MDE: Differential Evolution with Merit-based Mutation Strategy

Amin Ibrahim[1], Shahryar Rahnamayan[1], Miguel Vargas Martin[2]
University of Ontario Institute of Technology, Oshawa, ON, Canada
amin.ibrahim@uoit.ca, shahryar.rahnamayan@uoit.ca, miguel.vargasmartin@uoit.ca
[1]Faculty of Electrical, Computer, and Software Engineering, [2]Faculty of Business and IT

Abstract— **Currently Differential Evolution (DE) is arguably the most powerful and widely used stochastic population-based real-parameter optimization algorithm. There have been variant DE-based algorithms in the literature since its introduction in 1995. This paper proposes a novel merit-based mutation strategy for DE (MDE); it is based on the performance of each individual in the past and current generations to improve the solution accuracy. MDE is compared with three commonly used mutation strategies on 28 standard numerical benchmark functions introduced in the IEEE Congress on Evolutionary Computation (CEC-2013) special session on real parameter optimization. Experimental results confirm that MDE outperforms the classical DE mutation strategies for most of the test problems in terms of convergence speed and solution accuracy.**

Keywords – **Global optimization, Differential evolution, Evolutionary algorithms, P-metaheuristics, Merit-based selection.**

I. Introduction

NOWADAYS, real-world applications are increasingly complex and more encompassing, in the sense that more decision variables are used to model complex situations and more input data and parameters are available to capture the complexity of the problems. Since finding exact solutions in these applications still poses a real challenge despite the impact of recent advances in computer technology, there are numerous metaheuristics capable of finding "good" solutions in a "reasonable" time. Due to the inherent complexities and dynamics we have in nature, and its ability to tackle with its own problems, nature is the main source of inspiration for solving our complex problems in science and engineering [1].

Metaheuristics are high-level strategies for exploring search spaces by using variant search methods. Their main goal is efficiently exploring the search space in order to find optimal (or near to optimal) solutions in reasonable time. They solve problems which are "hard" to solve by exhaustive exploring. Metaheuristics have been used in many applications ranging from software engineering, energy systems design, bioinformatics, telecommunication, finance and others. A description of well- known metaheuristics can be found in [2, 3].

Metaheuristics can be divided into two main categories, namely, population-based metaheuristics (P-metaheuristics) and single solution-based metaheuristics (S-metaheuristics). Basically, they differ by the number of tentative candidate solutions which are involved in every iteration. S-metaheuristics start with a single initial solution which is replaced by a more accurate solution at every iteration. These types of optimization methods offer strong local search properties known as exploitation properties - however, they get trapped by local optima.

In contrast, P-metaheuristics use an entire set of candidate solutions called population which are improved at every iteration. The first step in P-metaheuristics is the initialization of the population. Next is the generation of current population. Then, new population is selected from previous and current population, based on their corresponding fitness values. Finally, this process is repeated until stopping criteria are met. The main advantage of P-metaheuristics is that the diversification of the population aids the search properties known as exploration properties.

Almost all P-metaheuristics are nature-inspired, in which multiple agents interact to solve or accomplish a given task. Though arguably nature-inspired algorithms are still at their early stages, many have shown a great potential in solving very complicated problems with diverse applications in engineering, business, economics, and communication networks. For example, EAs are nature-inspired population-based methods taken from the biological evolution of living organisms to adapt to their ecosystem.

The core component of any EA is the selection strategy [4]. The selection strategy determines which parents are selected for mating (reproduction) in the hope that they would generate better offspring (i.e. individuals with a bias toward better fitness). There are mainly four selection strategies in EA: roulette wheel selection, random, rank-based, and tournament-based selection. The roulette wheel selection assigns each individual a selection probability that is proportional to its relative fitness and as a result the fittest individuals will introduce a bias that may cause a premature convergence and a loss of diversity. Moreover, if the objective is minimization rather than maximization, a transformation is required. When all individuals are equally fit, this strategy is similar to random selection. The tournament selection strategy selects m fittest individuals from n randomly selected individuals based on tournament [5]. This strategy is subject to arbitrary stochastic effects in the same way as roulette-wheel selection - there is no guarantee that the best individual survive through the selection process.

The rank-based selection strategy is similar to roulette wheel selection however an individual assigned a rank instead of the fitness value. This imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time. All the above-mentioned selection strategies are based on individual's current performance and ignore past achievements of an individual throughout the generations.

This paper proposes a novel evolutionary probabilistic selection strategy called *merit-based selection* strategy to mitigate the above-mentioned and other problems associated with widely used selection strategies in EAs. The rest of the paper is organized as follows. Section II a description of DE algorithm. Section III provides a comprehensive review of the state-of-the-art work on improving the DE algorithm. Section IV provides the technical description of the proposed algorithm, called merit-based mutation strategy for DE (MDE). Section V presents the benchmark functions utilized in our experiments, and experimental results and analysis. Finally, the paper is concluded in Section VI.

II. DIFFERENTIAL EVOLUTION: A BRIEF INTRODUCTION

DE was introduced by Storn and Price [6] as a global efficient optimization algorithm. It attempts to solve a problem by selecting a better candidate solution based on its fitness value. Similar to all other P-methaueristcs, DE starts with a population of *NP D*-dimensional parameter vectors representing initial candidate solutions.

Assume that solution $X_{i,G}$ is a set of D-variable which is represented by a D-dimensional variable row vector.

$$X_{i,G}, i = 1,2,\dots,NP,$$

where G indicates the generation and NP is population size. The initial population ($G = 0$) is generated uniformly randomly as follows:

$$x_{i,j,0} = x_{j,min} + rand_{i,j}(0,1)\cdot\left(x_{j,max} - x_{j,min}\right), \quad (1)$$

where $rand_{i,j}(0,1)$ is a uniform random number between 0 and 1; $x_{j,max}$ and $x_{j,min}$ are variable boundaries.

