Differential Evolution with Multiple Strategies for Solving CEC2011 Real-world Numerical Optimization Problems

Saber M. Elsayed, Ruhul A. Sarker and Daryl L. Essam
School of Engineering and Information Technology
University of New South Wales at Australian Defence Force Academy
Canberra 2600, Australia
E-mails: saber.elsayed@student.adfa.edu.au, r.sarker@adfa.edu.au, d.essam@adfa.edu.au

Abstract—**Over the last two decades, many Differential Evolution (DE) strategies have been introduced for solving Optimization Problems. Due to the variability of the characteristics in optimization problems, no single DE algorithm performs consistently over a range of problems. In this paper, for a better coverage of problem characteristics, we introduce a DE algorithm framework that uses multiple search operators in each generation. The appropriate mix of the search operators, for any given problem, is determined adaptively. The proposed algorithm has been applied to solve the set of real world numerical optimization problems introduced for a special session of CEC2011.**

I. INTRODUCTION

SOLVING optimization problems has been a challenging research area in the computer science and optimization domains due to physical, geometric, and other limitations [1].

Evolutionary algorithms (EAs) have a long history of successfully solving optimization problems. The Evolutionary algorithms (EAs) family contains a wide range of algorithms that have been used to solve optimization problems, such as the genetic algorithm (GA) [2], particle swarm optimization (PSO) [3], differential evolution (DE) [4], evolutionary strategies (ES) [5], and evolutionary programming [6]. In this research, we consider DE for solving optimization problems, because DE usually converges quickly, incorporates a relatively simple and self-adapting mutation, and the same settings can be used for many different problems [4].

Due to the variability of problem characteristics, and their underlying mathematical properties, most EAs use specialized search operators that suit the problem on hand. Such an algorithm, even if it works well for one problem, or a class of problems, does not guarantee that it will work for another class of problems. This behavior is consistent with the no free lunch (NFL) theorem [7]. In other words, we can state that there is no single algorithm, or an algorithm with a known search operator, that will consistently perform best for all classes of optimization problems. This motivates us to consider multiple strategies in DE, in order to achieve a better coverage of problems.

In the literature, little has been reported on multiple strategies DE. Mallipeddi *et al.* [8] proposed an ensemble of mutation strategies and control parameters with DE (EPSDE) for solving unconstrained optimization problems. In EPSDE, a pool of distinct mutation strategies, along with a pool of values for each control parameter, coexists throughout the evolution process and competes to produce offspring. Mallipeddi *et al.* [9] also extended the algorithm for constrained optimization by adding an ensemble of four constraint handling techniques. Tasgetiren *et al.* [10] proposed an ensemble DE, in such a way that each individual is assigned to one of two distinct mutation strategies or a variable parameter search (VPS). VPS is used here to enhance the local exploitation capability. However, no adaptive strategy is used in this algorithm. Elsayed et al. [11] proposed a three-strategy based differential evolution algorithm (SAOMDE) for solving constrained optimization problems. SAOMDE utilizes the strengths of three well-known DE variants through an adaptive learning process. The experimental results showed that SAOMDE is not only superior to three single mutations based DEs, but was also better than some of the other state-of-the-art algorithms. Elsayed et al. [12] extended his work in [11] to solve a large set of constrained problems. The proposed algorithms were competitive to other state-of-the-art algorithms.

In this paper, we consider four different mutation strategies and one crossover operator within a single algorithm framework. In the framework, each combination of search operators (a mutation strategy with a crossover operator) has its own sub-population and the sub-population size varies adaptively, as the evolution progresses, depending on the reproductive success of the search operators. However, the sum of the size of all of the sub-populations is fixed during the entire evolutionary process. In implementing the adaptive approach, we propose a measure for reproductive success, based on fitness values and constraint violations. An operator may perform very well at an earlier stage of the evolution process and do badly at a later stage or vice-versa. To consider this fact in effectively designing our algorithm, we set a lower bound on the subpopulation size. The algorithm is applied to the set of real world optimization problems introduced for a special session to be organized in CEC2011 [13].

This paper is organized as follows: after the introduction, section 2 presents the DE algorithm with an overview of its parameters. Section 3 describes the design of our proposed algorithm. The experimental results and the analysis of those results are presented in section 4. Finally, the conclusions are given in section 5.

II. Differential Evolution (DE)

Differential Evolution (DE) is well-known for its success in continuous optimization [4] [14]. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third mutated vector. Then crossover takes place in which the mutated vector's parameters are mixed with the parameters of another predetermined target vector to yield a so-called child vector. If the child vector is better than the target vector, then the child vector replaces the target vector in the next generation. Each population vector has to serve once as the target vector, so that *PS* competitions take place in one generation [4].

