Enhanced Multi-operator Differential Evolution for Constrained Optimization

Saber Elsayed*, Ruhul Sarker* and Carlos Coello Coello†
*School of Engineering and Information Technology
University of New South Wales at Canberra
Canberra 2600, Australia
Email: {s.elsayed, r.sarker}@adfa.edu.au
†Depto. de Computación, CINVESTAV-IPN
Mexico
Email: ccoello@cs.cinvestav.mx

***Abstract*—Over the last two decades, many differential evolution algorithms have been introduced to solve constrained optimization problems. Due to the variability of characteristics of such problems, no single algorithm performs consistently well over all of them. In this paper, for a better coverage of the problem characteristics, we introduce an enhanced multi-operator differential evolution algorithm, which utilizes the strengths of multiple search operators at each generation, and places more emphasis on the best-performing ones during the optimization process based on three measures: (1) the quality of solutions; (2) the feasibility rate; and (3) diversity. In addition, an improved self-adaptive mechanism for automatically controlling the scaling factor and crossover rate is proposed. The performance of the algorithm is assessed using a well-known set of constrained problems, with the experimental results demonstrating that it is superior to state-of-the-art algorithms.**

I. Introduction

Many real-world decision processes involve solving optimization problems, that is, finding the best solutions in their feasible search spaces which, as they are bounded by constraint functions, are recognized as constrained optimization problems (COPs). A COP may contain different types of variables, and may have equality and/or inequality constraints with different mathematical properties. The feasible region of such a problem can be either a tiny or a significant portion of the search space and, moreover, either one single bounded region or a collection of multiple disjoint regions or, in some practical problems, even unbounded. Also, a large number of variables and constraints may add further complexity to the solution of COPs [1]. These different characteristics have made COPs a challenging research area in the optimization domain. In this paper, single-objective COPs are considered, with a COP formally expressed as:

$$\begin{aligned} &\text{minimize } f(\overrightarrow{x}) \\ &\text{subject to: } g_k(\overrightarrow{x}) \leq 0,\ k = 1, 2, \ldots, K \\ &h_e(\overrightarrow{x}) = 0,\ e = 1, 2, \ldots, E \\ &\underline{x}_j \leq x_j \leq \overline{x}_j,\ j = 1, 2, \ldots, D \end{aligned} \tag{1}$$

where $\overrightarrow{x} = [x_0, x_2, ..., x_D]$ is a vector with D decision variables, $f(\overrightarrow{x})$ the objective function, $g_k(\overrightarrow{x})$ the k^{th} inequality constraint, $h_e(\overrightarrow{x})$ the e^{th} equality constraint and each x_j has a lower limit $(\underline{x}_j)$ and an upper limit $(\overline{x}_j)$.

Over the years, evolutionary algorithms (EAs) have been used by researchers and practitioners to solve COPs. From the several EAs currently available, differential evolution (DE) [2] has become very popular, due to its good performance [3]. However, like any other EA, one DE operator and/or a set of parameters well suited for a certain set of problems may not work well for another one [1]. In addition, even if a set of parameters or operators works well during the early stages of the evolutionary process, it may not perform well in later ones and vice versa. Due to its importance for improving the performance of DE, many research studies have proposed different ways of self-adaptively managing the control parameters and/or a mix of operators[4, 3, 1]. However, proposing such techniques in the context of constrained optimization needs careful designs, and further research is then required.

For instance, Elsayed et al. [1] recently proposed a self-adaptive multi-operator DE (SAMO-DE) algorithm that dynamically places emphasis on the best-performing DE variants based on the quality of the fitness values and/or constraint violations. This approach was found to outperform state-of-the-art algorithms. However, incorporating further components, such as diversity, into this algorithm could enhance its performance. Also, considering the recent proposals in adapting DE control parameters, Tanabe and Fukunaga [5] proposed a mechanism that generates new control parameter settings based on some distributions around a single pair of parameters using historical memories. However, it directly used the raw values of the fitness improvements of successful individuals generated, which could affect the robustness of it, as such improvements may vary from one individual to another considering how close it is to the optimal solutions (in reality, the optimal solutions are not known in advance). In addition, extending the algorithm for solving COPs is not straight froward, as a careful improvement measure should be utilized by considering improvements in the feasible and infeasible regions. It is worth mentioning that, in the literature, there are other self-adaptive mechanisms which use a *counter* to record the number of successful individuals generated by a specific combination of parameters, and then the ones with higher probabilities are

highly likely to be selected. However, such mechanisms do not quantify the number of improvements in the quality of solutions.

Motivated by these gaps, the main objective of this paper is to enhance the performance of SAMO-DE in solving COPs, by incorporating (1) an improved self-adaptive mechanism for managing DE control parameters; (2) a new improvement measure, which uses the diversity plus the quality of solutions and feasibility rates, to place emphasis on the best-performing DE operators. To avoid deteriorating the algorithm's performance, this method avoids using the raw improvement values; (3) a simple information sharing mechanism, which removes the window size parameter introduced in the original SA-MODE; (4) more powerful DE operators; and (5) a linear reduction of the population size.

Considering the above mentioned points, the proposed algorithm starts by dividing the initial population into several sub-populations, each of which evolves using its own DE variant. Based on the improvement index of each operator, the sub-population size varies adaptively. At the end of every generation, all sub-populations are gathered and then redivided based on the new sub-population sizes. In addition, a linear reduction of the population size is carried out, in which the population size is set to a large value at the start of the evolutionary process, and then linearly reduced to a small value.

