2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

United Multi-Operator Evolutionary Algorithms

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

***Abstract*—Multi-method and multi-operator evolutionary algorithms (EAs) have shown superiority to any single EAs with a single operator. To further improve the performance of such algorithms, in this research study, a united multi-operator EAs framework is proposed, in which two EAs, each with multiple search operators, are used. During the evolution process, the algorithm emphasizes on the best performing multi-operator EA, as well as the search operator. The proposed algorithm is tested on a well-known set of constrained problems with 10D and 30D. The results show that the proposed algorithm scales well and is superior to the-state-of-the-art algorithms, especially for the 30D test problems.**

***Index Terms*— Constrained optimization, evolutionary algorithms, multi-method algorithms, multi-operator algorithms**

I. Introduction

Constrained optimization is a challenging research area in the science and engineering disciplines. Locating the optimal solution for such problems is often difficult, as the characteristics and mathematical properties do not follow any standard patterns or forms. The constrained optimization problems (COPs) may contain different types of variables, such as real, integer and discrete, and may have equality and/or inequality constraints. The objective and constraint functions could be either linear or nonlinear. The functions may be either continuous or discontinuous, and either unimodal or multimodal. The feasible region of such problems could be either a tiny or a significant portion of the search space. Moreover, the feasible region could be either one single bounded region or a collection of multiple disjoint regions. In some practical problems, the feasible region could even be unbounded. Finally, the optimal solution may exist either on the boundary, or in the interior, of the feasible region.

Over the last decade or so, EAs have recognized as powerful algorithms for solving both constrained and unconstrained optimization problems. The EAs family contains many different algorithms, such as the genetic algorithm (GA) [1], differential evolution (DE) [2] and evolution strategies (ES) [3, 4] .These algorithms do not require the satisfaction of any standard mathematical properties, such as continuity and differentiability, and have the ability to effectively deal with large search spaces [5]. Although there have been many EAs introduced in the literature, no single algorithm performs consistently over a wide range of problems [6].

The authors are with the School of Engineering and Information Technology, University of New South Wales at Canberra, Australia, emails: {s.elsayed, r.sarker and d.essam}@adfa.edu.au

The concept of multi-method algorithms has emerged recently that utilizes the strength of different EAs, within a single algorithm structure, to deal with different types of problems. Vrugt *et al*. [7] introduced an algorithm, known as AMALGAM (A Multi-ALgorithm Genetically Adaptive Multiobjective), that has been proven to be a powerful approach for solving multiobjective problems. Later, Vrugt *et al*. [8] extended their work for real valued function optimization. As tested on a set of benchmark problems, the algorithm obtained similar efficiencies as existing algorithms on relatively simple problems, but it was increasingly superior for more complex and higher dimensional multimodal optimization problems.

A brief review on the multi-operator based EAs is provided here. Yong *et al.* has recently proposed a composite DE algorithm (CoDE) [9], in which the algorithm randomly combines several trial vector generation strategies with a number of control parameter settings at each generation to create new trial vectors. CoDE has been tested on a set of unconstrained problems and showed competitive performance in comparison to other state-of-the-art algorithms. Elsayed *et al.* [10] proposed a mix of four different DE mutation strategies within a single algorithm framework to solve COPs which performed well for a set of constrained problems that was further extended and improved in [11, 12]. Elsayed *et al*. [13] also proposed two novel DE variants, each of which utilized the strengths of multiple mutation and crossover operators, to solve 60 constrained problems. The algorithm demonstrated superior performances in comparison with the state-of-the-art algorithms. Caraffini *et al.* [14] proposed a super fit multi-adaptive DE for solving unconstrained problems. They employed four DE operators with equal probability. In the algorithm, based on normalized relative fitness improvement and normalized distance to the best individual measures, the probabilities were updated. To add to this, the control parameters were adopted, in which F was generated using a Cauchy distribution, while Cr was generated based on a normal distribution. Both parameters were then adapted during the evolution process. To enhance the performance of the algorithm, the covariance matrix adaptive evolution strategy was also used as a local search. Brest *et al.* [15] proposed a DE algorithm which embedded a self-adaptation mechanism for parameter control. Here, the population was divided into sub-populations to apply more DE strategies, and a population diversity mechanism was also introduced. The algorithm was tested on a set of unconstrained problems.

In this research, the combination of multi-operator algorithms is explored. The algorithm is named as united

multi-operator EAs (UMOEAs). In this algorithm, the initial population is divided into two subpopulations; and each subpopulation is independently evolved using a multi-operator algorithm. The success rate of each multi-operator algorithm is recorded for a certain number of generations and the better performing multi-operator is used to evolve its own individuals for a number of subsequent generations (known as a cycle), while the other population is kept on hold. After this cycle, information from the best performing population is used to update the individuals of the worst performing population, and subsequently both multi-operator algorithms rerun independently in parallel. The process is continued up to a predefined number of fitness function evaluations and then the best performing multi-operator algorithm is selected to evolve only its assigned population during the rest of the evolution process.

The performance of the proposed algorithm is tested on a well-known set of constrained problems [16] which contains 18 test problems, with different mathematical properties, with 10 and 30 dimensions. From the results, the proposed algorithm shows consistently excellent performance in comparison with the state-of-the-art algorithm, as it is able to obtain better results, especially for the 30D test problems, and statistically significant.