The remainder of this section discusses the commonly used DE operators, and parameter analysis

A. Mutation

The simplest form of this operation is for a mutant vector to be generated by multiplying the amplification factor F by the difference of two random vectors, and the result is added to another third random vector, as shown in the following formula:

$$\overrightarrow{V_{z,t}} = x_{r_{1},t} + F \times \left(\left(x_{r_{2},t} - x_{r_{3},t}\right)\right) \tag{1}$$

where r_1, r_2, r_3 are random numbers $\{1,2, ..., \mathrm{PS}\}\backslash\{\mathrm{i}\}$, $r_1 \neq r_2 \neq r_3 \neq z$, *PS* is the population size the scaling factor F is a positive control parameter for scaling the difference vector, and *t* is the current generation.

This operation enables DE to explore the search space and maintain diversity. There are many strategies for mutation, such as DE/rand/1 [15], DE/best/1 [15], DE/rand-to-best/2 [16], rand/2/dir [17], DE/current-to-rand/1 [18], DE/current-to- best/1 [19]

B. Crossover

The DE family of algorithms uses the following two crossover schemes as discussed below.

B.1. Exponential

In this crossover, we first choose an integer *n* randomly within the range [1, D]. This integer acts as a starting point in the target vector, from where the crossover or exchange of components with the donor vector starts. We also choose another integer *L* from the interval [1, D]. *L* denotes the number of components that the donor vector actually contributes to the target. After the generation of n and L, the trial vector is obtained as

$$u_{i,j,t} = \begin{cases} v_{z,j,t} & for\ j = \langle l \rangle_D, \langle l+1 \rangle_D, \dots, \langle l+L-1 \rangle_D \\ x_{z,j,t} & for\ all\ other\ j \in [1, D] \end{cases} \tag{2}$$

where $z = 1,2,\dots,PS$, PS is the population size, $j = 1,2,\dots,\mathrm{D}$, and the angular brackets $\langle l \rangle_{\mathrm{D}}$ denote a modulo function with modulus D, with a starting index l. Also, as stated above, the integer L is drawn from $[1, \mathrm{D}]$.

B.2. Binomial

This crossover type is performed on each of the j^{th} variables whenever a randomly picked number (between 0 and 1) is less than or equal to a crossover rate (*Cr*). The generation number is indicated here by *t*. In this case, the number of parameters inherited from the donor has a (nearly) binomial distribution

$$u_{zj,t} = \begin{cases} v_{zj,t}, & if\ (rand \leq Cr\ or\ j\ =\ j_{rand}) \\ x_{zj,t}, & otherwise \end{cases} \tag{3}$$

where $rand \in [0,1]$, and $j_{rand} \in [1,2,\dots,D]$ is a randomly chosen index, which ensures $\vec{U}_{z,t}$ gets at least one component from $\vec{V}_{z,t}$.

C. A Brief Review and Analysis

From an analysis of the literature, we know that there is no single DE variant that will be able to reach the best results for all types of problems. Mezura et al. [17] performed a comparative study of several DE variants in solving unconstrained global optimization problems. They found that variants "*current-to-best/1*" and "*current-to-rand/1*", with arithmetic recombination, have difficulties when exploring the search space of multimodal functions. They suggested that a combination of arithmetic recombination with the DE mutation based on differences is not suitable for solving problems with high dimensionality. The alternatives "*best/1/bin*" and "*rand/1/bin*" are much better than "*best/1/exp*" and "*rand/1/exp*", due to the fact that in the exponential recombination, not all of the corners of the hypercube formed by the mutation vector and the current parent can be sampled, regardless of the "Cr" value used. The variant "*best/1/bin*" is the most competitive for unimodal problem with both separable and non-separable test functions. However, for multimodal functions, this variant provides competitive results only when the function is separable. For multimodal and non-separable functions, the variant "*rand/2/dir*" is more suitable, due to its ability to incorporate information about the fitness of the individuals in the mutation process.

