International Journal on Advanced Science Engineering Information Technology

Vol.7 (2017) No. 5
ISSN: 2088-5334

Analyzing the Scalability Performance of Crossover-First and Self-Adaptive Differential Evolution Algorithms for Complex Numerical Optimization

Jason Teo[#]

[#] *Faculty of Computing & Informatics, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu,Sabah, Malaysia.*
E-mail: jtwteo@ums.edu.my

***Abstract*— Two Crossover-first Differential Evolution (XDE) algorithms as well as four self-adaptive DE algorithms are compared in this study in terms of their optimization accuracy for solving a set of 15 complex, non-linear numerical optimization functions across 4 different dimensions of 10, 30, 50 and 100 optimization variables. XDE is a crossover-first variant of the original DE algorithm where XjDE is the crossover-first variant of the self-adaptive jDE algorithm. The original DE representing a fixed parameter strategy is tested against four self-adaptive algorithms, namely the DESACR, DESACRF, SDE and jDE algorithms. Although XDE is able to outperform XjDE in all 15 test problems for the lowest dimensional benchmark test setting of 10 variables, the crossover-first approach in XjDE is able to improve its performance and obtained better results over XDE in some of the test problems for the higher-dimensional benchmark test settings of 30, 50 and 100 variables. As such, this shows that there is some merit in adopting the crossover-first approach into the self-adaptive XjDE algorithm since the CR and F parameters are automatically adjusted and optimized by the algorithm itself as compared to the fixed CR and F in XDE which has to be manually tuned by hand. The results also show that different self-adaptive parameter tuning schemes have significantly different effects on the performance of DE as the number of optimization dimensions increases.**

***Keywords*— crossover-first differential evolution; evolutionary optimization; genetic operations; non-linear optimization.**

I. Introduction

The Crossover-First Differential Evolution (XDE) algorithm was recently introduced as a more efficient optimizer than the original Differential Evolution (DE) algorithm for solving complex, non-linear numerical optimization problems [1-4]. XDE uses a novel sequence of genetic operations for DE in that the crossover operation is first conducted before the mutation operation. This simple modification introduced in XDE by way of reversing the genetic operations in DE has proven to be highly effective in improving the optimization accuracy of DE, particularly in handling expanded and hybrid composition functions with highly complex search spaces [1].

In this paper, we extend the crossover-first approach to another well-known DE variant called jDE [5-7]. In jDE, rather than fixing the tunable parameters for its crossover rate (commonly referred to as CR) and scaling factor (commonly referred to as SF), it adopts a self-adaptive approach where the CR and SF are incorporated into the algorithm and subjected to evolutionary optimization in order to self-adjust and find the best rates for its CR and SF parameters [8]. Since the crossover-first methodology has proven to be effective in XDE, here we employ this crossover-first methodology to jDE, which we refer to as XjDE.

In order to test whether this approach improves upon the performance of XDE, we conducted comprehensive optimization tests on both XDE and XjDE using a set of 15 benchmark suite of non-linear numerical test problems proposed in the 2015 Congress on Evolutionary Computation (CEC2015) competition for global optimization across the four dimensions of 10, 30, 50 and 100 variables [9] which has been widely used in complex numerical optimization studies [10-15].

The rest of the paper is arranged as follows: Section 2 reports on the related work of this paper; Section 3 explains the methodology adopted in this study; Section 4 presents the results and analysis of this work; Section 5 concludes with a summary of the findings and possible future work.

II. Material And Method

A. *Related Material*

XDE algorithm [1] was recently proposed as an improvement over the original DE algorithm [2,18,19]. It proposes a novel reversal of genetic operations in the original DE algorithm.