The performance of the proposed algorithm was tested on a well-known set of 36 constrained test problems [6] (18 with 10 dimensions (10D) and 18 with 30D) with different mathematical properties. The quality of solutions and statistical test results proved the superiority of the algorithm to several state-of-the-art algorithms. It must be mentioned here that existing algorithms are able to solve the 10D problems with a reasonable accuracy, but many of them are not performing well for the 30D problems.

The rest of this paper is organized as follows: an overview of DE and its operators and parameters is provided in Section II; the proposed algorithm is illustrated in Section III; and the experimental results and conclusions are discussed in Sections IV and V, respectively.

II. Differential Evolution

DE is a population-based stochastic algorithm [7]. Its algorithmic steps start with a set of random vectors $(X = \{\overrightarrow{x}_1, \overrightarrow{x}_2, ... \overrightarrow{x}_{PS}\})$, where PS is the population size, each of which should be within the search domain. Then, a mutant population $(V = \{\overrightarrow{v}_1, \overrightarrow{v}_2, ... \overrightarrow{v}_{PS}\})$ is generated. Subsequently, every $\overrightarrow{x}_z$ is recombined with its corresponding $\overrightarrow{v}_z$ to generate a trial vector $\overrightarrow{u}_z$, where $z = \{1, 2, ..., PS\}$. A pairwise comparison between $\overrightarrow{x}_z$ and $\overrightarrow{u}_z$, with the winning vectors is performed, based on the fitness values and/or constraints violation, and this generates the population at the next generation X_{t+1}, (t is the generation number) [8]. Next, we provide a brief description of each of the above steps.

- **Initialization:** Each individual in the population is represented as a D-dimensional vector in which each variable is generated within its boundaries:

$$x_{z,j} = \underline{x}_j + rand_j(0,1) \times (\overline{x}_j - \underline{x}_j) \, \forall \, j = 1, 2, ...D \tag{2}$$

where $rand_j(0,1)$ is a uniform random number within $[0,1]$ [8].

- **Mutation:** In this step new solutions that are perturbations of the current ones in which, in its simplest form (DE/rand/1), $\overrightarrow{v}_z$ is generated by adding a scaled difference between two random vectors to a third one (equation 3).

$$\overrightarrow{v}_z = \overrightarrow{x}_{r_1} + F \times (\overrightarrow{x}_{r_2} - \overrightarrow{x}_{r_3}) \tag{3}$$

where $\overrightarrow{x}_{r_1}$, $\overrightarrow{x}_{r_2}$ and $\overrightarrow{x}_{r_3}$ are distinct solution vectors in the current population and none similar to $\overrightarrow{x}_z$, F a positive real number that controls the rate at which the population evolves [8]. Also, $\overrightarrow{x}_{r_1}$ is called the *base vector*.
Over the last two decades many variations of this operator have been introduced. For more details, readers are referred to [9].

- **Crossover:** Two well-known crossover schemes (binomial and exponential), exist in the literature. The former, which sometimes is called uniform or discrete [8], is conducted on every $j \in [1, D]$ with a predefined crossover probability. In particular, for each j, a uniform random number $(rand_j(0,1))$ is generated. If its value is less than Cr, the value of $\overrightarrow{u}_{z,j}$ will be copied from the corresponding value from $\overrightarrow{v}_{z,j}$; otherwise, it will be equal to $\overrightarrow{x}_{z,j}$:

$$u_{z,j} = \begin{cases} v_{z,j} & if \, (rand_j(0,1) \leq cr \text{ or } j = j_{rand}) \\ x_{z,j} & \text{otherwise} \end{cases} \tag{4}$$

where $j_{rand} \in 1, 2, ..., D$ is a random integer index which ensures that $\overrightarrow{u_z}$ obtains at least one component from $\overrightarrow{v_z}$.
On the other hand, exponential crossover is similar to a two-point crossover in which the first cut point (l) is randomly selected from a range $[1, D]$ and the second is determined such that L components are copied from $\overrightarrow{v}_z$ [10] as:

$$u_{z,j} = \begin{cases} v_{z,j} & \forall j = \langle l \rangle_D, \langle l+1 \rangle_D, ..., \langle l+L-1 \rangle_D \\ x_{z,j} & \text{otherwise} \end{cases} \tag{5}$$

where $\langle l \rangle_D$ denotes a modulo function with a modulus of D and $L \in [1, D]$.

- **Selection:** DE uses a simple one-to-one survivor selection in which, at generation (t), $\overrightarrow{u}_{z,t}$ competes against $\overrightarrow{x}_{z,t}$, and the better, in terms of the fitness value and/or constraint violation, is considered a vector in the new population at the next generation $(t+1)$.

A. Operators and parameters: a brief review

In this subsection, a brief review of recent studies on the adaptation of DE operators and parameters is provided.

1) Multi-operator DE: Over the last two decades, a great deal of work has been undertaken on dealing with DE search operators and parameters. However, a general conclusion accepted among researchers is that no single DE operator and/or parameter is suitable for all optimization problems. Thus, researchers have been motivated to propose multi-operator DE and multi-method frameworks, and some of these methods are described below.