In DE, there is no single fixed value for each parameter (the amplification factor of scaling factor F, the crossover rate Cr, and the population size PS) that will be able to solve all types of problems with a reasonable quality of solution. Many studies have been conducted on parameter selection. Storn *et al.* [4] recommended a population size of 5–20 times the dimensionality of the problem, and that a good initial choice of F could be 0.5. Gamperle *et al.* [20] evaluated different parameter settings of DE, where a plausible choice of the population size PS is between $3D$ and $8D$, with the scaling factor $F = 0.6$ and the crossover rate Cr in [0.3, 0.9]. Ronkkonen *et al.* [21] claimed that typically $0.4 < F < 0.95$ with $F = 0.9$ is a good first choice and that Cr, typically lies in

(0, 0.2) when the function is separable, while in (0.9, 1) when the function's parameters are dependent. Qin *et al.* [16] proposed a novel Self-adaptive Differential Evolution algorithm (SaDE), where the choice of learning strategy and the two control parameters F and Cr are not required being pre-specified. During evolution, the learning strategy and parameter settings are gradually self-adapted according to the learning experience. Abbass [22] proposed a self-adaptive version (crossover and mutation) for multi-objective optimization problems, where the scaling factor F is generated using a Gaussian distribution $N(0,1)$. Zaharie [23] proposed a parameter adaptation strategy for DE (ADE) based on the idea of controlling the population diversity, and implemented a multiple population approach. Das et al. [24] introduced two schemes for adapting the scale factor F in DE. In the first scheme (called DERSF: DE with random scale factor) they varied F randomly between 0.5 and 1.0 in successive generations. Zamuda et al. [25] presented Differential Evolution with Self-adaptation and Local Search for the Constrained Multiobjective Optimization algorithm (DECMOSA-SQP), which uses the self-adaptation mechanism from DEMOwSA algorithm and a SQP local search. For more details about DE and its parameters and variants, the readers are referred to Das et al. [26].

III. A SELF-ADAPTIVE MUTLI-OPERATOR DE

In this section, we first describe the self-adaptive multi-operator differential evolution algorithm. Finally, we introduce the improvement measure strategy for the adaptive selection of parameters.

A. A Self-Adaptive Multi-Operator Differential Evolution

In the evolution process, the relative performance of the search operators may vary with the progression of generations. This means that one operator may work well in the early stages of the search process and may perform poorly at the later stages, or vice-versa. So it is inappropriate to give equal emphasis on all the operators throughout the entire evolution process. To give higher emphasis on the better performing operators, we propose to change the subpopulation sizes, through dynamic adaptation, based on the relative performances of the operators. We introduce this algorithm as the Self-Adaptive Multi-Operator Differential Evolution (SAMODE). SAMODE uses four different mutation types and one crossover operator. SAMODE is initiated using a random initial population PS with N individuals. Then, PS is divided into four subpopulations p_i of equal size (here subscript i represents the i^{th} crossover mutation operator). Each subpopulation with n_i individuals evolves using their own mutation and the allocated crossover. The generated offspring $O_z = \{O_z^1, O_z^2, \dots, O_z^j, \dots, O_z^D\}$ are evaluated according to their fitness function values and constraint violations. An improvement index for each subpopulation is calculated using the method as discussed below. Based on the improvement index, the subpopulation sizes are either increased or decreased or kept unchanged. As this process may abandon certain operators which may be useful at the later stages of the evolution process, we set a minimum subpopulation size for each operator. Also, after every few (WS) generations the best solutions among the subpopulations are exchanged. The algorithm continues until the stopping criterion is satisfied. The main steps of the SAMODE algorithm are described in Table I.

B. Improvement Measure

To measure the improvement of each combination in a given generation, we consider both the feasibility status and the fitness value, where the consideration of any improvement in feasibility is always better than any improvement in the infeasibility. For constrained problems, for any generation $t > 1$, there arises one of three scenarios. These scenarios, in order from least desirable, to most desirable, are discussed below.

A) Infeasible to infeasible: For any combination i, the best solution was infeasible at generation $t-1$, and is still infeasible in generation t, then the improvement index is calculated as follows.

$$VI_{i,t} = \frac{\left|V_{i,t}^{best} - V_{i,t-1}^{best}\right|}{avg.V_{i,t}} = I_{i,t} \quad (4)$$

where, $V_{i,t}^{best}$ = constraint violation of the best individual at generation t, $avg.V_{i,t}$ = the average violation. Hence $VI_{i,t} = I_{i,t}$ above represents a relative improvement as compared to the average violation in the current generation. As we are using the ε-constraint handling technique, we may accept a new individual that has a larger violation than its donor, if both of them are infeasible and their violations are less than ε value, in this case we set $VI_{i,t} = 0$ in this case.

B) Feasible to feasible: For any combination i, the best solution was feasible at generation $t-1$, and was still feasible in generation t, then the improvement index is:

$$I_{i,t} = \max_i(VI_{i,t}) + \left| F_{i,t}^{best} - F_{i,t-1}^{best} \right| \times FR_{i,t} \quad (5)$$

where $I_{i,t}$= the improvement for combination i at generation t, $F_{i,t}^{best}$= the objective function for the best individual at generation t, the feasibility ratio of operator i at generation t:

$$FR_{i,t} = \frac{Number\ of\ feasible\ solutions\ in\ a\ combination\ i}{combination\ size\ at\ iteration\ t}. \quad (6)$$

To assign a higher index value to the subpopulation with a higher feasibility ratio, we multiply the improvement in fitness value by the feasibility ratio. To differentiate between the improvement index of feasible and infeasible subpopulations, we add a term $\max_i(VI_{i,t})$ in equation (5). If all the best solutions are feasible, then $\max_i(VI_{i,t})$ will be zero.

