606E+06	1.42712E+04	1.92026E+06	1.95332E+06	1.40836E+04	1.91925E+06	1.97709E+06	3.53818E+04	1.91000E+06	1.92000E+06	1.18000E+04	1.88000E+08	1.91000E+06	1.40754E+04
P12.1	9.36256E+05	9.392934E+05	1.54413E+03	9.49500E+05	9.71289E+05	1.03874E+04	9.43215E+05	9.48921E+05	3.91430E+03	9.39000E+05	9.43000E+05	2.63000E+03	9.37267E+05	9.40475E+05	2.16143E+03
P12.2	9.40088E+05	9.788815E+05	3.92621E+04	9.72102E+05	1.05630E+06	5.70416E+04	1.00815E+06	1.20594E+06	9.95328E+04	9.40000E+05	9.90000E+05	4.14000E+04	9.39771E+05	9.52462E+05	1.21308E+04
P12.3	9.36256E+05	9.392934E+05	1.54413E+03	9.46598E+05	9.75109E+05	1.18489E+04	9.47654E+05	9.58792E+05	5.99265E+03	9.39000E+05	9.43000E+05	2.63000E+03	9.34264E+05	9.40667E+05	2.99161E+03
P13	9.87519E+00	1.412068E+01	1.61602E+00	7.09556E+00	1.28182E+01	3.24134E+00	6.94322E+00	1.10675E+01	2.65228E+00	1.66000E+01	1.88000E+01	1.67000E+00	1.41104E+01	1.75903E+01	1.42842E+00
P14	8.59344E+00	1.147226E+01	3.27054E+00	8.39869E+00	9.35934E+00	9.45433E-01	8.61063E+00	1.09952E+01	2.38898E+00	8.78000E+00	1.39000E+01	4.08000E+00	1.31146E+01	1.99439E+01	2.87591E+00

Table III

COMPARISON SUMMARY BETWEEN TP-MODE AND OTHER STATE-OF-THE-ART ALGORITHMS

Criteria	Algorithms	better	equal	worse	Dec.
Best	TP-MODE vs. GA-MPC	8	8	6	≈
	TP-MODE vs. SAMO-DE	13	7	2	+
	TP-MODE vs. EPSDE	12	3	7	≈
	SHADE	11	8	3	≈
Mean	TP-MODE vs. GA-MPC	10	4	8	+
	TP-MODE vs. SAMO-DE	15	3	4	+
	TP-MODE vs. EPSDE	17	2	3	+
	SHADE	11	6	5	≈

Table IV

FRIEDMAN'S TEST RESULTS

	Best results	Mean results	Overall mean rank	Order
Algorithm	rank	rank		
TP-MODE	**2.43**	**2.32**	**2.38**	**1**
GA-MPC	2.73	2.84	2.79	2
SAMO-DE	3.50	3.50	3.50	5
EPSDE	3.18	3.70	3.44	4
SHADE	3.16	2.64	2.90	3

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

[6] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, 2011.

[7] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Differential evolution with multiple strategies for solving cec2011 real-world numerical optimization problems," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 1041–1048.

[8] 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.

[9] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, pp. 1679–1696, 2011.

[10] G. Iacca, F. Caraffini, and F. Neri, "Multi-strategy coevolving aging particle optimization," *International journal of neural systems*, vol. 24, no. 01, p. 1450008, 2014.

[11] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[12] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 243–259, 2009.

[13] S. M. Elsayed, R. A. Sarker, D. L. Essam, and N. M. Hamza, "Testing united multi-operator evolutionary algorithms on the cec2014 real-parameter numerical optimization," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 1650–1657.

[14] F. V. Nepomuceno and A. P. Engelbrecht, "A self-adaptive heterogeneous pso for real-parameter optimization," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 361–368.

[15] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[16] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 1, pp. 114–130, 2014.

[17] L.-A. Gordián-Rivera and E. Mezura-Montes, "A combination of specialized differential evolution variants for constrained optimization," in *Advances in Artificial Intelligence–IBERAMIA 2012*. Springer, 2012, pp. 261–270.

[18] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems," *Jadavpur University, Nanyang Technological University, Kolkata*, 2010.

[19] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[20] D. Zaharie, "Parameter adaptation in differential evolution by controlling the population diversity," in *Proceedings of the international workshop on symbolic and numeric algorithms for scientific computing*, 2002, pp. 385–397.

[21] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE transactions on evolutionary computation*, vol. 10, no. 6, pp. 646–657, 2006.

[22] A. Zamuda and J. Brest, "Self-adaptive control parameters randomization frequency and propagations in differential evolution," *Swarm and Evolutionary Computation*, vol. 25, pp. 72–99, 2015.

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

[24] R. Tanabe and A. Fukunaga, "Evaluating the performance of shade on cec 2013 benchmark problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 1952–1959.

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

[26] A. Draa, S. Bouzoubia, and I. Boukhalfa, "A sinusoidal differential evolution algorithm for numerical optimisation," *Applied Soft Computing*, vol. 27, pp. 99–126, 2015.

[27] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 689–707, 2014.

[28] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 398–417, 2009.

[29] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 55–66, 2011.

[30] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A self-adaptive combined strategies algorithm for constrained optimization using differential evolution," *Applied Mathematics and Computation*, vol. 241, pp. 267–282, 2014.

[31] 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.

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

[33] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß, "Algorithm selection based on exploratory landscape analysis and cost-sensitive learning," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012, pp. 313–320.

[34] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," 2009.

[35] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2010: Experimental setup," 2010.

[36] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory landscape analysis," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 829–836.

[37] K. M. Malan and A. P. Engelbrecht, "Particle swarm optimisation failure prediction based on fitness landscape characteristics," in *Swarm Intelligence (SIS), 2014 IEEE Symposium on*. IEEE, 2014, pp. 1–9.

[38] P. A. Consoli, L. L. Minku, and X. Yao, "Dynamic selection of evolutionary algorithm operators based on online learning and fitness landscape metrics," in *Simulated Evolution and Learning*. Springer, 2014, pp. 359–370.

[39] S. Elsayed and R. Sarker, "Differential evolution framework for big data optimization," *Memetic Computing*, vol. 8, no. 1, pp. 17–33, 2016.

[40] Y. Borenstein and R. Poli, "Information landscapes," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 1515–1522.

[41] Y. Borenstein and R. Poli, "Decomposition of fitness functions in random heuristic search," in *Foundations of Genetic Algorithms*. Springer, 2007, pp. 123–137.

[42] T. Jones, S. Forrest *et al.*, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms." in *ICGA*, vol. 95, 1995, pp. 184–192.

[43] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue, "A study of fitness distance correlation as a difficulty measure in genetic programming," *Evolutionary Computation*, vol. 13, no. 2, pp. 213–239, 2005.

[44] M. Weber, V. Tirronen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Computing*, vol. 14, no. 11, pp. 1187–1207, 2010.

[45] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges." *IEEE Trans. Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, 2015.

[46] 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.

[47] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 55–66, 2011.

[48] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Ga with a new multi-parent crossover for solving ieee-cec2011 competition problems," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 1034–1040.

[49] B. Rosner, R. J. Glynn, and M.-L. T. Lee, "The wilcoxon signed rank test for paired comparisons of clustered data," *Biometrics*, vol. 62, no. 1, pp. 185–192, 2006.