Tasgetiren and Suganthan [11] proposed a multi-population DE algorithm for solving real-parameter COPs in which each sub-population conducted its search in parallel with a re-grouping schedule after certain predefined number of generations. They applied two mutation operators (DE/rand/1 and DE/best/1) with a fixed probability of 0.5 to each. Although the algorithm produced encouraging results, it was not competitive with respect to some other DE variants. Also, no adaptive learning mechanism was used to update the probability of selecting each operator. Mallipeddi et al. [12, 13] proposed a framework that used a mix of mutation strategies and discrete control parameters in DE to solve unconstrained problems. In it, a pool of different mutation strategies, along with a pool of values for each control parameter, coexisted during the entire evolutionary process and competed to produce new individuals. The authors then extended this work to use a mix of four CHTs based on a DE algorithm (ECHT-DE) [14, 13] to solve COPs in which four populations were initialized, each using a different CHT. The algorithm [15] ranked second in the CEC2010 competition for solving COPs. However, using discrete values of F and Cr implies that many values could be ignored, which could improve the algorithm's performance. In addition, no adaptation mechanism was used to select F and Cr.

As an extension of [1], Elsayed et al. [16] proposed two novel DE variants, each of which utilized the strengths of multiple mutation and crossover operators to solve 60 constrained problems, producing results superior to those from state-of-the-art algorithms. In those algorithms, F and Cr were calculated by some DE mutation strategies. However, the multi-operator algorithms could be improved by considering (1) the diversity issue in the selection process; and (2) a more powerful self-adaptive mechanism for managing F and Cr, which will be described in detail later on.

In the unconstrained domain, several DE algorithms have been introduced, a few of which are discussed here. Qin et al. [4] proposed a self-adaptive DE algorithm (SaDE) where F and Cr did not have to be pre-specified. In addition, four mutation strategies were used and each individual in the population were assigned to one of them based on a given probability. Then, the selection probability of each operator was updated based on its success and failure rates during previous generations (a learning period). Although this algorithm showed a good performance for solving a set of unconstrained problems, the diversity issue was not incorporated within the framework. In addition, the self-adaptive mechanism to control F and Cr was not efficient enough. Zamuda and Brest [17] introduced an algorithm that employed two mutation strategies in jDE [18], with the population size adaptively reduced during the evolutionary process based on their earlier technique proposed in [19]. The algorithm was tested on 22 real-world applications and performed better than two other algorithms. It was then extended by embedding a self-adaptation mechanism for parameter control [20], whereby the three DE strategies were used, each of which was applied to a specific group of individuals based on predefined parameters. In addition, if an individual did not improve for a predefined number iterations, then it would be re-initialized with a probability of 0.1. The algorithm was tested on a set of unconstrained problems, but it was outperformed by other algorithms. In addition, no adaptive learning method was considered. An improved adaptive DE (ACDE) algorithm [21], in which four mutation strategies were used in a sequential manner, i.e., one mutation at every predefined number of generations, while F and Cr were updated using a Cauchy distribution, and with mean values based on previously found successful values. In addition, a mechanism was used to reduce the population size. The concern here is that such a sequential ordering of the mutation operators may not perform consistently over a wide range of problems. In addition, the weight of every successful F and Cr should be considered in the adaptation mechanism.

2) DE parameters: As previously mentioned, because the selection of DE parameters plays a pivotal role in its success, many researchers have proposed techniques for adapting them. In [2], it was suggested that PS be within [5D,20D] and F be set to a value of 0.5 . In another research study [22], setting $F \in [0.4, 0.95]$ was recommended. Self-adapting F using DE operators was introduced in [23] and then modified in [1]. In [4], F was generated using a normal distribution, $(N(0.5, 0.3))$ was randomly generated using an independent normal distribution, with its mean initially set to a value of 0.5 and its standard deviation was fixed at 0.1. Then, Cr was re-generated after a predefined number of generations.

Tvrdíkand and Poláková [24] introduced a DE algorithm for solving a set of COPs. In it, with a predefined probability, one set of control parameters was selected from 12 available sets and, during the evolutionary process, the probability of selecting each set was updated based on its success rate in the previous steps. The algorithm was evaluated using a set of unconstrained problems and showed to have competitive performance [25]. However, using the success rate of each set, might be insufficient, and the improvement in the fitness function and/or constraint violation should be taken into consideration.

Liu and Lampinen [26] proposed to self-adaptively control F and Cr using the concept of fuzzy logic. Brest et al. [18] introduced a self-adaptation method for F and Cr in which each individual in the population was assigned a different combination of their values. Zhang et al. [27] proposed an adaptive DE (JADE) in which, at each generation, Cr_z was independently generated, $N(\overline{Cr}, Cr_\sigma = 0.1)$ with $\overline{Cr}$ initially set to a value of 0.5 and then dynamically updated. Similarly, F_z was generated according to a Cauchy distribution with a location parameter ($\overline{F}$), the initial value of which was 0.5. They adopted a scaling parameter of 1 and, at the end of each generation, $\overline{F}$ was updated. As an improved version of JADE, Tanabe and Fukunaga [5, 28] proposed a success-history-based adaptive DE (SHADE) in which, instead of generating new control parameter settings based on some distribution around a single pair of parameters $(\overline{Cr}, \overline{F})$, historical memories (M_{Cr}, M_F) which stored sets of values of Cr and F, respectively, were adopted. As it performed well at earlier

generations, it generated new pairs of Cr and F by directly sampling the parameter space close to one of the stored pairs. This algorithm was tested on the CEC2013 unconstrained problems and performed better than state-of-the-art algorithms. However, it has some concerns, as discussed in Section I.

III. Enhanced Multi-operator DE

In this section, details of the proposed enhanced multi-operator DE algorithm (E-MODE) are discussed.

A. E-MODE

As previously discussed, it has been proved that the relative performance of a DE operator may vary during the evolutionary process, that is, one operator may work well in the early (or some early) stages of the search process and perform poorly in later ones, or vice versa [1]. Also, a DE operator may work well for a specific problem but badly for another. This encourages the use of multi-operator DE, bearing in mind that more emphasis should be placed on the better-performing ones at each stage of the evolutionary process. The main steps in E-MODE are presented in Algorithm 1.