C) Infeasible to feasible: For any combination i, the best solution was infeasible at generation $t-1$, and it is feasible in generation t, then the improvement index is:

$$I_{i,t} = \max_i(VI_{i,t}) + \left| V_{i,t-1}^{best} + F_{i,t}^{best} - F_{i,t-1}^{bv} \right| \times FR_{i,t} \quad (7)$$

where $F_{i,t-1}^{bv}$= fitness value of the least violated individual in generation t-1.

To assign a higher index value to an individual that changes from infeasible to feasible, we add $V_{i,t-1}^{best}$with the change of

TABLE I

A SELF-ADAPTIVE MULTI-OPERATOR DIFFERENTIAL EVOLUTION (SAMODE)

```
STEP 1: In generation $t = 0$, generate an initial random population of size $PS$. The variables in each individual ($z$) must be within the range as shown below:
    $x_{z,j} = x_{z,j,min} + u \times (x_{z,j,max} - x_{z,j,,min})$
where $x_{z,j,min}, x_{z,j,max}$ are the lower and upper bound for decision variable $x_j$, and $u$ is a random number, $u \in [0,1]$.
STEP 2: Divide $PS$ into four equal subpopulations $p_i$, where $i$ is the $i^{th}$ sub-population. Each subpopulation consists of $n_i$ individuals and each subpopulation has its own operators.
STEP 3: For each DE variant, generate the self-adaptive parameters $F$ and $Cr$ using equations (13) – (16), then generate the offsprings $O_z = \{O_z^1, O_z^2, \dots, O_z^j, \dots, O_z^D\}$ using the equations (9)- (12) and update the FEs.
STEP 4: Sort the individuals in each subpopulation according to their objective function and/or constraint violation.
    If $t \% WS = 0$; then (where % is the division remainder)
        1- Replace the worst 3 individuals in each sub-population with other 3 best solutions (the best solution in each other subpopulation).
        2- If there is any redundant decision vector, then replace it by generating a random vector.
    Else, go to STEP 5.
STEP 5: Store the best individual for each operator, based on objective function and/or constraint violation.
STEP 6:
  If $t > 1$; then
  Update $n_i$ according formula (8).
STEP 7: Stop if the termination criterion is satisfied; else set $t = t + 1$ and go to STEP 3.
```

fitness value in equation (7). We also keep the first term as in equation (5).

It is important to mention that if the problem on hand is an unconstrained problem, that then the improvement index will depend only on equation 5.

After calculating the improvement index for each subpopulation, the subpopulation sizes are calculated according to the following equation:

$$n_{i,t} = MSS + \frac{I_{i,t}}{\sum_{i=1}^{Nopt} I_{i,t}} \times (PS - MSS \times Nopt) \quad (8)$$

where, $n_{i,t}$= the sub-population size for the i^{th} operator at generation t, MSS = the minimum subpopulation size for each operator i at generation t, PS = total number of population, *Nopt* = number of operators

IV. Experimental Results and Analysis

In this section, we present the computational results for the proposed algorithm. The algorithms has been coded using Matlab, and have been run on a PC with a 3.5 GHz Core 2 Duo processor, 3G RAM, and windows XP. As indicated earlier, we have solved the set of real world test problems prepared for a special session on numerical optimization competition in CEC2011[13].

In the literature, the DE variants "*rand/ */ **" performs better, because it finds the new search directions randomly [17]. The investigation by Ting et al. [27] confirmed the benefit of using more than one difference vector (DV) in DE. Interestingly, two or three DV are good enough, but more than this may lead to premature convergence. So we consider the following mutation strategies:

1- rand/ 3:

$$\overrightarrow{V_{z,t}} = x_{best,t} + F_z \times \left(\left(x_{r_1,t} - x_{r_2,t}\right) + \left(x_{r_3,t} - x_{r_4,t}\right) + \left(x_{r_5,t} - x_{r_6,t}\right) \right) \quad (9)$$

2- best/ 3:

$$\overrightarrow{V_{z,t}} = x_{r_1,t} + F_z \times \left(\left(x_{r_2,t} - x_{r_3,t}\right) + \left(x_{r_4,t} - x_{r_5,t}\right) + \left(x_{r_6,t} - x_{r_7,t}\right) \right) \quad (10)$$