A. Mutation

For each target vector $X_{i,G}$, a mutant vector (noisy vector) $V_{i,G}$ is generated using certain mutation strategy. The general DE mutation and crossover strategy is denoted by *DE/X/Y/Z*; where *X* is the vector to be mutated; *Y* is the number of difference vectors used; and *Z* is the crossover scheme which is either binomial or exponential. The most commonly used mutation strategies are the followings:

DE/best/1: $$V_{i,G} = X_{best,G} + F\left(X_{b,G} - X_{c,G}\right) \quad (2)$$

DE/rand/1: $$V_{i,G} = X_{a,G} + F\left(X_{b,G} - X_{c,G}\right) \quad (3)$$

DE/rand-to-best/1: $$V_{i,G} = X_{a,G} + F\left(X_{best,G} - X_{b,G}\right) + F\left(X_{c,G} - X_{d,G}\right) \quad (4)$$

DE/best/2: $$V_{i,G} = X_{best,G} + F\left(X_{a,G} - X_{b,G}\right) + F\left(X_{c,G} - X_{d,G}\right) \quad (5)$$

DE/rand/2: $$V_{i,G} = X_{a,G} + F\left(X_{b,G} - X_{c,G}\right) + F\left(X_{d,G} - X_{e,G}\right) \quad (6)$$

where $X_{best,G}$ is the best individual vector in term of fitness value; and $X_{a,G}$, $X_{b,G}$, $X_{c,G}$, $X_{d,G}$ and $X_{e,G}$ are different random vectors chosen from the current population,

$$i \neq a \neq b \neq c \neq d \neq e$$

The population size should be $NP \geq 4$ and $F \in [0, 2]$ is a real constant number which controls the diversification of the solution vector.

B. Crossover

There are mainly two crossover strategies of DE: binomial and exponential. In the binomial crossover the trial vector is determined by following equation:

$$U_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Di,G+1}), \quad (7)$$

where

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & if\ rand_j \leq CR\ or\ j = k \\ x_{ji,G} & otherwise \end{cases} \quad (8)$$

For $j = 1,2,\dots,D$, $rand_j \in [0,1]$, $CR \in [0,1]$ and $k \in \{1,2,\dots,D\}$ is a random parameter index which guarantees that $U_{i,G+1}$ gets at least one variable from $V_{i,G+1}$.

On the other hand, in the exponential crossover the trial vector is generated using two-point modulo operation. The first point $n \in [1, D]$ is selected randomly to be the starting point in the target vector, where the exchange of variables with the mutant vector and the second point $L \in [1, D]$ determines the contribution (i.e., of variables) of the mutant vector. The starting position of crossover is selected from the mutant vector until a random number surpasses the CR value or the number of consecutive variables is equal to the random point selected, then the remaining vectors from the target vector.

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & for\ j = \langle n\rangle_D, \langle n+1\rangle_D, \dots, \langle n+L-1\rangle_D \\ x_{ji,G} & for\ all\ other\ j \in [1, D] \end{cases} \quad (9)$$

where the $\langle\ \rangle_n$ denote a modulo function with modulus D; and L is generated according to the following pseudo-code:

```
$L = 0$
DO
    $L = L + 1$
WHILE ($rand(0,1) < CR\ AND\ L < D$)
```

C. Selection

The following greedy selection is used to determine which vector $\{U_{i,G}, X_{i,G}\}$ should be a member of the next population based on cost value.

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & if\ f\left(U_{i,G+1}\right) \leq f\left(X_{i,G}\right) \\ X_{i,G} & Other\ wise \end{cases} \quad (10)$$

III. Related Works

Due to the simplicity and effectiveness of DE, many researchers are focusing on its improvement and this resulted many variants of the basic algorithm with improved performance. One variant of the DE algorithm involve slight modification of the mutation and crossover strategies. For example, Ali and Pant [7] proposed a Cauchy mutation strategy to get a better trade-off between the convergence rate and robustness. They have used a failure counter to track the progress of individuals so that the individuals that fail to show any improvement in the function value for a successive number of generations are subject to Cauchy mutation with the hope of pulling them out of a local optimum.

Similarly, Islam et al. [8] proposed mutation and crossover strategy similar to DE/current-to-best/1 scheme to overcome the premature convergence and stagnation problems observed in the classical DE. The proposed mutation strategy selects the best vector from a dynamic group of $q\%$ of the randomly selected population members. Moreover, their crossover strategy uses a vector that is randomly selected from the p top-ranking vectors according to their objective values in the current population (instead of the target vector) to promote the inclusion of generic information from the elite individuals in the current generation. The parameter p is linearly reduced with generations to maintain the exploration and exploitation stages by gradually downsizing the elitist portion of the population.

Gong and Cai [9] proposed rank-based mutation operators for the DE algorithm, where parents are selected based on their rankings. In each iteration, parents are assigned their corresponding ranking according to their fitness. The mutant vectors are then selected based on selection probability, which is proportional to the rankings of parents in the current population. This strategy imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time. However, it ignores past achievements (history) of individuals throughout the generations, thereby losing good historical information of individuals that can lead to better convergence.

Epitropakis et al. [10] proposed proximity-based DE where mutant vectors are selected from the nearest neighbors of the target vector instead of random vectors. The neighbor vectors are selected based on probabilities inversely proportional to the distance from the mutated individual. It has shown to reduce the excessive exploratory nature of DE and promote exploitation in some areas of the search space. The key role of the proximity framework is to exploit possible clustering structure of the population over the problem's minima and incorporate this information in the evolution phase of the algorithm. More on modified mutation and crossover strategies of DE can be found in [11] – [14].

Other variants of DE include adaptive parameter control and embedding new components in DE. Zhang and Sanderson [15] proposed a parameter adaptation strategy (called JADE) by evolving the mutation factors and crossover probabilities based on their historical record of success to eliminate the need of prior knowledge of the relationship among the parameter settings and the characteristics of optimization problems. The mutation factor for each individual is independently generated according to the Cauchy distribution with location updated based on the Lehmer mean in each generation.

Similarly, Brest et al. [16] proposed a self-adaptive approach for DE control parameters in attempt to determine the best values of F and CR for any given problem. Their approach extends the representation of each individual in the population with F and CR values to go through the evolution process in hope that the control parameters lead to better individuals.

Rahnamayan et al. [17] have proposed an opposition-based DE (ODE) based on opposition-based learning and generation jumping in order to accelerate the convergence rate of DE. The population initialization stage of ODE involves generating random solutions as well as the opposite population and, subsequently selecting the fittest individuals from the union of these two sets. Similarly, based on generation jumping rate, opposite of the current population is generated within reduced search space and only the elite members of the population advance to the next generation.

