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

Testing United Multi-Operator Evolutionary Algorithms on the CEC2014 Real-Parameter Numerical Optimization

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

***Abstract*—This paper puts forward a proposal for combining multi-operator evolutionary algorithms (EAs), in which three EAs, each with multiple search operators, are used. During the evolution process, the algorithm gradually emphasizes on the best performing multi-operator EA, as well as the search operator. The proposed algorithm is tested on the CEC2014 single objective real-parameter competition. The results show that the proposed algorithm has the ability to reach good solutions.**

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

I. Introduction

An optimization problem is an abstraction of the problem of selecting the best possible alternative of a vector from a set of candidate options[1]. Optimization is used for problems arising in network design and operation, finance, support vector machine, and many other engineering areas. In unconstrained global optimization, the objective functions may possess different properties, such as linear and /or nonlinear, continuous or discontinuous, and unimodal or multimodal, and their combination. In addition, the presence of large number of variables associated with such complex functions pose serious challenges to any optimization algorithm. This makes such problems a challenging research area in the evolutionary computation field.

Over the last few decades, EAs have shown their ability to successfully solve both constrained and unconstrained optimization problems. The EAs family contains different algorithms, such as the genetic algorithm (GA) [2], differential evolution (DE) [3], evolution strategies (ES) [4] and evolutionary programming (EP) [5]. Although there have been many EAs introduced in the literature, no single EA that performs consistently well for all types of problems. For instance, GA was well suited to parallel computing, and was able to solve noisy problems. However, their convergence pattern was slow in comparison with that of DE. Also, DE was suitable when the feasible patches were parallel to the axes but it could become stuck in a local optimum in multimodal functions, and the same is true for many other EAs. Due to this fact, the concept of multi-method EAs and multi-operator EAs have emerged.

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

The main idea of multi-method EAs is to utilize the strength of different EAs in dealing with different types of problems. Vrugt *et al*. [6] introduced an algorithm, known as A Multi-ALgorithm Genetically Adaptive Multiobjective (AMALGAM), that has been proven to be a powerful approach for solving multiobjective problems. Later, Vrugt *et al*. [7] 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.

In regard to multi-operator algorithms, Elsayed *et al.* [8] 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 [9, 10]. Brest *et al*. [11] proposed a DE algorithm which embedded a self-adaptation mechanism for parameter control. In it, the population was divided into sub-populations to apply more DE strategies, and a population diversity mechanism was also introduced. Yong *et al.* has recently proposed a composite DE algorithm (CoDE) [12], 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. 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. Mallipeddi *et al.* [14] proposed an ensemble of mutation strategies and control parameters with DE (EPSDE), in which a pool of distinct mutation strategies, along with a pool of values for each control parameter, coexisted throughout the evolutionary process and competed to produce offspring.. Also the idea of multi-population EA was recently addressed in [15-18].

In this research, a united multi-operator EAs (UMOEAs) is introduced. In it, three subpopulations are initiated with the same individuals, but each subpopulation is then 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 populations are kept on hold. After this cycle, information from the best performing population is used to update some individuals of the worst performing population,

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

and subsequently all 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[19], which contains 30 test problems, with different mathematical properties, with 10, 30, 50 and 100 dimensions. From the results, the proposed algorithm shows consistently ability to obtain good solutions.

This paper is organized as follows: after the introduction, section II presents a brief overview on GA, DE and ES. 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 considered in this research.

A. Differential Evolution

The basic search operators of DE are discussed here.

A.1. Mutation

In the basic mutation, 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 integer numbers [1, PS], $r_1 \neq r_2 \neq r_3 \neq z$, 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 this operation is to explore the search space and maintain diversity. For more strategies and their details, readers are referred to Das and Suganthan [20].

A.2. Crossover

The DE family of algorithms uses two crossover schemes. In this research, we use the binomial crossover, because it is widely accepted and is superior to the exponential one [21].

This crossover type 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 $\vec{v}_{z,t}$.

The selection process then takes place, in which $\vec{u}_{z,t}$ is selected if it is better, based on the objective function, than ($\vec{x}_{z,t}$).