This paper is organized as follows: after the introduction, section II presents a brief overview on GA and DE. In section III, the design of the proposed algorithm is discussed. The experimental results and analysis are demonstrated in section IV. Finally, the conclusions and future work are given in section V.

II. Basic Algorithms and Operators

The proposed framework can consider any number of EAs. In this section, we describe the algorithms and the operators we considered in this research.

A. Differential Evolution

In DE, two search operators are usually used as discussed below.

A.1. Mutation

In the standard mutation strategy, DE/rand/1, a mutant vector ($\vec{v}_{z,t}$) is generated by multiplying the amplification factor F by the difference of two random vectors, and the result is added to another third random vector (equation 1).

$$\vec{v}_{z,t} = \vec{x}_{r_1,t} + F.\left(\vec{x}_{r_2,t} - \vec{x}_{r_3,t}\right) \quad (1)$$

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

The purpose of the mutation operation is to explore the search space and maintain diversity. In the literature, there are many strategies for mutation, such as: DE/rand-to-best/2 [17], rand/2/dir [18], DE/current-to-best/1[19], and DE/Current-to-pbest [20].

A.2. Crossover

DE algorithms usually use two type of crossover operators. In this paper, we use the binomial crossover, because it is widely accepted and is superior to the exponential one [21].

The crossover operator is performed on each of the j^{th}variables whenever a randomly picked number ∈ [0,1] is less than or equal to a crossover rate (*Cr*). In this case, the number of parameters inherited from the mutant vector has a (nearly) binomial distribution

$$u_{zj,t} = \begin{cases} v_{zj,t}, & if\ (rand \leq Cr\ or\ j = j_{rand}) \\ x_{zj,t}, & otherwise \end{cases} \quad (2)$$

where $rand \in [0,1]$, and $j_{rand} \in [1, D]$ is a randomly chosen index, which ensures trial vector ($\vec{u}_{z,t}$) gets at least one component from $\vec{v}_{z,t}$.

B. Genetic Algorithms

In this paper, we use GA with simulated binary crossover (SBX) [22] and a non-uniform mutation (NU-M) [23] as well as MPC-GA [24, 25]. The reason for choosing these operators is that GA-MPC has shown its superiority to many other algorithms [24], and GA with SBX plus NU-M outperformed nine different GA variants, as reported in [26].

B.1. MPC-GA

In this algorithm, an initial population is generated randomly, with a size of *PS*. An archive pool (A_{arch}) is filled with the best *m* individuals (based on their constraint violations and/or fitness function). A tournament selection with size tc is applied, from which the best individual is selected and saved in the selection pool. Using the crossover operation, for a given crossover rate, three individuals from the selection pool is used to generate three offspring as follows:

$$\vec{y}_1 = \vec{x}_1 + \beta \times (\vec{x}_2 - \vec{x}_3) \quad (3)$$

$$\vec{y}_2 = \vec{x}_2 + \beta \times (\vec{x}_3 - \vec{x}_1) \quad (4)$$

$$\vec{y}_3 = \vec{x}_3 + \beta \times (\vec{x}_1 - \vec{x}_2) \quad (5)$$

On each new individual $\vec{y}_i$, a diversity operator is applied that helps to escape from any local minima and to move to better regions in the search space. In this case, for each individual a uniform random number ∈ [0, 1] is generated, if it is less than a predefined probability, p, then $y_i^j = x_{arch}^j$. Subsequently, the individuals from the archive pool are merged with all of the offspring, and the best *PS* individuals are selected as the population for the next generation.

B.2. SBX

We use SBX crossover as it is widely used in practice. The probability distribution of β in this crossover is similar to the probability distribution of β in binary-coded crossover.

Using a pair of parents $\vec{x}^1 = (x_1^1, x_2^1, \dots, x_n^1)$ and $\vec{x}^2 = (x_1^2, x_2^2, \dots, x_n^2)$, two offspring $\vec{y}^1 = (y_1^1, y_2^1, \dots, y_n^1)$ and $\vec{y}^2 = (y_1^2, y_2^2, \dots, y_n^2)$ are generated as follows:

1. Generate a uniform random number $rand \in [0,1]$.
2. Generate a random number $\bar{\beta}$ as follows:

$$\bar{\beta} = \begin{cases} (2.rand)^{\frac{1}{1+\eta}}, & rand \leq 0.5 \\ \left(\frac{1}{2(1-rand)}\right)^{\frac{1}{1+\eta}}, & otherwise \end{cases} \quad (6)$$

3. Generate two offspring as follows:

$$y_j^1 = \frac{1}{2}\left[(1+\bar{\beta}).x_j^1 + (1-\bar{\beta}).x_j^2\right] \quad (7)$$

$$y_j^2 = \frac{1}{2}\left[(1-\bar{\beta}).x_j^1 + (1+\bar{\beta}).x_j^2\right] \quad (8)$$

When compared to other real-coded crossover implementations, SBX works well in many test problems that have a continuous search space. The SBX operator can restrict child solutions to any arbitrary closeness to the parent solutions, thereby not requiring any separate mating restriction scheme for better performance. SBX is very useful for the problems in which the bounds of the optimum are not known and where multiple optima may exist [22].