IV. Proposed Algorithm: MDE

The idea of merit-based selection in general is not new – it has been used in professional sports for determining entry and seeding in all tournaments. In professional sports, merit-based ranking is done based on a ranking period, usually the immediate past 52 weeks performances of each individual. This approach gives a chance to those individuals who have not done well (because of injury or "luck") at the current stage of the tournament.

In this section, we discuss the main concepts behind a merit-based mutation strategy for differential evolution (MDE) algorithm. In nature, prominent species retain and utilize valuable past and current information to improve their fitness and the fitness of future generations (offspring). Similarly, MDE utilizes past and current performances of individuals to guide the search process. MDE is similar to the classical DE except that at the mutation stage the mutant vectors are selected based on their "merit" (i.e. past and current performance). The proposed scheme consists of two memories: the short-term and the long-term memories. The short-term memory stores individuals' current improvement (objective space) and the long-term memory stores the overall improvement made by each individual.

A. Short-term Performance

The short-term performance of MDE assigns performance weights proportional to each individual's improvement. They are computed as follows (for maximization problem the less than "<" operator should be replaced with greater than ">" operator):

$$W_i^S = \begin{cases} f(X_i^{prev}) - f(X_i^{Curr}) & if\ \ f(X_i^{Curr}) < f(X_i^{Prev}) \\ 0 & otherwise \end{cases} \tag{11}$$

where $i = \{1, 2, \ldots, NP\}$ (NP: population size) and the selection probability is defined as:

$$p_i^S = \frac{W_i^S}{\sum_{k=1}^{NP} W_k^S} \tag{12}$$

B. Long-term Performance

The main goal of the long-term performance measure is to eliminate the bias towards individuals which do not show any or little improvement in the current generation but have shown significant contribution in terms of fitness improvement throughout generations. Thus, the long-term weight stores the total performance of each individual in the past generations as well as the current generation.

At the initialization stage, the long-term weights of MDE computed as follows:

$$W_i^L = \frac{max[f(X_1), f(X_2), \ldots, f(X_{NP})] - min[f(X_1), f(X_2), \ldots, f(X_{NP})]}{NP} \quad (13)$$

The main purpose of Eq. (13) is to avoid favouritism towards the best individuals at the initialization stage that may cause premature convergence. The division by NP in Eq. (13) is avoid W_i^L from being the dominating value when individuals' future improvements are significantly smaller than W_i^L.

The weight factors in each generation are updated as follows (for maximization problem the less than "<" operator should be replaced with greater than ">" operator):

$$W_i^L = \begin{cases} W_i^L + W_i^S & if\ f(X_i^{Curr}) < f(X_i^{Prev}) \\ W_i^L & otherwise \end{cases} \quad (14)$$

where $i = \{1,2, \ldots, NP\}$ and the selection probability is defined as:

$$p_i^L = \frac{W_i^L}{\sum_{k=1}^{NP} W_k^L} \quad (15)$$

C. Mutation

The classical mutation strategies in Eqs. (2) to (6) involve selecting several mutant vectors randomly. However in MDE the mutant vectors are selected based on a selection pressure probability defined as below:

$$p_i = \frac{1}{2}[p_i^L + p_i^S] \quad (16)$$

where p_i^L and p_i^S the long and short-term selection probabilities of the i^{th} individual. Since the overall selection pressure is proportional to p_i^L and p_i^S, individuals with little or no improvement at the current stage would still have a chance to be selected based on their past performance. Algorithm 1 shows the pseudocode of the proposed MDE algorithm.

D. Advantages of the Proposed Algorithm

There are several technical and performance advantages of the proposed MDE algorithm over traditional selection strategies used in EA.

- The fitness of an individual is measured based on improvements over the short-term and the long-term (i.e. current performance is not the solely criteria for elitism). This modification allows individuals with little or no improvement at the current stage to still have a chance to be selected based on their past performance. This is similar to the ranking/point method used in professional sports such as tennis and golf.
- MDE is computationally lite; the complexity of computing the short-term and long-term performances is $O(NP)$- it eliminates the need of sorting required by some of the selection strategies such as elitism and/or rank-based selection methods. Note that $O(NP)$ storage is required this procedure.
- Unlike the traditional proportional fitness initialization, MDE initialization assigns equal weights to all individuals. This would not lead to favouritism towards the best individuals at the initialization stage which may cause premature convergence.
- Since MDE uses the improvements made by individuals (always positive) throughout the generation, it does not need any transformation or scaling required by roulette-wheel method when the objective function values are large, negative, or the objective is minimization rather than maximization.

Algorithm 1: Merit-based Mutation Strategy for DE (MDE)

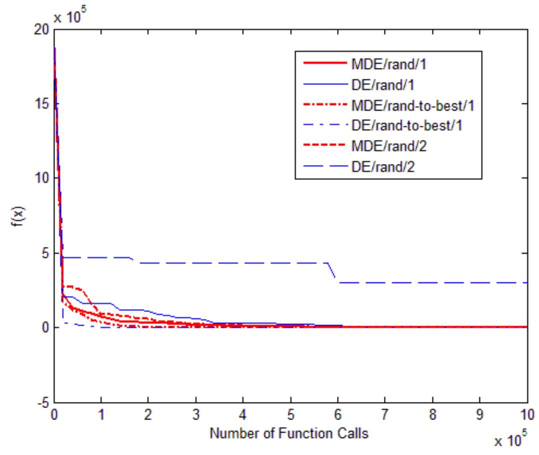
```
1.  Generate uniformly distributed random population $X_0$
    /* Evaluate current population fitness */
2.  $Val_i = f(X_i)\ for\ i = 1, 2, \ldots, NP$
    /* Initialize the short-term and long term-performance */
3.  $W_i^S = 0\ \ for\ i = 1, 2, \ldots, NP$
4.  If $max(val) > min(val)$
5.      $W_i^L = \frac{max(val) - min(val)}{NP}\ for\ i = 1, 2, \ldots, NP$
6.  Else
7.      $W_i^L = 1\ for\ i = 1, 2, \ldots, NP$
8.  End If
    /* Initialize selection probability */
9.  $p_i = 1/NP\ \ for\ i = 1, 2, \ldots, NP$
10. While $NFC\ <\ MAX_{NFC}$
11.     For $i = 1\ to\ NP$
12.         Select mutant vectors based on $p_i$ /*Roulette Wheel*/
13.         Apply mutation /* One of the mutation strategy from
            eqns. (2) to (6) */
14.         Apply binomial crossover /* Eqn. (8) */
15.         /*Update the short-term and long-term performance*/
16.         If $f(U_i) < f(X_i)$
17.             $X_i' = U_i$
18.             $W_i^S = |f(U_i) - f(X_i)|$
19.             $W_i^L = W_i^L + W_i^S$
20.         Else
21.             $X_i' = X_i$
22.             $W_i^s = 0$
23.         End If
24.     End For
        /*Update selection probability*/
25.     For $i = 1\ to\ NP$
26.         $p_i = \frac{1}{2}[W_i^s/sum(W^S)] + \frac{1}{2}[W_i^{L:}/sum(W^L)]$
27.     End For
28.     $X' = X$
29. End While
```