B. Genetic Algorithms

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

B.1. MPC-GA

In MPC-GA, first an initial population is randomly generated, with size PS. Then an archive pool (A_{arch}) is filled with the best m individuals (based on their constraint violations and/or fitness function). Then a tournament selection procedure, with size tc, takes place, from which the best individual is chosen and saved in the selection pool. For the crossover operation, with a crossover rate, for each three consecutive individuals in the selection pool, three offspring are generated as:

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

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

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

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

B.2. SBX

SBX is widely used in practice. The probability distribution of β in SBX is similar to the probability distribution of β in binary-coded crossover. From a pair of parents $\vec{x}^1 = (x_1^1, x_2^1, \ldots, x_n^1)$ and $\vec{x}^2 = (x_1^2, x_2^2, \ldots, x_n^2)$, two offspring $\vec{y}^1 = (y_1^1, y_2^1, \ldots, y_n^1)$ and $\vec{y}^2 = (y_1^2, y_2^2, \ldots, y_n^2)$ are generated in the following manner:

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:

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

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

SBX has been found to work well in many test problems that have a continuous search space, when compared to other real-coded crossover implementations. 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 also useful in problems where the bounds of the optimum point are not known and where there are multiple optima [22].

The mutation operator is applied to maintain genetic diversity from one generation to another. The non-uniform mutation is well-known in the literature. In it, the step size is decreased as the generations increase, thus making a uniform search in the initial stage and very little at the later stages [23]. 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:

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

where $\bar{x}_j$ and $\underline{x}_j$ are the upper and lower boundary 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 well for problems when a solution only needs to be refined during the later stages of the execution of an algorithm

C. Evolution Strategy

In this study, we consider only one ES variant, known as: Covariance Matrix Adaptation-ES (CMA-ES), as it showed its superiority to any other ES variants.

A.1. CMA-ES

CMA adapts a full covariance matrix of a normal search (mutation) distribution [27]. CMA-ES uses the following steps:

1- Create an initial population and evaluate the fitness function.

2- The best μ_{CMA} individuals are selected as a parental vector, and their centre is calculated according to:

$$\vec{x}_w^t = \sum_{k=1}^{\mu_{CMA}} w_k \vec{x}_k, \quad (11)$$

where

$$\sum_{z=1}^{\mu_{CMA}} w_z = 1\,,\ w_1 \geq w_2 \geq \cdots \geq w_{\mu_{CMA}} \geq 0 \quad (12)$$

3- Updated population is created according to:

$$\vec{x}_{z=1:PS}^{t+1} = \vec{x}_w^t + \sigma^t B^t Q^t G_{z=1:PS}, \quad (13)$$

where $G_{k=1:PS}$ are independent realizations of a D-dimensional standard normal distribution with zero-mean and a covariance matrix equal to the identity matrix I. These base points are rotated and scaled by the eigenvectors B^t and the square root of the eigenvalues Q^t of the covariance matrix C^t. The C^t, and the global step-size σ^t are continuously updated after each generation t [28]. Please note that the CMA-ES version using in this paper is available on https://www.lri.fr/~hansen/cmaes.m.

III. United Multi-operator Evolutionary Algorithms (UMOEAs)

In this section, the proposed algorithm is described as well as the improvement measure used in this research is elaborated.

A. UMOEAs

The pseudo-code of the proposed algorithm is given in this Algorithm 1. To start with, all multi-operator algorithms start with the same population, of a size PS, which is randomly generated using a uniform distribution. Let us name each subpopulations size as PS_1, PS_2 and PS_3, respectively. The solutions in the first population are evolved by a multi-operator DE algorithm, while the second and third subpopulations are evolved using a multi-operator GA algorithm and a multi-operator ES algorithm, respectively.

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 (14), otherwise it will be generated using (15).

$$\vec{v}_{z,t} = \vec{x}_{\varphi,t} + F_z.\left(\vec{x}_{r_1,t} - \vec{x}_{r_2,t}\right), \quad (14)$$

where φ is a random integer number between 1 and $\frac{PS_1}{2}$. It is worthy to mention here that φ is selected after PS_1 is sorted, based on the fitness function.

$$\vec{v}_{z,t} = \vec{x}_{i,t} + F_z.\left(\left(\vec{x}_{r_1,t} - \vec{x}_{r_2,t}\right) + \left(\vec{x}_{best,t} - \vec{x}_{i,t}\right)\right) \quad (15)$$

Note also that the binomial crossover with a crossover rate (Cr_z) is used to generate a final offspring. The reason for using the binomial, in this paper, is due to its superiority to the exponential one [21]. The vales of F_z and Cr_z are adaptively calculated as will be shown in section IV.