3- rand-to- current/ 2:

$$\overrightarrow{V_{z,t}} = x_{r_1,t} + F_z \times \left(\left(x_{r_2,t} - x_{z,t}\right) + \left(x_{r_3,t} - x_{r_4,t}\right) \right) \quad (11)$$

4- rand-to – best and current/ 2:

$$\overrightarrow{V_{z,t}} = x_{r_1,t} + F_z \times \left(\left(x_{best,t} - x_{r_2,t}\right) + \left(x_{r_3,t} - x_{z,t}\right) \right) \quad (12)$$

We use the variants with best value, as in the 2nd variant, because they are popular and they are good choices for both separable and non-separable unimodal test problems [17]. We select the 3rd variant, to get the benefit of the arithmetic recombination and the random effect to investigate the search space that would otherwise be unable to reach the good regions for the non-separable unimodal and multimodal problems [16], while the 4th variant showed encouraging results that were presented in [28].

As indicated earlier, we take advantage of the self-adaptive concept to generate F_z and Cr_z. The calculation of F_z and Cr_z is described below:

- For each individual in the population, generate two Gaussian numbers $N\,(0.5, 0.15)$ one for amplification factor, while the other number represents a crossover rate. The rationale behind using this Gaussian number value is to give more probability to values surrounding 0.5 and hence provide a relatively fair chance for both the perturbed individual and the original individual to be selected as the new offspring [29].
- To generate the new offspring (using equations 9 to 12), for variants rand/3 and best/3, the overall F_z and $Cr_z$are

calculated according to formulas (13) and (14), while the same parameters are calculated according to formulas (15) and (16). Note the difference between amplification values is multiplied by a Gaussian numbers $N\ (0.0, 0.5)$ according to [29].

$$F_z = F_{r_1,G} + N(0,0.5) \times (F_{r_2,G} - F_{r_3,G}) + N(0,0.5) \times (F_{r_4,G} - F_{r_5,G}) + N(0,0.5) \times (F_{r_6,G} - F_{r_7,G}) \qquad (13)$$

$$Cr_z = Cr_{r_1,G} + N(0,0.5) \times (Cr_{r_2,G} - Cr_{r_3,G}) + N(0,0.5) \times (Cr_{r_4,G} - Cr_{r_5,G}) + N(0,0.5) \times (Cr_{r_6,G} - Cr_{r_7,G}) \qquad (14)$$

$$F_z = F_{r_1,G} + N(0,0.5) \times (F_{r_2,G} - F_{r_3,G}) + N(0,0.5) \times (F_{r_4,G} - F_{r_5,G}) \qquad (15)$$

$$Cr_z = Cr_{r_1,G} + N(0,0.5) \times (Cr_{r_2,G} - Cr_{r_3,G}) + N(0,0.5) \times (Cr_{r_4,G} - Cr_{r_5,G}) \qquad (16)$$

We set the parameters: PS = 100, MSS at 10% of PS, WS is equal to 50 generations. 25 independent runs were performed for each test problem, and the stopping criterion was to run up to 150,000 fitness evaluations (FEs). The detailed results (best, median, average, standard deviation (St. d)), for 50,000, 100,000, and 150,000 FEs, are presented in Appendixes A-D.

For constraint handling: the penalty part in the objective functions is handled as follows:

$$\text{PENALTY} = max(0, \text{PENALTY} - tol), \qquad (17)$$

where tol is equal to the penalty value for the top θ-th individual and θ = (0.20 × PS). Note that tol is fixed for the first n_1 generations (here it is equal to 15% of the maximum generations), and then it is adaptively reduced to 0 during the next n_2 generations (here n_2 is equal to 35% of the maximum generations (here it is equal to 525 generations)).

V. Conclusion

During the last few decades, many differential evolution strategies have been introduced to solve optimization problems. Most of these algorithms were designed to use a single crossover and/or a single mutation operator. In this paper, we have shown that the efficiency of evolutionary algorithms can be improved by adopting a concept of self-adaptation with multiple search operators and constraint handling techniques.

In the proposed algorithm (SAMODE), the population was divided into a number of subpopulations, where one subpopulation was allocated to each operator. A meaningful learning strategy was used, that determines the changes of the sub-population sizes at a regular interval. In fact, subpopulation sizes were increased, decreased or kept unchanged, adaptively, depending on the evolution success. An index for measuring the evolution success was also introduced..

References

[1] J. J. Liang and P. N. Suganthan, " Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism," *in proceeding IEEE Congress Evolutionary Computation (CEC2006)*, Vancouver, BC, Jul. 2006, pp. 9-16.