V. Experimental Settings and Results

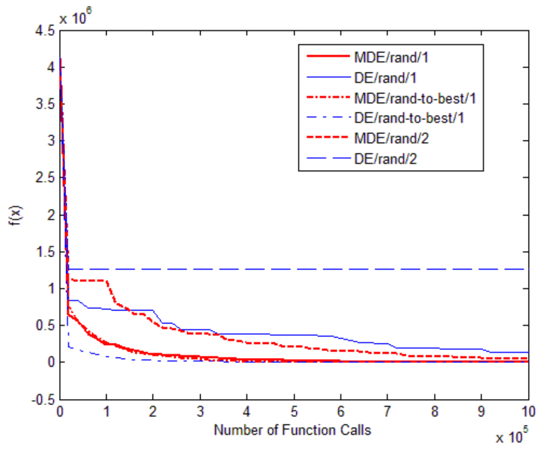
A. Benchmark Functions

In order to test the quality of the proposed algorithm, we have used 28 well-known numerical benchmark problems taken from the IEEE Congress on Evolutionary Computation

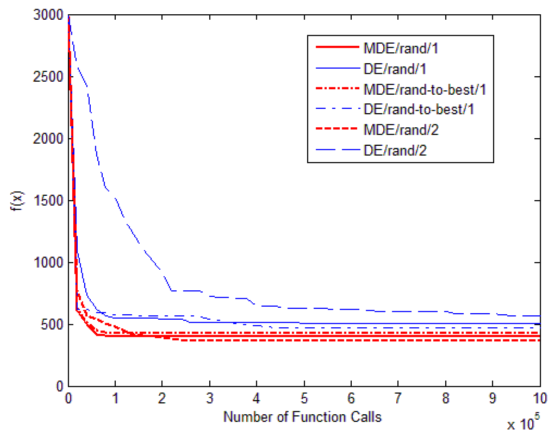
(CEC-2013) special session on real parameter optimization [20]. The definition of the benchmark functions and their global in [20]. Test problems $f_1 - f_5$ are shifted and/or rotated unimodal functions, problems $f6 - f20$ are multimodal shifted and/or rotated continuous functions (except $f13$), and problems $f21 - f28$ are composition functions (a function composed of three or more function from problems $f1 - f20$. The search range for all the problems are $[-100, 100]^D$. All the test functions used in this paper should be minimized.



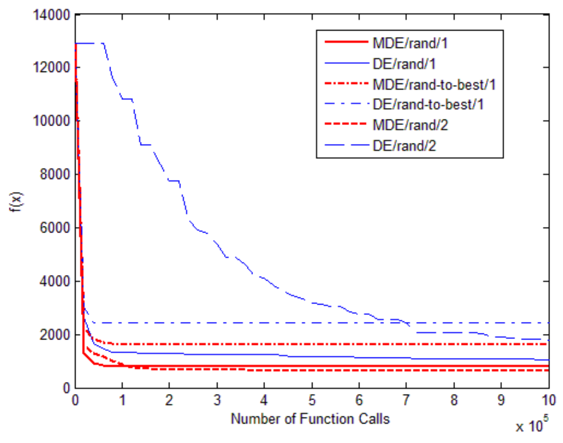
(a) $f_4, D = 30$



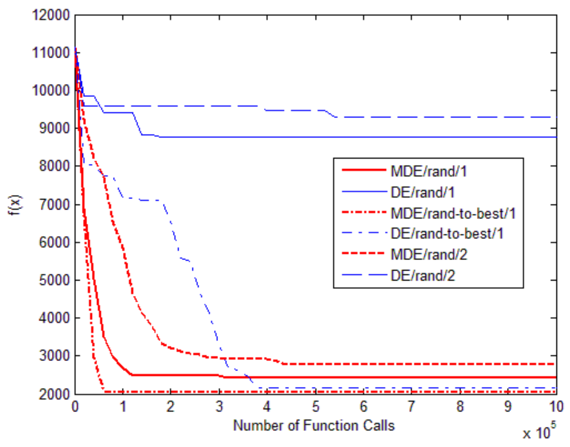
(b) $f_4, D = 100$



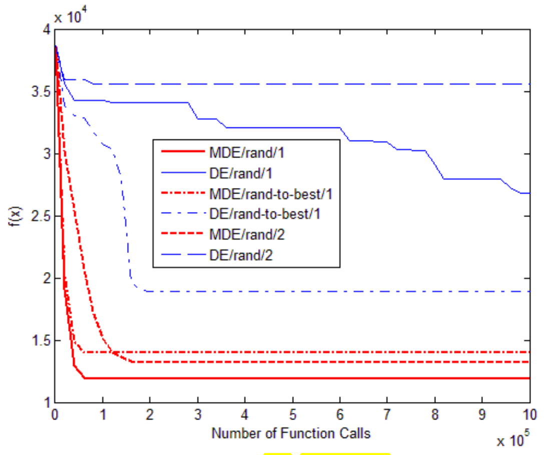
(c) $f_{17}, D = 30$



(d) $f_{17}, D = 100$



(e) $f_{22}, D = 30$



(f) $f_{22}, D = 100$

Fig. 1. Sample graphs (best solution versus NFCs) for performance comparison between DE and MDE for D = 30 and D = 100.