To maintain the genetic diversity from one generation to another, the use mutation operator is well-established. The performance of non-uniform mutation is well-known in this regard. In the process, the step size is decreased as the generation is increased, thus it helps to making bigger search steps in the initial stage and smaller steps at the later stages [23]. The offspring $x_z'(t) = (x_{z,1}'(t), x_{z,2}'(t), \dots, x_{z,D}'(t))$ is mutated according to:

$$x_{z,i}'(t) = x_{z,i}(t) + \delta_{z,i}(t) \quad (9)$$

using the random variation:

$$\delta_{z,j}(t) = \begin{cases} \left(\bar{x}_j - x_{z,i}(t)\right).\left(1 - [rand(t)]^{(1-\frac{t}{T})^b}\right), & if\ rand \leq 0.5 \\ \left(\underline{x}_j - x_{z,i}(t)\right).\left(1 - [rand(t)]^{(1-\frac{t}{T})^b}\right), & if\ rand > 0.5 \end{cases} \quad (10)$$

where $\bar{x}_j$ and $\underline{x}_j$ are the upper and lower bound of individual $x_{z,j}$, respectively, $rand(t)$ is a random number $\in [0,1]$, t is the generation number, T is the maximum number of generations, and b is a parameter to control the speed at which the step length decreases. This operator performs very well for problems when a solution only needs to be refined during the later stages of the execution of an algorithm

III. United Multi-operator Evolutionary Algorithms (UMOEAs)

In this section, we discuss the proposed algorithm, the improvement measure and the constraint handling technique used in this paper.

A. UMOEAs

The pseudo-code of the proposed algorithm is provided in this Algorithm 1. The algorithm starts with an initial population of size PS, that is randomly generated using a uniform distribution, in which each decision variable must be within its bounds. The individuals of the population is then divided into two subpopulations (PS_1 and PS_2). For a predefined number of generations, evolve the subpopulation PS_1 using a multi-operator DE algorithm, while PS_2 is evolved using a multi-operator GA algorithm.

In the multi-operator DE algorithm, for each individual in PS_1 a random number ($rand \in [0,1]$) is generated, if it is less than a predefined probability ($prob_1$), a new individual is generated using (11), otherwise it will be generated using (12).

ALGORITHM I. United multi-operator evolutionary algorithms pseudo-code

- Generate initial population; each variable is generated within its boundaries.
- Divide the population into two groups (PS_1, PS_2) with equal size
  - Initialize each algorithm's parameters and set $period = s_1 = s_2 = s_3 = s_4 = 0$;
  $prob_1 = prob_2 = 0.5$;

while $FFEs < maxFFEs$
- **if** $period < CS$ && $FFEs < MixStage$
  - $period = period + 1$;
  - evolve PS_1 using multi-operator DE, such that
    - **if** $rand < prob_1$
      - generate a new solution vector using DE1
      - if it is better than its parent, set $s\_1 = s\_1 + 1$;
    - **else**
      - generate a new solution vector using DE2
      - if it is better than its parent, set $s\_2 = s\_2 + 1$;
    - **end**
    - update $prob_1 = max(0.05, \frac{s_1}{s_1+s_2})$)
  - evolve PS_2 using multi-operator GA:
    - **if** $rand < prob2$
      - generate new solutions vector using MPC-GA;
    - **else**
      - generate new solutions vector using SBX-NUM
    - **end**
    - calculate the success of each GA and updated s_3 and s_4
    - update $prob_2 = max(0.05, \frac{s_3}{s_3+s_4})$)
    - calculate the improvement of each multi-operator at generation t, $imp1(t)$, $imp2(t)$
- **end**
- **if** $mod(period, CS) = 0$
  - **if** $sum(imp1) < sum(imp2)$
    - $best\_EA = 1$; **else**
    - $best\_EA = 2$;
  - **end**
- **end**
- **if** $period > CS$ && $period < 2CS$
  - **if** $best\_EA == 1$
    - evolve PS_1 using multi-operator DE; **else**
    - evolve PS_2 using multi-operator GA;
  - **end**
- **end**
- **if** $period == 2CS$
  - calculate the mean($\overline{x}$) and standard deviation (σ) vectors of the μ best individuals of the $best\_EA$
  - generate new population for the worst performing multi-operator algorithm, such as:
  - $\vec{x}_z = N(\overline{x}, \sigma)$, where $z > 1$
  - $period = s_1 = s_2 = s_3 = s_4 = 0$; $prob_1 = prob_2 = 0.5$;
- **end**

update FFEs; $t = t + 1$;

end

$$\vec{v}_{i,t} = \vec{x}_{\varphi,t} + F.\left(\vec{x}_{r_1,t} - \vec{x}_{r_2,t}\right) \quad (11)$$

where φ is a random integer number between $\frac{PS_1}{10}$ and $\frac{PS_1}{2}$.

$$\vec{v}_{i,t} = \vec{x}_{i,t} + F.\left(\left(\vec{x}_{r_1,t} - \vec{x}_{r_2,t}\right) + \left(\vec{x}_{\varphi,t} - \vec{x}_{i,t}\right)\right) \quad (12)$$

where φ is a random integer number between [1, $\frac{PS_1}{4}$]. It is worthy to mention here that φ is selected after PS_1 is sorted, based on the fitness function and/or constraint violation. Note also that the binomial crossover is considered in this study, as in the literature [21], it showed superiority to the exponential one.