[2] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," *Addison-Wesley Publishing Corporation*, Inc, Reading, MA, 1989.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimisation," *In Proceedings of the IEEE International Conference on Neural Network*, vol. 4, 1995, pp. 1942–1948.

[4] R. Storn and K. V. Price, "Differential evolution–A simple and efficient adaptive scheme for global optimization over continuous spaces," *Technical Report*, Computer Science Institute, University of California at Berkeley, USA, TR-95-012, 1995.

[5] I. Rechenberg, "Evolutions strategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution," *Fromman-Holzboog, Stuttgart,* 1973.

[6] L. J. Fogel, A. J. Owens and M. J. Walsh, "Artificial Intelligence Through Simulated Evolution," *John Wiley & Sons,* New York, 1966.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization*," In IEEE Trans. Evol. Comput.* vol. 1, no. 1, Apr. 1997, pp. 67–82.

[8] R. Mallipeddi, S. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *In Applied Soft Computing*, 2010.

[9] R. Mallipeddi, P. N. Suganthan, "Differential Evolution with Ensemble of Constraint Handling Techniques for solving CEC 2010 Benchmark Problems," *in proceeding in IEEE Congress on Evolutionary Computation*, 2010, pp. 1907-1914.

[10] M. F. Tasgetiren, P. N. Suganthan, Q. Pan, R. Mallipeddi and S. Sarman, "An Ensemble of Differential Evolution Algorithms for Constrained Function Optimization," in proceeding *IEEE CEC*, 2010, pp. 967-975.

[11] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A Three-Strategy Based Differential Evolution Algorithm for Constrained Optimization," *Lecture Notes in Computer Science, Neural Information Processing. Theory and Algorithms*, vol. 6443, 2010, pp.585-592

[12] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization problems," *Computers and Operations Research*, 2011, doi:10.1016/j.cor.2011.03.003.

[13] S. Das and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems," *Tech. Rep.*, December 2010.

[14] R. Storn, and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *in Journal of Global Optimization,* vo. 11, 1997, pp. 341–359

[15] R. Storn, "On the usage of differential evolution for function optimization," In*: Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, IEEE, Berkeley, 1996, pp. 519–523.

[16] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, 2009, pp. 398–417.

[17] E. Mezura-Montes, J. V. Reyes and C. A. C. Coello, "A Comparative Study of Differential Evolution Variants for Global Optimization," *in proceeding Genetic Evol. Comput. Conf. (GECCO)*, 2006, pp. 485–492.

[18] A. Iorio, and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," *in Australian Conference on Artificial Intelligence*, Cairns, Australia, 2004, pp. 861–872.

[19] K. V. Price, R. M. Storn and J. A. Lampinen, "Differential evolution: a practical approach to global optimization," in *Natural Computing* Series, Springer, Berlin, 2005.

[20] R. Gamperle, S. D. Muller and A. Koumoutsakos, "Parameter study for differential evolution," *In proc. WSEAS NNA-FSFS-EC*, Interlaken, Switzerland, 2002, pp. 293–298.

[21] J. Ronkkonen, S. Kukkonen and K. V. Price, "Real parameter optimization with differential evolution," *In proc. IEEE Congress on Evolutionary Computation,* vol. 1. Piscataway, NJ: IEEE Press, 2005, pp. 506–513.

[22] H. Abbass, “The Self-Adaptive Pareto Differential Evolution Algorithm,” In proc. *IEEE Congress on Evolutionary Computation (CEC)*, 2002, pp. 831-836.

[23] D. Zaharie, “Control of population diversity and adaptation in differential evolution algorithms,” *in proceedings of the 9th International Conference on Soft Computing*, Brno, 2003, pp. 41–46.

[24] S. Das, A. Konar and U. K. Chakraborty, “Two improved differential evolution schemes for faster global search,” *in proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, 2005, pp. 991–998.

[25] A. Zamuda, J. Brest, B. Boskovic and V. Zumer, “Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization,” *in IEEE congress Evolutionary Computation,* 2009, pp. 195 - 202.

[26] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-art", IEEE Trans. on Evolutionary Computation, vol. 15, no. 1, 2011, pp. 4-31.

[27] C. Ting and C. Huang, “Varying Number of Difference Vectors in Differential Evolution,” *in proc. IEEE Congress on Evolutionary Computation (CEC)*, 2009, pp. 1351-1358.

[28] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. C. Coello, “Modified differential evolution for constrained optimization,” in *Proc. IEEE Congress on Evolutionary Computation (CEC2006),* 2006, pp. 25–32.

[29] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, ‘‘Self-adaptive differential evolution,’’ *Lecture Notes in Artificial Intelligence*. Berlin, Germany: Springer-Verlag, 2005, pp. 192–199.