B. Parameter Settings

The classical DE has two problem-dependent control parameters F and CR which need to be tuned by the user. However, in order to maintain a consistent and fair comparison, the parameter settings of DE and MDE are kept the same for all experiments. In addition the parameters settings used in this study are extensively utilized in the literature. Parameter settings for all conducted experiments are as follows:

- Population size, $N_p = 100$ [15]-[17]
- Crossover factor, $CR = 0.9$ [6], [15], [18], [19]
- Mutation scaling factor, $F = 0.5$ [6], [15], [18]
- Maximum Function calls, $MAX_{NFC} = 10{,}000 * D$

To evaluate the performance of the proposed algorithm, we have used an error measure, defined as $|f(x) - f(x^*)|$ where x^* is the global optimum of the benchmark function and x is the best solution achieved after $10{,}000 * D$ function evaluations, where D is the dimensionality of the problem. Furthermore, all algorithms were executed 25 times independently and the mean and standard deviation of each algorithm were recorded.

C. Experimental Results and Analysis

Four series of experiments have been conducted to evaluate the performance of the proposed algorithm. Wilcoxon's rank-sum statistical [21] is conducted at the 5% significance level in order to evaluate the statistical significance the obtained results. Furthermore, error values less than 10^{-8} are reported as zero. In the first experiment, we have compared the performance of five commonly used traditional mutation strategies on the 28 benchmark problems to select the best and the worst mutation strategies for these test problems. The experiments consisted of having respective dimensionalities of 10, 30, 50, and 100.

TABLE I. The Number of Wins by Each Mutation Strategy Out of 28 Benchmark Problems For D = 10, 30, 50 and 100.

D	DE/best/1	DE/rand/1	DE/rand-to-best/1	DE/best/2	DE/rand/2
10	14	7	2	5	3
30	10	12	2	4	0
50	11	13	2	2	0
100	11	14	1	2	0
Total	**46**	**46**	**7**	**13**	**3**

TABLE II. Comparison of Three Binomial Mutation Strategies (DE/Rand/1, DE/Rand-to-best/1, and DE/Rand/2) Against Their Corresponding MDE Strategy (D = 30). Mean Best and Standard Deviation (Std Dev) of 25 Runs and 10,000·D Function Calls Are Reported. The Best Algorithm is highlighted in Grey. The Total Row Shows the Number of Wins by Each Mutation Strategy. * Indicates the Two-tailed Wilcoxon's Rank-sum Test at a 0.05 Significance Level.

	DE/rand/1		MDE/rand/1		DE/rand-to-best/1		MDE/rand-to-best/1		DE/rand/2		MDE/rand/2	
F	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.
f1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.72E-04	5.85E-04	0.00E+00*	6.82E-14	3.61E+00	1.04E+00	0.00E+00*	0.00E+00
f2	1.13E+06	5.65E+05	1.31E+05*	6.17E+04	5.83E+07	1.59E+07	4.23E+05*	2.45E+05	2.61E+08	6.31E+07	3.27E+05*	2.07E+05
f3	4.31E-01*	1.07E+00	8.49E+07	8.92E+07	1.60E+11	1.42E+11	1.07E+07*	1.62E+07	8.95E+11	3.14E+11	5.86E+06*	8.27E+06
f4	1.45E+03	6.76E+02	2.27E+02*	5.82E+01	7.06E+04	2.99E+04	2.63E+01*	1.52E+01	1.94E+05	4.61E+04	3.73E+02*	1.53E+02
f5	0.00E+00	5.68E-14	0.00E+00	0.00E+00	7.48E-02	2.51E-02	0.00E+00*	7.96E-14	3.05E+01	8.98E+00	0.00E+00*	0.00E+00
f6	1.65E+01*	1.22E+00	1.94E+01	1.11E+00	2.09E+01	6.96E-01	4.06E+00*	1.25E+00	3.42E+01	3.13E+00	3.94E+00*	7.75E-01
f7	3.64E-01*	2.41E-01	5.96E+01	1.72E+01	5.69E+02	1.05E+02	8.67E+01*	2.31E+01	1.36E+03	3.09E+02	3.86E+01*	2.51E+01
f8	2.09E+01*	4.74E-02	2.10E+01	5.04E-02	2.09E+01	5.63E-02	2.10E+01	2.74E-02	2.10E+01	3.92E-02	2.10E+01	4.62E-02
f9	3.91E+01	1.34E+00	2.89E+01*	6.98E+00	3.97E+01	9.69E-01	3.93E+01*	1.21E+00	3.88E+01	1.04E+00	3.97E+01	1.01E+00
f10	5.92E-03*	5.53E-03	5.32E-02	2.41E-02	8.94E+00	3.74E+00	2.69E-02*	1.46E-02	4.06E+02	8.64E+01	1.58E-02*	9.86E-03
f11	1.13E+02	3.11E+01	3.01E+01*	6.31E+00	2.14E+02	1.19E+01	4.30E+01*	1.45E+01	2.32E+02	1.05E+01	2.07E+01*	8.64E+00
f12	1.80E+02	6.02E+00	3.31E+01*	7.86E+00	2.27E+02	1.07E+01	5.40E+01*	1.31E+01	2.59E+02	1.29E+01	3.10E+01*	1.60E+01
f13	1.81E+02	8.85E+00	7.87E+01*	2.96E+01	2.32E+02	1.34E+01	1.08E+02*	2.03E+01	2.51E+02	1.80E+01	4.26E+01*	1.17E+01
f14	6.64E+03	5.49E+02	2.15E+03*	3.72E+02	8.00E+03	3.05E+02	1.69E+03*	6.03E+02	8.20E+03	2.31E+02	1.64E+03*	3.39E+02
f15	7.86E+03	3.18E+02	6.25E+03*	1.05E+03	8.42E+03	2.98E+02	7.82E+03*	6.75E+02	8.44E+03	3.17E+02	8.05E+03*	2.34E+02
f16	2.51E+00	2.42E-01	2.47E+00	2.53E-01	2.42E+00	3.12E-01	2.33E+00	6.40E-01	2.50E+00	2.02E-01	2.41E+00	2.20E-01
f17	1.81E+02	1.24E+01	7.07E+01*	7.54E+00	2.58E+02	8.49E+00	6.23E+01*	7.49E+00	2.75E+02	5.99E+00	5.41E+01*	5.27E+00
f18	2.14E+02	9.07E+00	1.07E+02*	2.56E+01	2.57E+02	8.51E+00	1.66E+02*	4.58E+01	2.83E+02	1.43E+01	1.88E+02*	3.68E+01
f19	1.54E+01	8.77E-01	3.43E+00*	1.27E+00	1.91E+01	1.76E+00	3.15E+00*	5.40E-01	2.63E+01	1.78E+00	3.29E+00*	9.81E-01
f20	1.50E+01	1.87E-05	1.45E+01	2.65E-01	1.50E+01	2.79E-12	1.46E+01	4.91E-01	1.50E+01	9.98E-10	1.49E+01	2.25E-01
f21	2.80E+02	4.00E+01	3.57E+02	7.03E+01	2.71E+02*	4.57E+01	3.33E+02	7.80E+01	3.70E+02	2.22E+01	2.60E+02*	4.90E+01
f22	7.20E+03	8.85E+02	2.21E+03*	4.40E+02	8.54E+03	2.08E+02	1.83E+03	3.99E+02	8.69E+03	2.89E+02	1.76E+03*	4.33E+02
f23	7.99E+03	2.71E+02	5.26E+03*	1.02E+03	8.50E+03	4.29E+02	6.41E+03	1.59E+03	8.71E+03	2.54E+02	7.78E+03*	7.35E+02
f24	3.00E+02	2.04E+00	2.92E+02*	7.21E+00	3.02E+02	3.42E+00	2.99E+02*	4.11E+00	3.05E+02	3.05E+00	3.03E+02	2.88E+00
f25	2.99E+02	2.54E+00	2.97E+02	6.56E+00	3.00E+02*	2.27E+00	3.00E+02	2.16E+00	3.00E+02	2.28E+00	3.00E+02	3.26E+00
f26	3.71E+02	3.73E+01	3.20E+02	7.94E+01	3.95E+02	1.47E+01	3.48E+02*	7.50E+01	4.03E+02	5.31E+00	3.62E+02*	5.64E+01
f27	1.29E+03	3.15E+01	1.22E+03	7.97E+01	1.31E+03	2.32E+01	1.31E+03*	2.95E+01	1.34E+03	2.80E+01	1.31E+03	3.49E+01
f28	3.00E+02	0.00E+00	3.00E+02	4.69E-13	3.03E+02*	8.17E-01	6.35E+02	5.13E+02	4.99E+02	4.39E+01	3.00E+02*	0.00E+00
Total	**8**		**22**		**4**		**24**		**3**		**25**	