If the new offspring is better than its parent (based on the fitness function and/or constraints violation), the success of the corresponding mutation (s_1 or s_2, respectively) is increased by one. After each generation, $prob_1$ is updated, such that $prob_1 = \frac{s_1}{s_1+s_2}$.

Following the same methodology, in the multi-operator GA, to generate new individuals, a random number ($rand \in [0,1]$) is generated, then if it is less than a predefined probability ($prob_2$), three individuals are generated using MPC-GA, otherwise two individuals are produced using SBX-NU. MPC-GA uses an archive of individuals, as shown in section II.B, once new PS_2 individuals are generated, those individuals in the archive and the new PS_2 are merged, and the best PS_2 are passed on to the next generations. Next that, the number of individuals generated by MPC-GA and that passed on to the next generation is assigned to s_3, while those generated by SBX-NU and passed on to the next generation are assigned to s_4. Consequently, $prob_2$ is $\frac{s_3}{s_3+s_4}$.

Subsequently, the improvements of each multi-operator algorithm at generation (t) ($imp_1(t)$ and $imp_2(t)$) are calculated as shown in the following section.

The abovementioned process is repeated for CS generation (named as a cycle). The summation of both improvements are calculated, such that $\sum_{c=1}^{CS} imp_{o,c}$, where o is 1 or 2. Then, the best performing multi-operator is selected to evolve only its population for CS generations, while the other population is kept on hold. Once this step is over, all parameters are re-set to their initial values and the population which is on hold is injected using information from that population which was successful, such that: for the best μ individuals in the successful population, the mean and standard deviation vectors ($\overline{x}$ and σ, respectively) are calculated, as:

$$\overline{x}_j = \frac{\sum_{i=1}^{\mu} x_{i,j}}{\mu} \quad (13)$$

$$\sigma_j = \frac{\sqrt{\sum_{i=1}^{\mu}(x_{i,j} - \overline{x}_j)^2}}{\mu} \quad (14)$$

and hence, the population which was on hold is updated by generating a Gaussian number random with $\overline{x}_j$ and σ_j:

$$x_{z,j} = N(\overline{x}_j \sigma_j), \text{ where } z > 1 \quad (15)$$

The process of using two multi-operator algorithms is used up to a predefined number of fitness function evaluations ($MixStage$), i.e.$MixStage = \frac{maxFFEs}{3}$, not for all the evolution process. So that if this condition is met, the best performing multi-operator algorithm is selected to evolve only its population individuals, while the other population is kept on hold until the end of the evolution process

B. Improvement Measure

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

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

$$VI_{i,t} = \frac{\left|Vio_{i,t}^{best} - Vio_{i,t-1}^{best}\right|}{avg.Vio_{i,t}} = I_{i,t} \quad (16)$$

where $Vio_{i,t}^{best}$ is the constraints violation of the best individual at generation t and $avg.V_{ioi,t}$ the average violation. Hence $VI_{i,t} = I_{i,t}$ above represents the relative improvement in comparison to the average violation in the current generation.

2. Feasible to feasible: For any multi-operator algorithm i, if the best solution was feasible at generation $t-1$ and was still feasible in generation t, then the improvement index is:

$$I_{i,t} = \max_i(VI_{i,t}) + \left| F_{i,t}^{best} - F_{i,t-1}^{best}\right| \times FR_{i,t} \quad (17)$$

where $I_{i,t}$ is the improvement for i^{th} multi-operator variant at generation t, $F_{i,t}^{best}$ the objective function for the best individual at generation t, and the feasibility ratio of a variant i at generation t is:

$$FR_{i,t} = \frac{Number\ of\ feasible\ solutions\ in\ a\ subpopulation\ i}{Subpopulation\ size\ at\ iteration\ t} \quad (18)$$

To assign a higher index value to a multi-operator algorithm with a higher feasibility ratio, the improvement of fitness value is multiplied by the feasibility ratio. To differentiate between the improvement index of feasible and infeasible groups of individuals, a term $\max_i(VI_{i,t})$ is added to (17). If all the best solutions are feasible, then $\max_i(VI_{i,t})$ will be zero.

3. Infeasible to feasible: For any multi-operator algorithm i, if the best solution was infeasible at generation $t-1$ and it is feasible in generation t, then the improvement index is:

$$I_{i,t} = \max_i(VI_{i,t}) + \left| vio_{i,t-1}^{best} + F_{i,t}^{best} - F_{i,t-1}^{bv}\right| \times FR_{i,t} \quad (19)$$

where $F_{i,t-1}^{bv}$ is the 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, $vio_{i,t-1}^{best}$ is added with the change of fitness value in (19).

C. Discussion

Here, some issues, regarding the design of the proposed algorithm, are discussed.

1- The reason for generating new individuals for the worst performing multi-operator algorithm, instead of directly copying them from the best performing one, is to maintain diversity. However, it may not be efficient to generate a totally random population, as this may cost fitness evaluations without any valuable outcome. Therefore, information from the best μ individuals in the successful population is considered, as shown in (13) - (15).