E-MODE starts with a random initial population of size PS, i.e., $X = \{\overrightarrow{x}_1, \overrightarrow{x}_2, ..., \overrightarrow{x}_z, ..., \overrightarrow{x}_{PS}\}$. Then, each individual is evaluated, and the number of current fitness evaluations (cfe) is increased by 2, as evaluating the constraints is counted as one fitness evaluation[1]. Then, the entire population is randomly divided into nps sub-populations of equal size, i.e., $X^i = \{\overrightarrow{x^i}_1, \overrightarrow{x^i}_2, ..., \overrightarrow{x^i}_{ps_i}\}$, where $\sum_{i=1}^{nps} ps_i = PS$. For every sub-population, new trial individuals, u^i, are generated using the assigned DE operators. It is worth mentioning that, to reduce the number of fitness evaluations, if $\overrightarrow{u_z^i}$ is infeasible, the objective value is not calculated and, as it takes the fitness value of its parent, cfe is only increased by 1. On the other hand, if $\overrightarrow{u_z^i}$ is feasible, as its fitness value is calculated, cfe is increased by 2. A pairwise comparison between every individual in X^i and its corresponding one in u^i is then carried out, and the winner survives to the next generation (see Section III-B). Subsequently, the improvement in each DE is calculated, as discussed in Section III-C. Based on the improvement index, the sub-population 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 optimization process, we set a minimum sub-population size to each operator.

To share information among the nps operators, at the end of every generation, all sub-populations are gathered in X. Then, at generation $t+1$, they are re-divided into nps sub-populations based on the new ps values, ensuring that the best nps individuals in X are not within the same X^i. Note that this information exchanging procedure removes the window size parameter introduced in [1].

In order to maintain diversity at early generations, while enhancing the exploitation ability in later ones [29], a linear reduction of PS is carried out at the end of each generation by removing one from the worst 5% individuals, such that

[1]This was a condition in the CEC2010 competition [6], which is used in this paper to assess E-MODE.

Algorithm 1 General framework of E-MODE

1: At $t = 1$, generate initial PS random individuals (X). The variables of each individual $(\overrightarrow{x_z})$ must be within their boundaries;
2: Calculate the fitness value and constraint violations of $(\overrightarrow{x_z})$ $\forall z = 1, 2, ..., PS$;
3: $cfe \leftarrow 2 \times PS$, as evaluating the constraints is counted;
4: Divide X into nps sub-populations, i.e., $\sum_{i=1}^{nps} ps_i = PS$;
5: **while** $cfe < FFE_{max}$ **do**
6: Randomly distribute X over $X_i, \forall i = 1, 2, ..., nps$, where every X_i is of size ps_i; and make sure that there is no more than one individual, from the best nps solutions, in the same sub-population.
7: **for** $i = 1$ to nps **do**
8: Generate new offspring using the i^{th} DE variant;
9: Update cfe;
10: Calculate the improvement index $(I_{i,t})$ (Section III-C);
11: Self-adaptively manage F and Cr (Section III-D);
12: **end for**
13: Calculate new $ps_{i,t+1}, \forall i = 1, 2, ..., nps$ (equation (12));
14: Update PS_{t+1} (equation (6));
15: $t \leftarrow t + 1$; and go to step 5;
16: **end while**

$$PS_{t+1} = \text{round}\left(\left(\left(\frac{PS_{min} - PS_{max}}{FFE_{max}}\right) \times cfe\right) + PS_{max}\right) \quad (6)$$

where PS_{max} and PS_{min} are the maximum and minimum values of PS, respectively, and FFE_{max} is the maximum number of fitness evaluations.

The algorithm continues until the stopping criterion is satisfied.

B. Selection process

The selection process between any offspring and its parent follows one of three scenarios [30]: (1) for two feasible solutions, the fittest one (according to the fitness value) is selected; (2) a feasible point is always better than an infeasible one; and (3) for two infeasible solutions, the one with a smaller sum of constraint violations (ψ) is chosen, where ψ of an individual $(\overrightarrow{x_z})$ is calculated based on equation (7).

$$\psi_z = \sum_{k=1}^{K} \max\left(0, g_k(\overrightarrow{x_z})\right) + \sum_{e=1}^{E} \max\left(0, |h_e(\overrightarrow{x_z})| - \epsilon_e\right) \quad (7)$$

where $g_k(\overrightarrow{x_z})$ and $h_e(\overrightarrow{x_z})$ are the k^{th} inequality and e^{th} equality constraints, respectively. Every equality constraint (h_e), ϵ_e is initialized with a large value and then reduced to 0.0001. Setting the initial value of ϵ is problem dependent, as indicated in [31, 32].

C. Improvements measure

At any given generation, for measuring the improvement in each operator, the feasibility rate, quality of solutions and

diversity criteria are considered. Generally, for any generation $(t > 1)$, one of the following three scenarios arises:

1) ***Infeasible to infeasible***: the best solution in sub-population (X_i) is infeasible in both generations $t-1$ and t.
2) ***Infeasible to feasible***: the best solution in sub-population (X_i) is infeasible at generation $t-1$ and becomes feasible at generation t.
3) ***Feasible to feasible***: the best solution in sub-population (X_i) is feasible at both generations, $t-1$ and t.