If the new offspring is better (based on the fitness function) than its parent, 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 mechanism 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,

ALGORITHM I. UNITED MULTI-OPERATOR EVOLUTIONARY ALGORITHMS PSEUDO-CODE

- Generate initial population of a size *PS*; each variable is generated within its boundaries.
- set $PS = PS_1 = PS_2 = PS_3$
  - 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** $period < CS$ **&** $FFEs \leq MixStage$
  - $period = period + 1$;
  - 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 update s_3 and s_4
    - update $prob_2 = max(0.05, \frac{s_3}{s_3 + s_4})$
  - evolve PS_3 using multi-operator ES:
  - record $bf_{t,1}, bf_{t,2}, bf_{t,3}$
- **end**
- **if** $mod(period, CS) = 0$
  - Decide which multi-operator algorithm is the best ($best\_EA$), as shown in III.B.
- **end**
- **if (** $period > CS$ **&** $period < 2CS$) or $FFEs > MixStage$
  - **if** $best\_EA == 1$
    - evolve PS_1 using multi-operator DE; **else if** $best\_EA == 2$
    - evolve PS_2 using multi-operator GA; **else**
    - evolve PS_3 using multi-operator ES;
  - **end**
- **end**
- **if** $period == 2CS$
  - calculate the mean($\overline{x}$) and standard deviation (σ) vector of the μ best individuals of the $best\_EA$, and replace the *k*-th individual with $\vec{x}_k = N(\overline{x}, \sigma)$, where κ is the second worst individual.
  - replace the worst individuals in the worst performing multi-operator algorithms by the best individual found so far.
  - period= $s_1 = s_2 = s_3 = s_4 = 0$; $prob_1 = prob_2 = 0.5$;
- **end**

update FFEs; $t = t + 1$;

end

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 $\frac{s_3}{s_3 + s_4}$.

As mentioned earlier, CMA-ES was found the best ES variant in the literature. Therefore, instead of losing its search power, by using another ES variant, we decided to use it only as a multi-operator ES.

The abovementioned process is repeated for CS generation (named as a cycle). Then, the best performing multi-operator is selected (as will be shown in III.B) to evolve only its population for the subsequent CS generations, while the other sub-populations are kept on hold. Once this step is finished, all parameters ($s_1, s_2, s_3, s_3, prob\_1$ and $prob_2$) are re-set to their initial values and an information scheme is applied, such that:

- The worst individual in each subpopulation is replaced with the best individual found so far, with a confirmation that the solution is not redundant.
- For the best μ individuals in the successful population, the mean and standard deviation vectors ($\overline{x}$ and σ, respectively) are calculated, as:

$$\overline{x}_j = \frac{\sum_{i=1}^{\mu} x_{i,j}}{\mu} \tag{16}$$

$$\sigma_j = \frac{\sqrt{\sum_{i=1}^{\mu} (x_{i,j} - \overline{x}_j)^2}}{\mu} \tag{17}$$

then the second worst individual in each sub-population, which was on hold, is replaced by the a new solution vector that is generated as:

$$x_{z,j} = N(\overline{x}_j \sigma_j) \tag{18}$$

After passing information, each subpopulation is sorted and the mean vector of CMA-ES is updated, if required, using (11).

Note that the abovementioned procedure of using three multi-operator algorithms is used up to a predefined level, here it is $\frac{maxFFEs}{2}$ generations. After that level, the best algorithm is used to evolve its population until an overall stopping criterion is met.

B. Handling Bound Constraints

In this research, the bound constrains are handled as follows

$$x_{z,j} = \begin{cases} max\ (\underline{x}_j, 2 \times \underline{x}_j - \ x_{z,j}) & x_{z,j} < \underline{x}_j \\ min\ (\overline{x}_j, 2 \times \overline{x}_j - \ x_{z,j}) & x_{z,j} > \overline{x}_j \end{cases}, \tag{19}$$

where $\underline{x}_j$ and $\overline{x}_j$ are the lower and upper boundary of x_j.