2- The reason for using different values for φ in (11) and (12) is that to maintain diversity as well as to enhance the intensification. Note that equation (12) is similar to the DE/Current-to-best/1 variant if $\varphi = 1$.

3- The reason for using two multi-operator algorithms only up to $MixStage$ fitness evaluations (here equal to $\frac{maxFFEs}{3}$), and not for all the evolution process, is to reduce the time complexity of the algorithm to reach the optimal solution, especially at this stage the decision of which multi-operator algorithm performs best can usually be justifiably made.

4- The point behind reusing two multi-operator algorithms, instead of one, after every $2CS$ generations, is that passing good information for a poor multi-operator algorithm may help it to reach better solutions latter on.

5- It is important to mention here that a minimum threshold to use an operator in each multi-operator algorithm is set, i.e. 5%, to keep the benefit from poorly performing operators as they may perform better at later generations.

D. Constraint Handling

In this paper, we consider the selection of the individuals for the purposes of a tournament [27] as follows: *i*) between two feasible solutions, the fittest one (according to fitness function) is better, *ii*) a feasible solution is always better than an infeasible one, *iii*) between two infeasible solutions, the one having the smaller sum of constraint violations is preferred.

In this approach, no penalty factor is required, since the selection procedure only performs pair-wise comparisons. Therefore, feasible solutions have fitness equal to their objective function value, and the use of constraint violation in the comparisons aims to push infeasible solutions towards the feasible region.

The equality constraints are also transformed to inequalities of the following form, where ε is a small value, here it is equal to 0.0001:

TABLE I. DETAILS OF ALL PARAMETERS VALUES

General: $PS = 200 \rightarrow PS_1 = PS_2 = 100$, $CS = 25$, $\mu = 25$ and $MixStage = \frac{maxFFEs}{3}$.
DE: $Cr = \{0.4, 0.85, 0.99\}$ and $F = [0.4, 0.95]$, $\varphi \in [\frac{PS_1}{10}, \frac{PS_1}{2}]$ in (11) [28], while it is $\varphi \in [1, \frac{PS_1}{2}]$ in (12).
GA: $p = 0.2$, $arch = \frac{PS_1}{2}$, *Cr*=100%, *mutation rate* =0.1, *tournament selection size* = 2 or 3 randomly, $\eta = 3$ and $b = 5$ [26].

$$|h_e(\vec{x})| - \varepsilon \leq 0, for\ e = 1, \dots, E \qquad (20)$$

where E is the number of equality constraints.

IV. EXPERIMENTAL RESULTS

In this section, the performance of the proposed algorithm is discussed and analyzed by solving a well-known set of constrained problems which contain 18 test problems with both 10 and 30 dimensions [16]. The algorithm was run 25 times for each test problem, where the stopping criterion was to run for up to 200K FEs for the 10D instances, and 600K FEs for the 30D problems.

To begin with, all parameter values are provided in Table I. It is worthy to mention here that multiple Cr values used, instead of one, as it is confirmed in the literature that small values of Cr are good for separable functions, while large values are suitable for non-separable functions. Hence for each new individual a random value from the Cr list was selected, while F was selected within a range, which is well-known in the literature.

A. Comparison with the state-of-the-art-algorithms

Here the computational results of UMOEAs are compared with the state-of-the-art algorithms, εDEag [29], which won the CEC2010 constrained optimization competition, and improved jDE (*jDEsoco*). The detailed results are shown in Appendix A.

It is important to highlight that UMOEAs was able to reach 100% feasibility ratios for both the 10D and 30D instances, but εDEag obtained 100% and 95.11% feasibility ratios for the 10D and 30D test problems, respectively. The average feasibility ratio for *jDEsoco* was 98%.

Considering the quality of the solutions obtained, a summary is reported in Table I. From this table, UMEAs was found superior performance to the other algorithms for the majority of the test problems, especially for the 30D instances.

Furthermore, the Wilcoxon signed rank test [30] was used which allowed us to statistically judge the difference between paired scores when it was not possible to make the assumptions required by the t test, such as that the population should be normally distributed. The results based on the best and average fitness values are presented in the last column in Table II. As a null hypothesis, it is assumed that there is no significant difference between the best and/or

TABLE II. COMPARISON AMONG UMOEAS, EDEAG AND JDESOCO

D	Comparison	Results	Better	Equal	Worse	Dec.
10D	UMOEAs – to – εDEag	Best	5	12	1	≈
		Average	10	6	2	+
	UMOEAs – to – *jDEsoco*	Best	6	12	0	+
		Average	15	1	2	+
30D	UMOEAs – to – εDEag	Best	16	1	1	+
		Average	15	1	2	+
	UMOEAs – to – *jDEsoco*	Best	16	0	2	+
		Average	17	0	1	+

mean values of two samples whereas the alternative hypothesis is that there is a significant difference at a 5% significance level. Based on the results, one of three signs (+, -, and ≈) was assigned for the comparison of any two algorithms, where "+" means the first algorithm was significantly better than the second, "-" means that the first algorithm was significantly worse, and "≈" means that there was no significant difference between the two algorithms. From Table II, it is clear that UMOEAs was statistically better than εDEag, in regard to both the best and average results for the 30D test problems, while the performance of UMOEAs was statistically better in regard to the average results for the 10D test problems. The performance of UMOEAs was also better than *jDEsoco* in regards to both the best and average results