In the first scenario, the improvement index of operator $i(I_{i,t})$ at generation t is

$$I_{i,t} = \max\left(0, \frac{\psi\left(\overrightarrow{x^i}_{best,t-1}\right) - \psi\left(\overrightarrow{x^i}_{best,t}\right)}{\psi\left(\overrightarrow{x^i}_{best,t-1}\right)}\right) + \max\left(0, \frac{f\left(\overrightarrow{x^i}_{best,t-1}\right) - f\left(\overrightarrow{x^i}_{best,t}\right)}{\left|f\left(\overrightarrow{x^i}_{best,t-1}\right)\right|}\right) \quad (8)$$

As the main objective of any COP is to find a feasible (nearly) optimal solution, any improvement in feasibility is always considered better than that producing an infeasible solution, i.e., variants that fall in scenarios 2 or 3 are ranked on top of those in the first one.

$$I_{i,t} = max(I_{i,t}) + \left(\frac{\psi\left(\overrightarrow{x^i}_{best,t-1}\right) - \psi\left(\overrightarrow{x^i}_{best,t}\right)}{\psi\left(\overrightarrow{x^i}_{best,t-1}\right)} + \max\left(0, \frac{f\left(\overrightarrow{x^i}_{best,t-1}\right) - f\left(\overrightarrow{x^i}_{best,t}\right)}{\left|f\left(\overrightarrow{x^i}_{best,t-1}\right)\right|}\right)\right) \times FR_{i,t} \quad (9)$$

where $FR_{i,t}$ is the feasibility rate (the fraction of feasible solutions in X_i) of the i^{th} operator at t. In case that $\psi\left(\overrightarrow{x}_{best,t-1}\right)$ or $f\left(\overrightarrow{x}_{best,t-1}\right)$ is zero, the corresponding denominator component is removed. Note that equations (8) and (9) use the improvement rates in fitness values and violation, instead of raw values, as in [1]. Also, in cases 2 and 3, $\psi\left(\overrightarrow{x}_{best,t}\right)$ is always 0.

The average diversity rate of every X_i is calculated by

$$div_i = \frac{\sum_{z=1}^{ps_i} d\left(\overrightarrow{x^i}_{z,t}, \overrightarrow{x^i}_{best,t}\right)}{ps_{i,t}} \forall\, i = 1, 2, .., nps \quad (10)$$

where $d\left(\overrightarrow{x^i}_z, \overrightarrow{x^i}_{best}\right)$ is the Euclidean distance between the z^{th} individual and the best individual in the i^{th} sub-population.

Then, the improvement index is updated such that

$$I_{i,t} = (1-c) \times \frac{I_{i,t}}{\sum_{i=1}^{nps} I_{i,t}} + c \times \frac{div_{i,t}}{\sum_{i=1}^{nps} div_{i,t}} \quad (11)$$

where c is a parameter that controls the contribution of the diversity component in the selection process. It is set to a value of 0.5 at the beginning of the evolutionary process (exploration) and then linearly reduced to 0 at the end (exploitation).

Finally, ps_i at generation $t+1$ is calculated as

$$ps_{i,t+1} = MSS + \frac{I_{i,t}}{\sum_{i=1}^{nps} I_{i,t}} \times (PS - MSS \times nps) \quad (12)$$

where MSS is the minimum sub-population size assigned to every operator.

D. Adaptation of F and Cr

As previously mentioned, Tanabe et al. [5] proposed a mechanism for self-adaptively controlling F and Cr. In this paper, an enhanced version is proposed to deal with COPs. To do this, we have to consider all of the three scenarios mentioned above, assigning higher ranks to those individuals which fall in cases 2 and 3 than those in case 1. Additionally, to avoid deteriorating the algorithm's robustness by using the raw fitness and/or total constraint violations improvements, the fitness and/or total constraint violations improvements rates are considered, which adds another difference between the proposed method with that in [5]. The proposed self-adaptive method is as follows:

1) A historical memory with H entries for both parameters (M_{Cr}, M_F) is initialized, where all values are set to a value of 0.5.
2) Each individual $\overrightarrow{x}_z$ is associated with its own Cr_z and F_z

$$Cr_z = \text{randni}(M_{Cr,r_z}, \sigma_{cr}) \quad (13)$$

$$F_z = \text{randci}(M_{F,r_z}, \sigma_F) \quad (14)$$

where r_z is randomly selected from $[1, H]$. randni(μ, σ) and randci(μ, σ) are values randomly selected from the normal and Cauchy distributions with mean μ and variance σ, $\sigma_F = \sigma_{cr} = 0.1$.
3) At the end of each generation, Cr_z and F_z used by the successful individuals are recorded in S_{Cr} and S_F and then the contents of memory are updated as follows:

$$M_{Cr,d} = mean_{WA}\left(S_{Cr}\right)\ if\ S_{Cr} \neq \text{null} \quad (15)$$

$$M_{F,d} = mean_{WL}\left(S_F\right)\ if\ S_F \neq \text{null} \quad (16)$$

where $1 \leq d \leq H$ is the position in the memory to be updated. It is initialized to 1, and then incremented whenever a new element is inserted into the history, and if it is greater than H, it is set to 1. mean$_{WA}(S_{Cr})$ and mean$_{WL}(S_F)$ are computed as follows:

$$\text{mean}_{WA}(S_{Cr}) = \sum_{\gamma=1}^{|S_{Cr}|} w_\gamma . S_{cr,\gamma} \quad (17)$$

$$\text{mean}_{WL}(S_F) = \frac{\sum_{\gamma=1}^{|S_F|} w_\gamma . S_{F,\gamma}^2}{\sum_{\gamma=1}^{|S_F|} w_\gamma . S_{F,\gamma}} \quad (18)$$

where

$$w_\gamma = \frac{\beta_\gamma}{\sum_{\gamma=1}^{|S_{Cr}|} \beta_\gamma} \tag{19}$$

and β_γ is calculated as follows:

- Firstly, for every successful individual $(\gamma \in 1, 2, ... |S_{Cr}|)$[2]which falls in scenario 1, its improvement I_γ is

$$\beta_\gamma = I_\gamma = \max\left(0, \frac{\psi_{\gamma,t-1} - \psi_{\gamma,t}}{\psi_{\gamma,t-1}}\right) + \max\left(0, \frac{f_{\gamma,t-1} - f_{\gamma,t}}{|f_{\gamma,t-1}|}\right) \tag{20}$$

- Then, for every successful individuals $(\gamma \in 1, 2, ... |S_{Cr}|)$ which falls in scenario 2 or 3, its improvement I_γ is

$$\beta_\gamma = \max(0, I_\gamma) + \frac{\psi_{\gamma,t-1} - \psi_{\gamma,t}}{\psi_{\gamma,t-1}} + \max\left(0, \frac{f_{\gamma,t-1} - f_{\gamma,t}}{|f_{\gamma,t-1}|}\right) \tag{21}$$

Note that to avoid dividing a value by zero, the corresponding denominator component is removed. Also, the above mentioned steps are carried out to every sub-population separately.

IV. Experimental Results

In this section, the computational results obtained by E-MODE for the set of CEC2010 constrained problems [6] are presented and analyzed. The algorithm was run 25 times for each test problem for up to $FFE_{max} = 200,000$ and $FFE_{max} = 600,000$ FEs for the 10D and 30D problems, respectively.

Regarding the parameter values used in this study: $nps = 2$. The first sub-population used DE/rand-to-*p*best with archive/1/bin (equation 22), and the second one used DE/current-to-*p*best with archive/1/bin (equation 23), such that

$$u^1_{z,j} = \begin{cases} x^1_{r_1,j} + F_z.(x^i_{\phi,j} - x^1_{r_1,j} + x^1_{r_2,j} - \widetilde{x}_{r_3,j}) \\ \qquad if(rand \leq cr_z \text{ or } j = j_{rand}) \\ x^2_{z,j} \qquad \text{otherwise} \end{cases} \tag{22}$$

$$u^2_{z,j} = \begin{cases} x^2_{z,j} + F_z.(x^i_{\phi,j} - x^2_{z,j} + x^2_{r_1,j} - \widetilde{x}_{r_3,j}) \\ \qquad if(rand \leq cr_z \text{ or } j = j_{rand}) \\ x^2_{z,j} \qquad \text{otherwise} \end{cases} \tag{23}$$

where $r_1 \neq r_2 \neq r_3 \neq z$ are random integer numbers, with $\overrightarrow{x}_{r_1}$ and $\overrightarrow{x}_{r_2}$ randomly selected from x^i, $x^i_{\phi,j}$ was selected from the best 10% individuals in x^i [27], while $\widetilde{x}_{r_3,j}$ was chosen from the union of the entire X and archive AR. Initially, the archive was empty. Then, parent vectors which failed in the selection process were added to it and, once its size exceeded a threshold, $1.4PS$, randomly selected elements were deleted to make space for the newly inserted ones [27].

PS_{max} had 150 individuals and PS_{min} had 40, which was equal to the minimum value that could be used, as $MSS = 0.1 \times PS$ [1] and at least 4 individuals were required to carry out the mutation operator (equation 22). $H = 30$, F_z was set within $[0.4 - 1.0]$ and $Cr_z \in [0.2 - 1.0]$, as recommended in [3].

[2]$|S_{Cr}|$ is the number of successful Cr recorded in S_{Cr}, and $|S_{Cr}|$=$|S_F|$

A. Results and comparison with state-of-the-art algorithms

Firstly, from the results obtained, as shown in Appendix A, E-MODE was able to attain a 100% feasibility rate (FR) for all the problems with both 10D and 30D. Considering the quality of solutions obtained, for the 10D problems, it was found that E-MODE was successfully able to obtain the best-known solutions [1] for all test problems, except for C12. Although the algorithm got stuck in local solutions several times for C02, C08 and C12, its performance was robust for all the other test problems.

Considering the 30D problems, E-MODE performed well, as it was able to obtain the best known solutions. Also, its performance was remarkably improved in solving C08. However, its performance for C02 was not as expected. After an investigation, we found that the reason was related to the boundary limits set for F and Cr. By changing both of them to $[0 - 1]$ in that particular problem, E-MODE was able to obtain much better results. However, for two multi-modal problems (C01 and C13), the algorithm got stuck in local optima. Nevertheless, the quality of such local optima was not far from the best-known solutions.