C. Deciding the Best Performing MOEA

To decide which multi-operator algorithm could be used after every cycle. The following steps are conducted:

1- At each generation, record the absolute function error of the best solution found and the optimal solution obtained by each MOEA, such that: $bf_{tcs,\mathcal{B}} = |f(x_{best,\mathcal{B}}) - f(x^*)|$, where $\mathcal{B}$ = 1, 2 and 3 and refers to the multi-operator DE, multi-operator GA, and multi-operator ES, respectively, while $tcs = CS/2 : CS$ generations.

2- Fit an exponential model (ae^{bx}) for each bf, and generate the exponential model coefficients (a and b).

3- Consequently, for each multi-operator, the expected absolute error after subsequent CS generations is calculated.
4- The algorithm with the minimum expected absolute error is selected as the best multi-operator algorithm.

D. 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 in generating 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 (16) - (17).
2- The reason for using three multi-operator algorithms only up to $MixStage$ fitness evaluations (here equal to $\frac{maxFFEs}{2}$), and not for all the evolution process, is to reduce the time complexity of the algorithm, especially at this stage the decision of which multi-operator algorithm performs best may be justifiably made
3- The point behind reusing three multi-operator algorithms, instead of one, after every $2CS$ generation, is that passing good information for a poor multi-operator algorithm may help it to reach better solutions latter on.
4- 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.

IV. Experimental Results

In this section, the performance of the proposed algorithm is discussed and analyzed by solving a set of problems presented in the CEC2014 competition on real-parameter optimization [19], which contains 30 test problems with 10, 30, 50 and 100 dimensions. The algorithm was run 51 times for each test problem, where the stopping criterion was to run for up to 10,000D FEs. The algorithm was coded using Matlab R2012b, and was run on a PC with a 3.4 GHz Core I7 processor with 16 GB RAM, and windows 7.

To begin with, all parameter values are provided in Table I. To add to this, Cr and F are self-adaptively calculated, as follows:

- At $t = 1$, each individual in PS_1 is assigned with $\dot{F}_z$ and $\dot{Cr}_z$, where $\dot{F}_z = N(0.5,1)$ and $\dot{Cr}_z = N(0.5,1)$. If the value is less than 0.1 or larger than 1.0, it is truncated to 0.1 and 1, respectively.
- Then, to generate a new solution, based on (14) or (15), both parameters are calculated as follows:

Table I. Details of All Parameters values

General: $PS_1 = PS_2 = PS_3 = 100$, $CS = 50$, for 10D and 100 for all other dimensions and $\mu = 2$.

DE: $\varphi \in [1, \frac{PS_1}{2}]$ in (14) [29]. F and Cr are self-adaptively calculated as shown in (20) and (21).

GA: $p = 0.1$, $arch = \frac{PS_1}{2}$, Cr=100%, *mutation rate* = 0.1, *tournament selection size* is randomly 2 or 3, $\eta = 3$ and $b = 5$ [26].

CMA-ES: $\mu_{CMA} = 0.5PS\_3$, $\sigma = 1.5$

$$F_z = \begin{cases} \dot{F}_{r_1} + rand_1 \times (\dot{F}_{r_2} - \dot{F}_{r_3}), & if\ (rand_2 < \tau_1) \\ rand_3 & otherwise \end{cases} \quad (20)$$

$$Cr_z = \begin{cases} \dot{Cr}_{r_1} + rand_4 \times (\dot{Cr}_{r_2} - \dot{Cr}_{r_3}), if\ (rand_5 < \tau_1) \\ rand_6 \qquad\qquad otherwise \end{cases} \quad (21)$$

where $rand_\Gamma \in [0,1]$ for $\Gamma = 1,2 \ldots,6$ and $\tau_1 = 0.75$. If the value is less than 0.1 or larger than 1, it is truncated to 0.1 and 1, respectively.

- If the new offspring is better than its parent, then $\dot{F}_z = F_z$ and $\dot{Cr}_z = Cr_z$.