B. Scaling Analysis

In this analysis, the relationship between the dimensionality of the test problem and the average number of function evaluations needed to find the best solutions with the tolerance limit (here equal to 0.0001) was derived. Due to the number of pages limitation, only one test problem, C07, was chosen for the purpose. The problem is known to be difficult as the objective function is non-separable, multi-modal and is shifted by a matrix shi, while the constraint is separable, multi-modal and also shifted by the same matrix. The optimal solution of this problem is at $f(x^*) = 0$. The problem was solved using different dimensions, i.e. D = 5, 10, 15, 20, 25 and 30 variables. For each D, the algorithm was run over 50 trials, and the average FFEs, to reach the stopping criteria, were recorded. It should be noted that the evaluation of a constraint was counted as one. It is also worthy to mention here that up to only 30 variables were used, as the available data are up to 30 dimensions.

Figure 1 shows the average FFEs for each dimension. For a further investigation, a regression line [31] is fitted to help readers to approximate the average fitness evaluations that are required for different dimensions.

So, the regression equation for C07 was:

$$FEs = 139.2D^2 + 7820D - 2017 \qquad (18)$$

and the coefficient of determination [31] was 99.68%. This means that the line is highly fitted to predict future values.

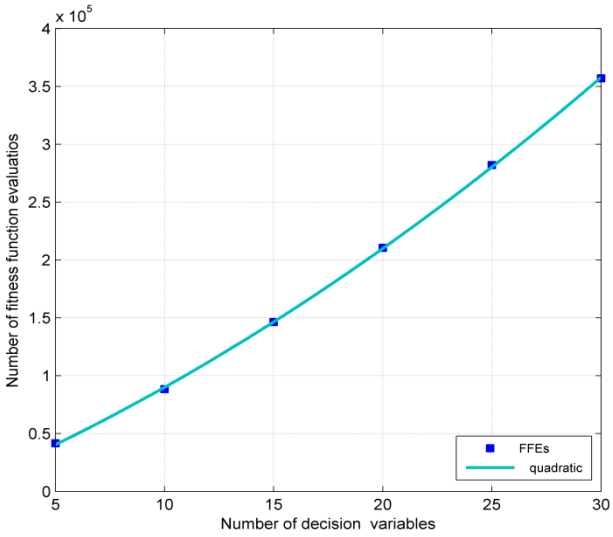

Fig.1. Average number of FFEs versus the problem dimension for C07 using 50 runs. The line represents the quadratic regression fitting of the data.

C. A Brief Discussion

Here the effects of CS and $MixStage$ are analyzed. However, due to the maximum number of pages limitation, the detailed results are not presented here.

Firstly, each 30D constrained problem was solved with three different values for CS, CS= 25, 50 and 75 generations. The ranking mechanism presented in [10] was used to rank all variants. Based on that, the variant with $CS = 50$ is the best.

In regards to $MixStage$, each 30D problem was solved with two different values, such that $MixStage = \frac{maxFFEs}{3}$ and $\frac{maxFFEs}{2}$. Based on the ranking mechanism, $MixStage = \frac{maxFFEs}{3}$ was found the best.

V. CONCLUSIONS AND FUTURE WORK

In the last decade, many EAs have been introduced to solve constrained optimization problems. Most of those algorithms were designed to use a single crossover and/or a single mutation operator. In this paper, we adopted the concept of multiple algorithms empowered by multiple operators, in which the initial population was divided into two subpopulations, and each subpopulation was independently evolved using its assigned multi-operator algorithm. After a predefined number of fitness evaluations, the best performing multi-operator continued to evolve its own subpopulation, while the other group of individuals was on hold. Then, after a few generations, information from the best performing subpopulation was used to update the worst performing subpopulation, and hence the two multi-operator algorithms were rerun in parallel again. The procedure was continued until a defined stage and then the best performing

algorithm was selected to evolve its population and the worst one was totally disregarded.

The algorithm was tested on the CEC2010 constrained benchmark problems, and showed better performance in comparison with the-state-of-the-art algorithms.

For future work, we wish to deeply analyze the algorithm's components and to also test it on real-world and dynamic problems.

REFERENCES

[1] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. MA: Addison-Wesley, 1989.

[2] R. Storn and K. Price, "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute Technical Report, Tech. Rep. TR-95-012, 1995.

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

[4] T. Bäck, *Evolutionary Algorithms in Theory and Practice : Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press, 1996.

[5] R. Sarker, J. Kamruzzaman, and C. Newton, "Evolutionary optimization (EvOpt): a brief review and analysis," *International Journal of Computational Intelligence and Applications,* vol. 3, pp. 311-330, 2003.

[6] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation,* vol. 1, pp. 67-82, 1997.

[7] J. A. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," in proceeding *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 2007, pp. 708–711.

[8] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 243-259, 2009.

[9] Y. Wang, Z. Cai, and Q. Zhang, "Differential Evolution With Composite Trial Vector Generation Strategies and Control Parameters," *IEEE Transactions on Evolutionary Computation,* vol. 15, pp. 55-66, 2011.

[10] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization Problems," *Computers and Operations Research,* vol. 38, pp. 1877-1896, 2011.