Overall, mutation strategies, DE/best/1 and DE/rand/1 were among the best mutation strategies and DE/rand/2 was the worst in term of solution accuracy. Furthermore, DE/rand/1 performed very well for unimodal problems ($f1 - f5$) and DE/best/1 was the dominant mutation strategy for composition functions ($f21 - f28$). Table I summarizes the total number of best performances by each mutation strategy for $D = 10, 30, 50$, and 100. For multimodal functions ($f6 - f20$) DE/best/1 and DE/rand/1 had roughly the same number of best performances.

In the second set of experiments, we compared performance of the best mutation strategy (DE/rand/1) against MDE/rand/1 (i.e. the three mutant vectors in DE/rand/1 are selected based on the proposed algorithm). The proposed method outperformed the classical DE/rand/1 for most benchmark problems. As the dimension increased from $D = 10$ to$D = 50$, the performance of MDE gets better and outperforms the classical DE/rand/1 in most the benchmark problems (see Table III). Tables II and IV, columns 2 through 5 show the mean best and standard deviation of DE/rand/1 and MDE/rand/1 for $D = 30$ and *100*.

In the third set of experiments, we compared performance of the second worst mutation strategy (DE/rand-to-best/1) against MDE/rand-to-best/1 (i.e. the four mutant vectors in DE/rand-to-best/1 are selected based on the proposed algorithm). The proposed algorithm improved the performance of the corresponding algorithm for most of the test problems in all dimensions except for functions $f8$ and $f28$. In overall, the proposed algorithm outperformed the classical DE in 80% of the test functions. Tables II and IV, columns 6 to 9 show the mean best and standard deviation of DE/rand-to-best/1 and MDE/rand-to-best/1 for $D = 30$ and *100*.

TABLE III. THE OVERALL COMPARISON OF DE AND MDE: NUMBER OF WINS BY EACH MUTATION STRATEGY OUT OF 28 BENCHMARK PROBLEMS FOR D = 10, 30, 50 AND 100.

D	rand/1		rand-to-best/1		rand/2	
	DE	MDE	DE	MDE	DE	MDE
10	12	19	6	23	4	26
30	8	22	4	24	3	25
50	7	21	7	22	1	27
100	12	17	6	23	2	26
Total	**39**	**79**	**23**	**92**	**10**	**104**

The last set of experiments involved the worst mutation strategy (DE/rand/2) against MDE/rand/2 (i.e. the five mutant vectors in DE/rand-to-best/1 are selected based on the proposed algorithm). The corresponding MDE strategy outperformed the classical mutation strategy almost in all benchmark problems and in all tested dimensions. It has shown the most improvement as compared to MDE and the corresponding other mutation strategies and this can be attributed to the fact that DE/rand/2 has five randomly (blindly) selected mutant vectors instead of three mutant vectors in DE/rand/1. Tables II and IV, columns 10 to 13 show the mean best and standard deviation of DE/rand/2 and MDE/rand/2 for $D = 30$ and *100*. Figure 1 illustrates the performance comparison between traditional DE and MDE for the three mutation strategies when D = 30 and 100. These curves show MDE converges faster than the conventional DE in all mutation strategies. Table III summarizes the overall performance of MDE for all tested dimensions (*D = 10, 30, 50* and *100*). Based on the observed results in four sets of experiments, the results can be summarized as follows:

TABLE IV. COMPARISON OF THREE BINOMIAL MUTATION STRATEGIES (DE/RAND/1, DE/RAND-TO-BEST/1, AND DE/RAND/2) AGAINST THEIR CORRESPONDING MDE STRATEGY (D = 100). MEAN BEST AND STANDARD DEVIATION (STD DEV) OF 25 RUNS AND 10,000·D FUNCTION CALLS ARE REPORTED. THE BEST ALGORITHM IS EMPHASIZED IN GREY. THE TOTAL ROW SHOWS THE NUMBER OF WINS BY EACH MUTATION STRATEGY. * INDICATES THE TWO-TAILED WILCOXON'S RANK-SUM TEST AT A 0.05 SIGNIFICANCE LEVEL.