A. Results for 10D

The computational results ($f(x_{best}) - f(x^*)$) of UMOEAs for the 10D problems are shown in Table II. From the results obtained, it was obvious that UMOEAs performed excellent in unimodal problems (F01:F03). For multimodal problems (F04:F16), UMOEAs was able to obtain the optimal solutions on six problems, while it was very close to the optimal solution for the rest. For hybrid functions (F17:

Table II. Results for 10D

	Best	Worst	Median	Mean	Std
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F04	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F05	**0.0000E+00**	2.0148E+01	2.0051E+01	1.6895E+01	7.3603E+00
F06	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F07	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F08	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F09	9.9496E-01	9.9496E+00	3.9798E+00	4.6504E+00	1.9437E+00
F10	6.2454E-02	3.6023E+00	1.8736E-01	6.3404E-01	1.1413E+00
F11	3.5399E+00	6.8628E+02	1.2862E+02	1.5908E+02	1.6367E+02
F12	**0.0000E+00**	1.5119E-02	**0.0000E+00**	8.8934E-04	3.5928E-03
F13	2.8347E-03	2.2347E-02	8.4565E-03	9.4554E-03	5.0520E-03
F14	1.8202E-02	1.9796E-01	8.1067E-02	8.3410E-02	3.3301E-02
F15	3.2154E-01	1.2021E+00	6.6304E-01	6.5615E-01	2.0031E-01
F16	2.0719E-01	2.7579E+00	1.5503E+00	1.5529E+00	6.4679E-01
F17	**0.0000E+00**	9.2044E+01	1.6194E+00	9.8968E+00	1.6577E+01
F18	**0.0000E+00**	3.9798E+00	9.9496E-01	9.9496E-01	9.5433E-01
F19	1.9432E-02	1.0195E+00	5.6273E-02	1.5661E-01	2.5583E-01
F20	5.9453E-05	1.4036E+00	2.1172E-01	2.9843E-01	2.8495E-01
F21	3.8216E-05	8.2245E+00	3.3548E-01	5.5689E-01	1.1326E+00
F22	8.3735E-06	6.4507E-01	2.6868E-01	2.3474E-01	1.9678E-01
F23	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F24	1.0000E+02	1.2118E+02	1.1231E+02	1.1253E+02	3.6055E+00
F25	1.0000E+02	1.9855E+02	1.2529E+02	1.3194E+02	2.4292E+01
F26	1.0001E+02	1.0007E+02	1.0002E+02	1.0002E+02	1.4847E-02
F27	6.9818E-01	2.0000E+02	1.8837E+00	1.7345E+01	5.3817E+01
F28	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F29	1.0000E+02	2.2120E+02	2.0000E+02	2.0322E+02	2.1594E+01
F30	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**

TABLE III. RESULTS FOR 30D

	Best	Worst	Median	Mean	Std
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F04	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F05	1.9998E+01	2.0437E+01	2.0000E+01	2.0050E+01	1.2597E-01
F06	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F07	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F08	**0.0000E+00**	3.9798E+00	9.9496E-01	1.3464E+00	1.1572E+00
F09	3.9798E+00	1.5919E+01	8.9546E+00	8.8378E+00	2.7762E+00
F10	1.2633E+00	1.2086E+02	5.7752E+00	8.9251E+00	1.6538E+01
F11	1.3254E+02	3.2300E+03	1.3547E+03	1.4588E+03	7.9087E+02
F12	**0.0000E+00**	9.0581E-03	1.8568E-03	2.5570E-03	2.3493E-03
F13	1.4689E-02	8.3997E-02	5.5679E-02	5.4565E-02	1.5475E-02
F14	1.3918E-01	3.2366E-01	2.0890E-01	2.0361E-01	4.0160E-02
F15	2.2988E+00	4.8044E+00	3.1164E+00	3.2456E+00	5.2092E-01
F16	8.0072E+00	1.1426E+01	1.0083E+01	9.9269E+00	7.4097E-01
F17	2.7772E+01	1.5818E+03	9.6351E+02	9.7741E+02	3.6055E+02
F18	3.9494E+00	5.9259E+01	1.9478E+01	2.1214E+01	1.0418E+01
F19	2.1027E+00	4.9401E+00	3.5409E+00	3.5573E+00	6.8967E-01
F20	4.1889E+00	2.3428E+01	9.7627E+00	1.1018E+01	4.4544E+00
F21	6.7115E+00	8.8759E+02	3.0691E+02	3.3816E+02	2.1940E+02
F22	7.8765E-01	2.7944E+02	5.3104E+01	9.5412E+01	8.0479E+01
F23	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F24	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F25	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F26	1.0003E+02	1.0015E+02	1.0007E+02	1.0008E+02	2.8264E-02
F27	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F28	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F29	2.0000E+02	2.0909E+02	2.0601E+02	2.0480E+02	2.9796E+00
F30	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**

F22), UMOEAs was able to obtain the optimal solution in two occasions, and very close to the optimal solution for the rest test problems. However it trapped in local solutions for all the composition test instances (F23:F30).