APPENDIX A

THE FUNCTION VALUES ACHIEVED VIA SAMODE FOR TEST PROBLEMS (T01: T05)

FEs		T01 FM	T02 L-J	T03BCB	T04 STR	T05 Si(B)	T06 Si(C)
5×10^4	Best	2.5397909E-11	-1.563487E+01	1.151489E-05	1.3770762E+01	-3.060236E+01	-2.914009E+01
	Median	8.9616882E-06	-1.012841E+01	1.151489E-05	1.4329113E+01	-2.323829E+01	-2.026685E+01
	Worst	1.2547264E+01	-7.994769E+00	1.151489E-05	2.0957397E+01	-1.867590E+01	-1.473680E+01
	Mean	3.0769135E+00	-1.075262E+01	1.151489E-05	1.4339590E+01	-2.336857E+01	-2.066230E+01
	St. d.	5.0495423E+00	1.727348E+00	0.000000E+00	1.4046937E+00	2.824114E+00	2.996494E+00
1×10^5	Best	1.6373060E-30	-2.605013E+01	1.151489E-05	1.3770762E+01	-3.663566E+01	-2.916610E+01
	Median	6.9955090E-24	-2.177124E+01	1.151489E-05	1.3835990E+01	-3.156253E+01	-2.742402E+01
	Worst	1.0942277E+01	-1.601185E+01	1.151489E-05	1.4329113E+01	-2.264400E+01	-2.294724E+01
	Mean	1.2120264E+00	-2.203367E+01	1.151489E-05	1.4009624E+01	-3.064824E+01	-2.647254E+01
	St. d.	3.3762168E+00	2.540347E+00	0.000000E+00	2.6986018E-01	3.736597E+00	2.537221E+00
1.5×10^5	**Best**	**0.0000000E+00**	**-2.842253E+01**	**1.151489E-05**	**1.3770762E+01**	**-3.684393E+01**	**-2.916612E+01**
	Median	0.0000000E+00	-2.732327E+01	1.151489E-05	1.3770762E+01	-3.409253E+01	-2.742977E+01
	Worst	1.0942277E+01	-2.610048E+01	1.151489E-05	1.4329113E+01	-3.044259E+01	-2.300593E+01
	Mean	**1.2120256E+00**	**-2.706978E+01**	**1.151489E-05**	**1.3940690E+01**	**-3.359474E+01**	**-2.763470E+01**
	St. d.	3.3762171E+00	6.624811E-01	0.000000E+00	2.5022231E-01	1.575135E+00	1.923527E+00

APPENDIX B

THE FUNCTION VALUES ACHIEVED VIA SAMODE FOR TEST PROBLEMS (T06: T11.2)

FEs		T07 SPRP	T08 TNEP	T09 LSTP	T10 CAAD	T11.1 DED	T11.2 DED
5×10^4	Best	8.205665E-01	2.200000E+02	3.703545E+04	-2.1426120E+01	5.060692E+04	1.054473E+06
	Median	1.273927E+00	2.200000E+02	3.338802E+05	-2.0951694E+01	2.990633E+05	1.109735E+06
	Worst	1.701162E+00	2.200000E+02	4.935347E+05	-1.7588117E+01	8.214131E+05	1.199612E+06
	Mean	1.286197E+00	2.200000E+02	3.306209E+05	-2.0630305E+01	3.371945E+05	1.113047E+06
	St. d.	1.931515E-01	0.000000E+00	7.862102E+04	8.8482937E-01	2.325415E+05	5.266132E+04
1×10^5	Best	5.080677E-01	2.200000E+02	1.603233E+03	-2.1790737E+01	5.737258E+04	1.069534E+06
	Median	9.988410E-01	2.200000E+02	2.958368E+04	-2.1581890E+01	1.063038E+05	1.079238E+06
	Worst	1.331249E+00	2.200000E+02	6.328176E+04	-2.1291989E+01	4.871657E+05	1.138672E+06
	Mean	9.733051E-01	2.200000E+02	3.071812E+04	-2.1582474E+01	1.378632E+05	1.082730E+06
	St. d.	1.785556E-01	0.000000E+00	1.227919E+04	1.2938440E-01	9.487752E+04	1.459006E+04
1.5×10^5	**Best**	**5.000000E-01**	**2.200000E+02**	**1.600299E+03**	**-2.1821665E+01**	**5.098140E+04**	**1.067501E+06**
	Median	8.401270E-01	2.200000E+02	2.321185E+03	-2.1624388E+01	5.273020E+04	1.071150E+06
	Worst	9.943334E-01	2.200000E+02	3.218016E+03	-2.1415837E+01	5.980997E+04	1.072628E+06
	Mean	**8.166238E-01**	**2.200000E+02**	**2.368206E+03**	**-2.1658906E+01**	**5.327838E+04**	**1.070799E+06**
	St. d.	1.193672E-01	0.000000E+00	4.770382E+02	1.1295769E-01	2.031709E+03	1.272746E+03