F	DE/rand/1		MDE/rand/1		DE/rand-to-best/1		MDE/rand-to-best/1		DE/rand/2		MDE/rand/2	
	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.
f1	0.00E+00	1.11E-13	0.00E+00	0.00E+00	4.41E-01	2.19E-01	0.00E+00*	1.14E-13	6.31E+03	1.15E+03	0.00E+00*	0.00E+00
f2	2.10E+07	6.34E+06	2.21E+06*	5.83E+05	5.08E+09	8.45E+08	3.17E+06*	1.10E+06	9.89E+09	1.86E+09	1.73E+06*	5.70E+05
f3	6.12E+08*	4.70E+08	8.11E+13	2.38E+14	1.18E+14	5.63E+13	1.41E+09*	9.08E+08	7.99E+14	4.71E+14	4.93E+08*	3.47E+08
f4	1.51E+05	1.32E+04	5.96E+03*	1.91E+03	9.50E+05	2.40E+05	3.09E+03*	3.29E+03	9.99E+05	2.03E+05	5.56E+04*	1.74E+04
f5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.13E+00	2.24E+00	0.00E+00*	5.57E-14	5.23E+03	1.05E+03	0.00E+00*	0.00E+00
f6	1.23E+02*	3.91E+01	1.88E+02	4.37E+01	1.20E+02*	4.20E+01	1.48E+02	3.12E+01	1.54E+03	3.56E+02	1.28E+02*	3.56E+01
f7	1.95E+02*	7.00E+01	1.11E+06	1.20E+06	5.80E+03	1.52E+03	1.82E+02*	2.51E+01	1.38E+04	4.37E+03	1.06E+02*	1.42E+01
f8	2.13E+01*	1.92E-02	2.13E+01	1.52E-02	2.13E+01	1.68E-02	2.13E+01	1.65E-02	2.13E+01	1.58E-02	2.13E+01	2.04E-02
f9	1.60E+02*	3.64E+00	1.60E+02	1.36E+00	1.60E+02	1.48E+00	1.60E+02	1.38E+00	1.61E+02	1.72E+00	1.60E+02	2.32E+00
f10	8.55E-02*	4.18E-02	1.48E-01	5.97E-02	2.26E+02	4.28E+01	7.81E-02*	3.81E-02	1.17E+04	1.74E+03	6.68E-02*	4.29E-02
f11	8.76E+01*	5.11E+01	3.10E+02	5.74E+01	9.52E+02	5.14E+01	4.36E+02*	6.31E+01	1.16E+03	4.68E+01	1.75E+02	3.09E+01
f12	8.49E+02	2.45E+01	3.50E+02*	6.26E+01	1.03E+03	3.26E+01	5.70E+02*	1.17E+02	1.31E+03	6.51E+01	5.36E+02*	3.36E+02
f13	8.59E+02	2.68E+01	5.87E+02*	4.93E+01	1.03E+03	3.49E+01	1.02E+03	1.42E+02	1.31E+03	7.51E+01	9.04E+02*	3.06E+01
f14	2.54E+04	2.20E+03	1.34E+04*	6.80E+02	3.33E+04	2.93E+02	1.19E+04*	1.48E+03	3.33E+04	6.22E+02	1.27E+04*	1.60E+03
f15	3.24E+04	4.62E+02	3.17E+04*	3.12E+02	3.32E+04	4.24E+02	3.27E+04	5.84E+02	3.36E+04	3.24E+02	3.28E+04*	4.31E+02
f16	3.92E+00	3.07E-01	4.00E+00	2.29E-01	3.95E+00	2.35E-01	3.90E+00	2.19E-01	4.08E+00	1.43E-01	4.06E+00	2.11E-01
f17	6.95E+02	5.43E+01	5.96E+02	1.21E+02	1.09E+03	2.45E+01	5.18E+02*	5.45E+01	1.48E+03	5.79E+01	3.93E+02*	4.58E+01
f18	9.28E+02*	2.46E+01	9.99E+02	3.88E+01	1.11E+03	2.68E+01	1.11E+03	5.67E+01	1.49E+03	6.64E+01	9.76E+02*	1.98E+01
f19	6.93E+01	4.76E+00	5.77E+01*	1.30E+01	9.54E+01	4.47E+00	6.18E+01*	2.47E+01	4.12E+05	5.79E+05	2.46E+01*	3.83E+00
f20	5.00E+01	1.49E-10	5.00E+01*	8.84E-04	5.00E+01	0.00E+00	5.00E+01	0.00E+00	5.00E+01	8.95E-14	5.00E+01	9.25E-13
f21	3.50E+02*	5.00E+01	6.14E+03	3.37E+03	5.79E+02	1.23E+02	3.80E+02*	4.00E+01	3.92E+03	2.70E+02	3.70E+02*	4.58E+01
f22	2.57E+04	1.32E+03	1.40E+04*	2.18E+03	3.40E+04	4.34E+02	1.17E+04*	1.86E+03	3.43E+04	3.29E+02	1.21E+04*	1.02E+03
f23	3.29E+04	6.55E+02	3.18E+04*	6.39E+02	3.43E+04	5.87E+02	3.37E+04	5.83E+02	3.48E+04	7.14E+02	3.40E+04*	3.34E+02
f24	6.08E+02	4.11E+00	6.06E+02	8.09E+00	6.13E+02	4.22E+00	6.12E+02	4.46E+00	6.19E+02	5.22E+00	6.12E+02*	3.94E+00
f25	6.04E+02	2.36E+00	6.03E+02	3.43E+00	6.04E+02*	2.50E+00	6.09E+02	3.31E+00	6.08E+02	3.46E+00	6.09E+02	4.28E+00
f26	7.08E+02	4.69E+00	7.04E+02	5.36E+00	7.14E+02	5.31E+00	7.09E+02	3.51E+00	7.24E+02	4.07E+00	7.06E+02*	6.08E+00
f27	4.34E+03	4.00E+01	4.33E+03	4.55E+01	4.42E+03	5.19E+01	4.38E+03	5.68E+01	4.51E+03	4.39E+01	4.40E+03*	4.05E+01
f28	3.62E+03*	1.07E+03	2.08E+04	7.23E+03	5.19E+03*	7.74E+01	6.89E+03	1.98E+03	1.23E+04	1.85E+03	3.49E+03*	9.02E+02
Total	**2**		**26**		**6**		**23**		**2**		**26**	