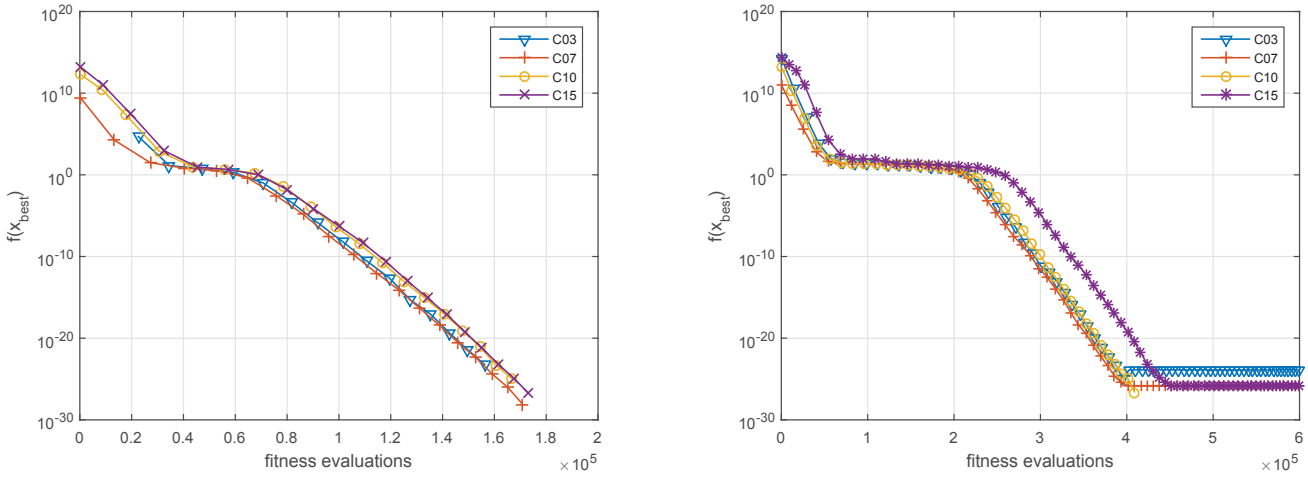
Considering the number of $FFEs$ taken to attain $\left|f(\overrightarrow{x}_{best}) - f\left(\overrightarrow{x^*}\right)\right| \leq 0.0001$, where $\overrightarrow{x^*}$ is the best known solutions (Table I), it was noted that E-MODE had the ability to converge quickly to the best-known solutions. A few plots are also depicted in Fig. 1. These plots show that the E-MODE is able to converge near-optimal solutions very quickly, i.e., after 20% and 30% of the maximum number of fitness evaluations for the 10D and 30D problems, respectively, then gradually fine-tune the solution until it reaches the optimality.

Table I. Average number of cfe to attain $\left|f(\overrightarrow{x}_{best}) - f\left(\overrightarrow{x^*}\right)\right| \leq 0.0001$

Problem	cfe		Problem	cfe	
	10D	30D		10D	30D
C01	5.688E+04	3.740E+05	C10	9.000E+04	2.561E+05
C02	9.401E+04	5.242E+05	C11	1.083E+05	3.111E+05
C03	8.327E+04	2.618E+05	C12	2.000E+05	6.000E+05
C04	8.179E+04	2.452E+05	C13	1.620E+05	6.000E+05
C05	1.200E+05	3.672E+05	C14	9.720E+04	2.563E+05
C06	1.265E+05	3.751E+05	C15	9.270E+04	2.632E+05
C07	8.299E+04	2.497E+05	C16	7.389E+04	5.831E+04
C08	1.901E+05	2.616E+05	C17	7.168E+04	2.500E+05
C09	8.869E+04	2.557E+05	C18	8.054E+04	3.469E+05

E-MODE was also compared with four state-of-the-art algorithms: (1) SAMO-DE [1]; (2) DE with dynamic parameters selection (DE-DPS) [3]; (3) εDEag [33] (the winner of the CEC2010 competition); and (4) an adaptive ranking mutation operator-based DE (ECHT-ARMOR-DE) [34], which used an ensemble of CHTs as well as operators and parameters. Note that, as previously mentioned, many of the existing approaches are able to produce good results for the 10D problems. However, their performance deteriorates when solving the 30D ones.

Firstly, E-MODE was able to reach 100% FRs for both the 10D and 30D problems. εDEag attained 100% and 95.1%FRs



(a) 10D (b) 30D

Figure 1. Convergence plots of E-MODE for C03, C07, C10 and C15 with 10D and 30D and seed/run $= 1$

for the 10D and 30D problems, respectively, while the FRs of ECHT-DE and ECHT-ARMOR-DE were less than 100% for the 10D and 30D problems, with no exact percentages reported in their papers.

Regarding the quality of solutions achieved, a comparison summary is provided in Table II. For the 10D problems, it was noticed that all algorithms were almost the same, as they obtained the best-known solutions, but for the average results, E-MODE was able to obtain better results for more problems than all the other algorithms. Considering the 30D problems, E-MODE was found to be superior to all the other algorithms.

The Wilcoxon test was carried out and its results are reported in Table II. From this table, no significant difference was noticed in the 10D problems. However, for the 30D problems, it was clear that E-MODE statistically outperformed all the other algorithms. Furthermore, the Friedman test was used to rank all algorithms based on the average fitness values obtained, as shown in Table III. The results revealed that E-MODE ranked 1^{st}, followed by DE-DPS. Although SAMO-DE was ranked last based on the 10D average results, it ranked 3^{rd} in the 30D problems.

V. Conclusions

Due to the importance of COPs in real-world applications, over the last two decades, many DE algorithms have been introduced to solve them. However, no single operator and/or a combination of parameters was able to solve a wide range of COPs consistently. This motivated researchers to propose new algorithms which used an ensemble of multi-operator DE and/or with self-adaptive mechanisms for managing the control parameters, but many of them were mainly proposed to solve unconstrained problems, and adapting them to solve COPs needs careful designs. On the other hand, the performance of existing multi-operator DE algorithms, which solved COPs, could be improved by incorporating extra components, such as diversity.