[11] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "On an evolutionary approach for constrained optimization problem solving," *Applied Soft Computing,* vol. 12, pp. 3208-3227, 2012.

[12] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An Improved Self-Adaptive Differential Evolution Algorithm for Optimization Problems," *Industrial Informatics, IEEE Transactions on,* vol. 9, pp. 89-99, 2013.

[13] S. Elsayed, R. Sarker, and D. Essam, "Self-adaptive differential evolution incorporating a heuristic mixing of operators," *Computational Optimization and Applications,* pp. 1-20, 2012.

[14] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinali, G. Iacca, and E. Mininno, "Super-fit Multicriteria Adaptive Differential Evolution," in proceeding *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 1678-1685.

[15] 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 proceeding *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 377-383.

[16] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2010 competition and special session on single objective constrained real-parameter optimization," Technical Report, Nangyang Technological University, Singapore, Tech. Rep. 2010.

[17] 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, pp. 398-417, 2009.

[18] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in proceeding *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004, p. 165.

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

[20] Z. Jingqiao and A. C. Sanderson, "JADE: Adaptive Differential Evolution With Optional External Archive," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 945-958, 2009.

[21] E. Mezura-Montes, J. V. Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in proceeding *the 8th annual conference on Genetic and evolutionary computation*, Seattle, Washington, USA, 2006, pp. 485-492.

[22] R. B. Agrawal, K. Deb, K. Deb, and R. B. Agrawal, "Simulated Binary Crossover for Continuous Search Space," *Complex Systems,* vol. 9, pp. 115–148, 1995.

[23] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1992.

[24] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems," in proceeding *IEEE Congress on Evolutionary Computation*, 2011, pp. 1034-1040.

[25] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A new genetic algorithm for solving optimization problems," *Engineering Applications of Artificial Intelligence,* vol. 27, pp. 57-69, 2014.

[26] S. Elsayed, R. Sarker, and D. Essam, "A Comparative Study of Different Variants of Genetic Algorithms for Constrained Optimization, Simulated Evolution and Learning." vol. 6457, K. Deb, *et al.*, Eds., ed: Springer Berlin / Heidelberg, 2010, pp. 177-186.

[27] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering,* vol. 186, pp. 311-338, 2000.

[28] R. Sarker, S. Elsayed, and T. Ray, "Differential Evolution with Dynamic Parameters Selection for Optimization Problems," *Evolutionary Computation, IEEE Transactions on,* vol. PP, pp. 1-1, 2013.

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

[30] G. W. Corder and D. I. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Hoboken, NJ: John Wiley, 2009.

[31] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis, 4th Edition*. New York: John Wiley and Sons, 2006.