APPENDIX C

THE FUNCTION VALUES ACHIEVED VIA SAMODE FOR TEST PROBLEMS (T11.3: T11.8)

FEs		T11.3 ELD	T11.4 ELD	T11.5 ELD	T11.6 ELD	T11.7 ELD	T11.8 HS
5×10^4	Best	1.538776E+04	1.801246E+04	3.265246E+04	1.2220462E+05	1.8767501E+06	8.999364E+05
	Median	1.538846E+04	1.809868E+04	3.269273E+04	1.2320932E+05	1.9064872E+06	9.019300E+05
	Worst	1.539000E+04	1.826069E+04	3.277270E+04	1.2422421E+05	1.9201869E+06	9.030068E+05
	Mean	1.538854E+04	1.811387E+04	3.269348E+04	1.2326650E+05	1.9025652E+06	9.018665E+05
	St. d.	4.852291E-01	7.142721E+01	2.746227E+01	4.6079497E+02	1.1420833E+04	7.695111E+02
1×10^5	Best	1.544470E+04	1.798727E+04	3.272766E+04	1.2278751E+05	1.9538641E+06	9.315602E+05
	Median	1.544557E+04	1.810490E+04	3.275331E+04	1.2346686E+05	2.4679356E+06	9.337597E+05
	Worst	1.544982E+04	1.819694E+04	3.281111E+04	1.3887132E+05	4.3992597E+06	9.393771E+05
	Mean	1.544605E+04	1.810514E+04	3.275859E+04	1.2509061E+05	2.6529169E+06	9.342638E+05
	St. d.	1.296112E+00	5.951129E+01	2.124390E+01	3.6601154E+03	6.7436872E+05	2.237141E+03
1.5×10^5	**Best**	**1.544419E+04**	**1.804684E+04**	**3.272195E+04**	**1.2278753E+05**	**1.8891308E+06**	**9.243537E+05**
	Median	1.544422E+04	1.816215E+04	3.273663E+04	1.2406353E+05	1.9596745E+06	9.256328E+05
	Worst	1.544433E+04	1.843584E+04	3.275306E+04	1.2546347E+05	2.3630687E+06	9.269190E+05
	Mean	**1.544423E+04**	**1.818559E+04**	**3.273758E+04**	**1.2409967E+05**	**1.9876716E+06**	**9.258000E+05**
	St. d.	3.733311E-02	1.071408E+02	7.777908E+00	8.0725815E+02	1.2556507E+05	6.715913E+02

APPENDIX D

THE FUNCTION VALUES ACHIEVED VIA SAMODE FOR TEST PROBLEMS (T11.9: T14)

FEs		T11.9 HS	T11.10 HS	T12 (Messenger full)	T13 (Cassini 2)
5×10^4	Best	9.006293E+05	9.0080026E+05	9.5035041E+00	1.145571E+01
	Median	9.020528E+05	9.0205914E+05	1.7903401E+01	1.518932E+01
	Worst	9.018787E+05	9.0313057E+05	2.8853159E+01	2.237408E+01
	Mean	9.018687E+05	9.0201383E+05	1.8219148E+01	1.603435E+01
	St. d.	7.594502E+02	5.6445804E+02	5.5008868E+00	2.928409E+00
1×10^5	Best	9.305771E+05	9.2943047E+05	7.8766563E+00	8.610868E+00
	Median	9.342159E+05	9.3344461E+05	1.1630885E+01	1.113643E+01
	Worst	9.415123E+05	9.3626902E+05	1.7799026E+01	1.682923E+01
	Mean	9.347829E+05	9.3365580E+05	1.2052303E+01	1.186582E+01
	St. d.	2.375813E+03	1.8676930E+03	2.9684533E+00	2.267839E+00
1.5×10^5	**Best**	**9.243233E+05**	**9.2437627E+05**	**6.9432150E+00**	**8.610634E+00**
	Median	9.261120E+05	9.2576546E+05	1.0864857E+01	1.010574E+01
	Worst	9.270246E+05	9.2725380E+05	1.5618800E+01	1.662200E+01
	Mean	**9.260222E+05**	**9.2584595E+05**	**1.1067471E+01**	**1.099524E+01**
	St. d.	7.297982E+02	7.6154770E+02	2.6522779E+00	2.388975E+00