- As the dimension increases the performance of MDE gets better in term of the solution accuracy in most of the experimented DE mutation strategies. This suggests that MDE may be suitable for large-scale global optimization problems.
- The performance of MDE based mutation strategies were better than those of the classical mutation strategies. This suggests the proposed merit-based computation has the capability of improving the performance of algorithms involving some sort of selection strategy (at least it would be computationally "cheaper" than other selection strategies).
- The elitism of an individual should be measured based on its performance, not only at the current generation (MDE short-term performance) but also the overall performance throughout the generations (MDE long-term performance).
- As the number of randomly selected vectors increases (e.g. three random vectors in DE/rand/1 to five random vectors in DE/rand/2) the performance of the corresponding MDE algorithm showed better improvement. This suggests the proposed merit-based selection strategy can significantly increase the performance of algorithms with many random mutant vectors.

VI. Conclusion

In this paper, we proposed a novel merit-based mutation strategy for DE (MDE). MDE uses short-term performance (the performance of each individual in the current generation) and long-term performance (the performance of each individual throughout the generations) to minimize the possible negative effects of a pure "blind" selection strategy employed by the traditional DE mutation strategies. The performance of MDE was compared to three traditional DE mutation strategies namely, DE/rand/1, DE/rand-to-best/1 and DE/rand/2. Experimental results on 28 benchmark problems showed that the performance of the traditional DE mutations can significantly be improved by MDE.

Almost in all test problems, the proposed algorithm outperformed the parent traditional DE mutation strategy in term of solution accuracy. As the number of randomly selected mutant vectors increased (e.g., three random mutant vectors in DE/rand/1 to five mutant random vectors in DE/rand/2) the performance of the corresponding MDE algorithm for DE/rand/2 showed better improvement as compared to MDE for DE/rand/1. This shows that as the number of randomly selected vectors increase the solution accuracy of DE degrades and merit-based selection strategy can greatly improve the performance.

For future work, first, we would like to investigate the impact of short-term to long-term performance weights (in the current proposal the short-term and the long-term have the same impact factor in the selection process). Second, we would like to investigate the impact of using only n generations' performance as supposed to all generations. Last but not least, we would like to extend the merit-based selection strategy to evolutionary multi-objective algorithms involving random, rank or elite selection mechanism.

References

[1] S. Binitha & S. S. Sathya. A Survey of Bio inspired Optimization Algorithms, International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-2, Issue-2, 2012.

[2] E. G. Talbi, Metaheuristics from design to implementation, John Wiley and Sons, ISBN: 978-0470-27858-1, 2009.

[3] M. Gendreau & J. Y. Potvin (eds.). Handbook of Metaheuristics, International Series in Operations Research & Management Science, Springer, 2010.

[4] C. R. Reeves & J. E. Rowe. "Genetic algorithms-principles and perspectives: a guide to GA theory," (Vol. 20). Springer, 2002.

[5] B. L. Miller & D. E. Goldberg. "Genetic algorithms, tournament selection, and the effects of noise." Complex Systems 9.3 pp. 193-212, 1995.

[6] R. Storn & K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report. International Computer Science Institute, Berkley, 1995.

[7] M. Ali & M. Pant. Improving the performance of differential evolution algorithm using Cauchy mutation. Soft Computing, 15(5), pp. 991-1007, 2011.

[8] S. Islam, S. Das, S. Ghosh, S. Roy, & P. N. Suganthan. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 42(2), pp. 482-500, 2012.

[9] Gong, W. & Cai, Z. Differential evolution with ranking-based mutation operators. Cybernetics, IEEE Transactions on, 43(6), 2066-2081, 20013.

[10] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, & M. N. Vrahatis. Enhancing differential evolution utilizing proximity-based mutation operators. Evolutionary Computation, IEEE Transactions on, 15(1), pp. 99-119, 2011.

[11] H. Y. Fan & J. Lampinen. A trigonometric mutation operation to differential evolution. Journal of Global Optimization, 27(1), pp. 105-129, 2003.

[12] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, & M. F. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing, 11(2), pp. 1679-1696, 2011.

[13] S. Das, A. Abraham, U. K. Chakraborty, & A. Konar. Differential evolution using a neighborhood-based mutation operator. Evolutionary Computation, IEEE Transactions on, 13(3), pp. 526-553, 2009.

[14] P. Kaelo & M. M. Ali. Differential evolution algorithms using hybrid mutation. Computational Optimization and Applications, 37(2), pp. 231-246, 2007.

[15] J. Zhang & A. C. Sanderson. JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, 13(5), pp. 945-958, 2009.

[16] J. Brest, S. Greiner, B. Boskovic, M. Mernik, & V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. Evolutionary Computation, IEEE Transactions on, 10(6), pp. 646-657, 2006.

[17] S. Rahnamayan, H. R. Tizhoosh, & M. M. Salama. Opposition-based differential evolution. Evolutionary Computation, IEEE Transactions on, 12(1), pp. 64-79, 2008.

[18] X. Yao, Y. Liu, & G. Lin. Evolutionary programming made faster. Evolutionary Computation, IEEE Transactions on, 3(2), 82-102, 1999.

[19] F. Neri, & V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review, 33(1-2), 61-106, 2010.

[20] J. J Liang, B. Y Qu, & P. N. Suganthan. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, 2013.

[21] D. J. Sheskin, Handbook of parametric and nonparametric statistical procedures: crc Press, 2003