APPENDIX A

FUNCTION VALUES ACHIEVED BY UMOEAs, εDEAG AND *JDEsoco* FOR THE CEC2010 TEST PROBLEMS

Prob.	Alg.	10D			30D		
		Best	**Mean**	***St. d***	**Best**	**Mean**	***St. d***
C01	UMOEAs	**-7.473104E-01**	**-7.473104E-01**	**2.766290E-16**	**-8.218844E-01**	-8.177187E-01	4.068346E-03
	εDEag	**-7.473104E-01**	-7.470402E-01	1.323339E-03	-8.218255E-01	**-8.208687E-01**	**7.103893E-04**
	jDEsoco	-0.7473103	-7.3836E-01	1.6006E-02	-0.8218841	-8.1238E-01	1.3187E-02
C02	UMOEAs	**-2.2777104E+00**	**-2.277631E+00**	**2.674215E-04**	**-2.280972E+00**	**-2.276652E+00**	**3.215291E-03**
	εDEag	-2.277702E+00	-2.269502E+00	2.3897790E-02	-2.169248E+00	-2.151424E+00	1.197582E-02
	jDEsoco	-1.1176240	5.6359E-01	1.1044	0.6792295	1.5603	8.4705E-01
C03	UMOEAs	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.0000000E+00**	**6.441640E-24**	**1.398980E-23**
	εDEag	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	2.867347E+01	2.883785E+01	8.047159E-01
	jDEsoco	**0.00000E+00**	7.8105	2.9437	9.993685E-22	6.1447E+01	5.7577E+01
C04	UMOEAs	**-1.000000E-05**	**-1.000000E-05**	**8.879988E-17**	**-3.333331E-06**	**-3.333299E-06**	**2.710191E-11**
	εDEag	-9.992345E-06	-9.918452E-06	1.5467300E-07	4.698111E-03	8.162973E-03	3.067785E-03
	jDEsoco	**-1.00000E-05**	**-1.00000E-05**	9.4831E-16	8.0490978E-05	3.5187E-04	2.3948E-04
C05	UMOEAs	**-4.836106E+02**	**-4.836106E+02**	**0.00000E+00**	**-4.83611E+02**	**-4.83611E+02**	**2.850879E-06**
	εDEag	**-4.836106E+02**	**-4.836106E+02**	3.89035E-13	-4.531307E+02	-4.495460E+02	2.899105E+00
	jDEsoco	**-483.6106247**	-3.0217E+02	3.0256E+02	-19.9503700	1.0822E+02	1.5203E+02
C06	UMOEAs	**-5.786624E+02**	**-5.786624E+02**	**3.268074E-08**	-5.306375E+02	**-5.306351E+02**	**4.396221E-03**
	εDEag	-5.786581E+02	-5.786528E+02	3.6271690E-03	-5.285750E+02	-5.279068E+02	4.748378E-01
	jDEsoco	**-578.662366**	-5.7408E+02	1.6461E+01	**-530.6377271**	-4.7284E+02	1.2743E+02
C07	UMOEAs	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.0000000E+00**	**5.043447E-26**	**5.859419E-26**
	εDEag	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	1.147112E-15	2.603632E-15	1.233430E-15
	jDEsoco	**0.00000E+00**	6.4192E-27	1.3515E-26	4.2197747E-26	8.7396E-24	3.2529E-23
C08	UMOEAs	**0.00000E+00**	4.522781E+00	**4.806185E+00**	**0.0000000E+00**	**2.218930E-26**	**3.449849E-26**
	εDEag	**0.00000E+00**	6.727528E+00	5.560648E+00	2.518693E-14	7.831464E-14	4.855177E-14
	jDEsoco	**0.00000E+00**	**3.7421**	1.0330E+01	7.2310934E-26	8.2585E+01	2.4395E+02
C09	UMOEAs	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	**7.354649E-28**	**5.092577E-25**	**9.259689E-25**
	εDEag	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	2.770665E-16	1.072140E+01	2.821923E+01
	jDEsoco	**0.00000E+00**	5.2898E-01	1.4620	2.7729234E-25	2.4743	8.7782
C10	UMOEAs	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.0000000E+00**	**1.376304E-25**	**1.413405E-25**
	εDEag	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	3.252002E+01	3.326175E+01	4.545577E-01
	jDEsoco	**0.00000E+00**	3.1712E+01	1.8188E+01	1.0862047E-25	2.9386E+01	7.1786
C11	UMOEAs	**-1.52271E-03**	**-1.52271E-03**	**1.559697E-12**	**-3.92344E-04**	**-3.92344E-04**	**1.35647E-10**
	εDEag	**-1.52271E-03**	**-1.52271E-03**	6.3410350E-11	-3.268462E-04	-2.863882E-04	2.707605E-05
	jDEsoco	**-0.0015227**	-8.2555E-03*	2.3807E-02	-0.0003920	1.1667E-03	5.2690E-03
C12	UMOEAs	-1.992458E-01	-1.992458E-01	**5.970216E-11**	**-1.99264E-01**	**-1.99263E-01**	**4.08248E-08**
	εDEag	**-5.700899E+02**	**-3.367349E+02**	1.7821660E+02	-1.991453E-01	3.562330E+02*	2.889253E+02
	jDEsoco	-0.1992457	-2.2365E+01*	1.1083E+02	-0.1992634	-1.9925E-01	2.3453E-05
C13	UMOEAs	**-6.842937E+01**	-6.640174E+01	2.260741E+00	-6.561403E+01	-6.334698E+01	1.229909E+00
	εDEag	**-6.842937E+01**	**-6.842936E+01**	**1.0259600E-06**	-6.642473E+01	-6.535310E+01	5.733005E-01
	jDEsoco	-68.4293648	-6.8315E+01	5.7018E-01	**-68.4293209**	**-6.7537E+01**	**5.0553E-01**
C14	UMOEAs	**0.00000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.0000000E+00**	**1.558464E-25**	**2.326714E-25**
	εDEag	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	5.015863E-14	3.089407E-13	5.608409E-13
	jDEsoco	**0.00000E+00**	9.1221E-01	2.4538	5.7101696E-26	1.5946E-01	7.9732E-01
C15	UMOEAs	**0.00000E+00**	**2.893417E-21**	**1.446705E-20**	**6.323509E-27**	**8.833195E-23**	**2.391952E-22**
	εDEag	**0.000000E+00**	1.798980E-01	8.8131560E-01	2.160345E+01	2.160376E+01	1.104834E-04
	jDEsoco	2.0257948E-26	1.2452E+09	3.8127E+09	9.6993452E-16	1.5357E+09	1.6045E+09
C16	UMOEAs	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**	**0.000000E+00**	**0.000000E+00**	**0.0000000E+00**
	εDEag	**0.000000E+00**	3.702054E-01	3.7104790E-01	**0.000000E+00**	2.168404E-21	1.062297E-20
	jDEsoco	**0.000000E+00**	4.1102E-01	3.8359E-01	0.0812907	7.3206E-01	2.9943E-01
C17	UMOEAs	**5.694188E-23**	**1.210981E-17**	**2.126749E-17**	**5.186611E-22**	**9.653885E-03**	**1.320570E-02**
	εDEag	1.463180E-17	1.249561E-01	1.9371970E-01	2.165719E-01	6.326487E+00	4.986691E+00
	jDEsoco	0.0302187	8.8958E+01	9.9131E+01	2.9943E-01	5.0398E+02	4.4832E+02
C18	UMOEAs	**2.451780E-27**	**7.761344E-26**	**7.179142E-26**	**1.418552E-24**	**8.103770E-21**	**1.859545E-20**
	εDEag	3.731440E-20	9.678765E-19	1.8112340E-18	1.226054E+00	8.754569E+01	1.664753E+02
	jDEsoco	0.1100901	4.0500E+02	7.3762E+02	17.5655328	3.0849E+02	3.0538E+02