Therefore, in this paper, an enhanced multi-operator DE has been proposed. In its process, the initial population was divided into a number of sub-populations, each of which was allocated to a different operator. Then, the sub-population sizes were either increased or decreased or kept unchanged, adaptively, depending on the quality of solutions, feasibility rate and diversity, bearing in mind using a simple information sharing scheme. In addition, an improved self-adaptive mechanism for controlling F and Cr has been presented. The algorithm also used a linear reduction mechanism to reduce the population size.

The proposed algorithm has tested on the CEC2010 problems and showed superior performance to our earlier proposed algorithm in terms of the quality of its solutions. In addition, the algorithm had the ability to converge quickly to the best-known solutions. However, its performance may need further improvements for some multi-modal problems. Nevertheless, its performance was much better than state-of-the-art algorithms.

References

[1] 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.

[2] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[3] R. Sarker, S. 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, Oct 2014.

[4] 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, no. 2, pp. 398–417, 2009.

[5] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 71–78.

[6] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization," *Technical Report, Nanyang Technological University, Singapore*, 2010.

[7] R. Storn and K. Price, *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995, vol. 3.

Table II. COMPARISON SUMMARY OF E-MODE AGAINST DE-DPS,εDEAG, ECHT-DE, ECHT-ARMOR-DE (VALUES IN ROWS 1 AND 2 REFER TO NUMBERS OF TEST PROBLEMS FOR WHICH E-MODE WAS BETTER, SIMILAR AND WORSE THAN THE ALGORITHM IN THE 1^{st} COLUMN, BASED ON BEST AND AVERAGE FITNESS VALUES OBTAINED, RESPECTIVELY AND p A PROBABILITY USED TO MAKE DECISIONS BASED ON WILCOXON TEST)

Algorithms	**Criteria**	**10D**					**30D**				
		Better	**Similar**	**Worse**	p	**Decision**	**Better**	**Similar**	**Worse**	p	**Decision**
E-MODE vs. SAMO-DE	**Best fitness values**	2	15	1	1.0	$\approx$	15	1	2	0.005	$+$
	Average fitness values	13	2	3	0.121	$\approx$	14	0	4	0.002	$+$
E-MODE vs. DE-DPS	**Best fitness values**	1	16	1	0.655	$\approx$	11	6	1	0.034	$+$
	Average fitness values	7	9	2	0.26	$\approx$	14	2	2	0.039	$+$
E-MODE vs. εDEag	**Best fitness values**	5	12	1	0.345	$\approx$	17	0	1	0.0002	$+$
	Average fitness values	9	6	3	0.347	$\approx$	16	0	2	0.002	$+$
E-MODE vs. ECHT-ARMOR-DE	**Best fitness values**	2	16	0	0.180	$\approx$	13	4	1	0.001	$+$
	Average fitness values	8	7	3	0.05	$+$	16	1	1	0.0004	$+$

Table III. RANKS OF 5 ALGORITHMS BASED ON FRIEDMAN TEST

	E-MODE	**SAMO-DE**	**DE-DPS**	**εDEag**	**ECHT-ARMOR-DE**
10D	**2.28**	3.53	2.47	3.47	3.25
30D	**1.44**	3.03	2.47	3.94	4.11

[8] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[9] 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.

[10] D. Zaharie, "A comparative analysis of crossover variants in differential evolution," *Proceedings of IMCSIT*, pp. 171–181, 2007.

[11] M. Tasgetiren and P. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problem," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 33–40.

[12] 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, no. 2, pp. 1679–1696, 2011.

[13] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," in *Swarm, Evolutionary, and Memetic Computing*. Springer, 2010, pp. 71–78.

[14] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 561–579, 2010.

[15] R. Mallipeddi and P. N. Suganthan, "Differential evolution with ensemble of constraint handling techniques for solving cec 2010 benchmark problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[16] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Self-adaptive differential evolution incorporating a heuristic mixing of operators," *Computational Optimization and Applications*, vol. 54, no. 3, pp. 771–790, 2013.

[17] A. Zamuda and J. Brest, "Population reduction differential evolution with multiple mutation strategies in real world industry challenges," in *Swarm and Evolutionary Computation*, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds. Springer, 2012, pp. 154–161.

[18] 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.

[19] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[20] J. Brest, B. Boskovic, A. Zamuda, I. Fister, and E. Mezura-Montes, "Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 377–383.

[21] T. J. Choi and C. W. Ahn, "An adaptive cauchy differential evolution algorithm with population size reduction and modified multiple mutation strategies," in *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2*. Springer, 2015, pp. 13–26.

[22] J. Rönkkönen *et al.*, *Continuous Multimodal Global Optimization with Differential Evolution-Based Methods*. Lappeenranta University of Technology, 2009.

[23] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," in *IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2002, pp. 831–836.

[24] J. Tvrdík and R. Polakova, "Competitive differential evolution for constrained problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[25] J. Tvrdík and R. Polakova, "Competitive differential evolution applied to cec 2013 problems," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1651–1657.

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

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

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

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

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

[31] E. Mezura Montes and C. A. Coello Coello, "Adding a diversity mechanism to a simple evolution strategy to solve constrained optimization problems," in *IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2003, pp. 6–13.

[32] C. Si, J. An, T. Lan, T. Ußmüller, L. Wang, and Q. Wu, "On the equality constraints tolerance of constrained optimization problems," *Theoretical Computer Science*, vol. 551, pp. 55–65, 2014.

[33] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–9.

[34] W. Gong, Z. Cai, and D. Liang, "Adaptive ranking mutation operator based differential evolution for constrained optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 716–727, April 2015.

APPENDIX

Due to the number of pages limitation, detailed results can be found in supplementary material (HERE).