B. Results for 30D

The computational results of the proposed algorithm for the 30D test problems are shown in Table III.

From the results, UMOEAs was able to obtain the optimal solution on unimodal problems (F01:F03). For multimodal problems, UMOEAs was robust for F04, F06 and F07, while it was very close to the optimal solution for the rest problems. For hybrid functions, the best solutions obtained were close to the optimal; however it often trapped in local solutions. For the composition problems, UMOEAs got stuck in local solutions.

C. Results for 50D

UMOEAs's computational results of the 50D test problems are shown in Table IV.

From Table IV, UMOEAs was able to obtain the optimal solutions in unimodal problems (F01:F03). For multimodal problems, UMOEAs was robust in solving F07, efficient in solving F04, F06, F07 and F08, while it got stuck in local solutions for the rest test problems. This was also the situation for the hybrid and composition problems, although its performance in solving the hybrid problems is a little bit better than its performance in solving the composition problems.

TABLE IV. RESULTS FOR 50D

	Best	Worst	Median	Mean	Std
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F04	**0.0000E+00**	3.9866E+00	**0.0000E+00**	7.8169E-02	5.5824E-01
F05	1.9999E+01	2.0484E+01	2.0008E+01	2.0116E+01	1.7971E-01
F06	**0.0000E+00**	1.5764E+00	**0.0000E+00**	6.0329E-02	3.0164E-01
F07	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F08	**0.0000E+00**	1.2934E+01	3.9798E+00	4.2920E+00	3.0407E+00
F09	1.0945E+01	2.6864E+01	1.8904E+01	1.9361E+01	3.6065E+00
F10	2.9445E+00	3.6574E+02	1.4879E+01	7.5519E+01	9.4506E+01
F11	5.1140E+01	7.2694E+03	4.2520E+03	3.9779E+03	1.9904E+03
F12	**0.0000E+00**	3.5210E-03	8.3144E-04	1.1125E-03	9.0616E-04
F13	5.1169E-02	1.4507E-01	1.0184E-01	9.8518E-02	2.0232E-02
F14	1.5815E-01	2.9943E-01	2.2299E-01	2.2434E-01	3.3788E-02
F15	1.9838E+00	7.3589E+00	5.4349E+00	5.4600E+00	9.6612E-01
F16	1.7632E+01	2.0716E+01	1.9249E+01	1.9208E+01	7.2823E-01
F17	1.2863E+03	3.6025E+03	2.4129E+03	2.4501E+03	4.5975E+02
F18	1.7161E+01	2.3467E+02	7.3680E+01	9.0006E+01	6.1214E+01
F19	7.2231E+00	1.9386E+01	1.1668E+01	1.1731E+01	2.0700E+00
F20	2.4250E+01	1.6553E+02	6.3973E+01	7.0815E+01	2.9985E+01
F21	6.1101E+02	2.1939E+03	1.4660E+03	1.4670E+03	3.9168E+02
F22	2.8061E+01	7.1403E+02	3.4735E+02	3.6419E+02	1.7639E+02
F23	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F24	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F25	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F26	1.0005E+02	2.0000E+02	1.0015E+02	1.0406E+02	1.9575E+01
F27	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F28	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F29	2.1044E+02	2.2230E+02	2.1628E+02	2.1638E+02	2.5214E+00
F30	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**

TABLE V. RESULTS FOR 100D

	Best	Worst	Median	Mean	Std
F01	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F02	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F03	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F04	2.1124E+01	2.6842E+01	2.2564E+01	2.3113E+01	1.4634E+00
F05	2.0000E+01	2.0063E+01	2.0000E+01	2.0003E+01	1.0160E-02
F06	**0.0000E+00**	2.7946E+00	5.7677E-01	8.5169E-01	8.7062E-01
F07	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
F08	7.9597E+00	8.6561E+01	2.2884E+01	2.4523E+01	1.0486E+01
F09	4.0793E+01	6.6662E+01	5.3728E+01	5.3845E+01	6.6793E+00
F10	1.3549E+02	5.4553E+03	3.0478E+03	2.9895E+03	1.5700E+03
F11	5.2659E+03	1.0635E+04	7.9323E+03	7.8789E+03	1.4403E+03
F12	8.4500E-05	2.5297E-03	6.0510E-04	6.7674E-04	4.1803E-04
F13	1.3983E-01	2.9804E-01	1.9857E-01	2.0507E-01	3.3606E-02
F14	1.8673E-01	2.7700E-01	2.2707E-01	2.2661E-01	2.1455E-02
F15	8.9694E+00	1.5416E+01	1.1355E+01	1.1655E+01	1.4143E+00
F16	3.9980E+01	4.3720E+01	4.2697E+01	4.2594E+01	7.7180E-01
F17	3.1437E+03	6.6751E+03	5.2092E+03	5.2953E+03	7.9441E+02
F18	2.6448E+02	7.2428E+02	4.0065E+02	4.0972E+02	1.0333E+02
F19	1.6091E+01	6.9883E+01	6.0341E+01	5.8398E+01	8.7355E+00
F20	1.9153E+02	4.6476E+02	3.1161E+02	3.1187E+02	6.6812E+01
F21	2.1095E+03	1.0833E+04	4.1689E+03	4.4189E+03	1.5951E+03
F22	5.8648E+01	1.6983E+03	9.5655E+02	9.2693E+02	3.2174E+02
F23	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F24	2.0000E+02	2.0001E+02	2.0000E+02	2.0000E+02	1.5050E-03
F25	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F26	1.0025E+02	2.0000E+02	2.0000E+02	1.9804E+02	1.3968E+01
F27	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F28	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**
F29	2.2791E+02	2.7500E+02	2.5881E+02	2.5493E+02	1.1799E+01
F30	2.0000E+02	2.0000E+02	2.0000E+02	2.0000E+02	**0.0000E+00**

D. Results for 100D

Table V presents the computational results for the 100Dtest problems. From results obtained, UMOEAs performed well in solving unimodal problems. Furthermore, the algorithm performed well in F06 and F07 and could

reach near optimal solutions for many multi-modal problems, except F10 and F11. For the hybrid and composition problems, UMOEAs's performance was not good enough. However, it was noticed that, for the composition problems, the algorithm was able to reach the same solutions as those obtained in30D and 50D.

E. Computational Complexity

To this end, the computational complexity of the proposed algorithm is calculated based on all problem dimensions. A summary of the results is shown in Table V.

TABLE V. COMPUTATIONAL COMPLEXITY

	T_0	T_1	$\hat{T}_1$	$\frac{(\hat{T}_1 - T_1)}{T_0}$
10D	0.1092	0.935761	4.4822	32.4762
30D		1.219444	5.8698	42.5858
50D		1.483131	6.9143	49.7356

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 three 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 some individuals in the worst performing subpopulations, and hence the three 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 CEC2014 real-parameter benchmark problems, and showed good performance in many occasions.

In the future work, to select the best EA during the evolution process, we wish to analyze the effect of using different statistical models to fit the performance of each EA.

REFERENCES

[1] S. P. Boyd and L. Vandenberghe, *Convex optimization*: Cambridge university press, 2004.

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

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[15] S. Biswas, S. Kundu, D. Bose, S. Das, P. N. Suganthan, and B. K. Panigrahi, "Migrating forager population in a multi-population Artificial Bee Colony algorithm with modified perturbation schemes," in proceeding *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, 2013, pp. 248-255.

[16] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," *Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies,* vol. 76, pp. 214-220, 2002.

[17] D. Bose, S. Biswas, S. Kundu, and S. Das, "A Strategy Pool Adaptive Artificial Bee Colony Algorithm for Dynamic Environment through Multi-population Approach," in *Swarm, Evolutionary, and Memetic Computing*. vol. 7677, B. Panigrahi*, et al.*, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 611-619.

[18] G. Yue-Jiao, Z. Jun, H. S. Chung, C. Wei-Neng, Z. Zhi-Hui, L. Yun, and S. Yu-Hui, "An Efficient Resource Allocation Scheme Using Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 16, pp. 801-816, 2012.

[19] J. J. Liang, B.-Y. Qu, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," Computational Intelligence Laboratory and Nanyang Technological University, China and Singapore, Tech. Rep. 201311, 2014.

[20] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation,* vol. 15, pp. 4-31, 2011.

[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] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in proceeding *IEEE Congress onEvolutionary Computation*, 1996, pp. 312-317.
[28] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation,* vol. 9, pp. 159-195, 2001.
[29] 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.