The basic DE algorithm is a population-based, real-valued, stochastic global optimizer that conducts the following operations in the following order after initialization: 1. mutation; 2. crossover; 3. selection; 4. repeat until termination. It also requires three user-defined parameters to be set prior to the optimization run: 1. F: scaling factor; 2. CR: crossover rate; and 3. NP: population size. The reader may refer to [2] for a detailed treatment of DE including the description of the algorithm in pseudocode. In the description that follows, only the key genetic operations are described. In brief, given a minimization problem *f*:

$$f(x)^* = x_i \in \Omega \min f(x_i) \tag{1}$$

where x_i is a vector with D dimensions, x^* is the global solution and $\Omega \subseteq R^D$, DE will attempt to optimize the vector of variables $x = \{x_1, x_2, \ldots, x_D\}$, where $x_i^G = \{x_{i,1}^G, x_{i,2}^G, \ldots, x_{i,D}^G\}$ represents the *i*th individual in the population of solutions of the *G*th generation of optimization iteration.

Importantly, a new trial solution is generated in the following order:

i. Mutation: for each parent $x^{G,i}$, a new vector is created as follows:

$$v_i^{G+1} = \{x_{r1}^G + F.(x_{r2}^G - x_{r3}^G\} \tag{2}$$

where $r1$,$r2$, and $r3$ are randomly chosen from [1,*NP*] and $i \neq r1 \neq r2 \neq r3$.

ii. Crossover: for each parent $x^{G,i}$, a trial solution is created as follows:

$$u_{i,j}^{G+1} \begin{cases} v_{i,j}^{G+1} & \text{if } R_j < CR \text{ or } j = j_{rand} \\ x_{i,j}^G & \text{otherwise.} \end{cases} \tag{3}$$

where R_j is a uniform random [0,1] and j_{rand} is random integer [1,*D*].

iii. Selection: the new trial solution competes with the parent for survival to the next optimization iteration:

$$x_{i,j}^{G+1} \begin{cases} u_i^{G+1} & \text{if } f(u_i^{G+1}) < f(x_i^G) \\ x_i^G & \text{otherwise.} \end{cases} \tag{4}$$

In XDE, the crossover operation is first conducted before the mutation operation as follows [1]:

i. Crossover: for each parent $x^{G,i}$, a new vector is created as follows:

$$v_{i,j}^{G+1} \begin{cases} x_{r1,j}^G & \text{if } R_j < CR \text{ or } j = j_{rand} \\ x_{i,j}^G & \text{otherwise.} \end{cases} \tag{5}$$

where where $r1$ is randomly chosen from [1,*NP*], R_j is a uniform random [0,1], j_{rand} is random integer [1,*D*] and $i \neq r1$.

ii. Mutation: for each parent $x^{G,i}$, a trial solution is created as follows:

$$u_{i,j}^{G+1} \begin{cases} x_{r2,j}^G + \text{F}.(x_{r3,j}^G - x_{r4,j}^G) & \text{if } R_j < MR \\ v_{i,j}^{G+1} & \text{otherwise.} \end{cases} \tag{6}$$

where $r2$,$r3$, and $r4$ are randomly chosen from [1,*NP*], R_j is a uniform random [0,1] and $i \neq r1 \neq r2 \neq r3 \neq r4$. *MR* denotes the explicit mutation rate parameter that is tunable.

iii. Selection: the new trial solution competes with the parent for survival to the next optimization iteration:

$$x_{i,j}^{G+1} \begin{cases} u_i^{G+1} & \text{if } f(u_i^{G+1}) < f(x_i^G) \\ x_i^G & \text{otherwise.} \end{cases} \tag{7}$$

B. jDE Algorithm

The jDE algorithm functions in exactly the same way as DE with the exception that the CR and SF parameters are self-adapted in the following manner [3]:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 * F_u, & \text{if } \text{rand}_2 < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases} \tag{8}$$

$$CR_{i,G+1} = \begin{cases} rand_3, & \text{if } \text{rand}_4 < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

where new control parameters F and CR are produced before a new trial solution is generated using the crossover and mutation operations. τ1 and τ2 are set to 0.1 and represent the probabilities of adjusting CR and F.

C. XjDE Algorithm

In the proposed adoption of the crossover-first approach to the genetic operations in jDE, a similar process of conducting the generation of trial solutions is implemented as in XDE previously described in Section II. The only difference is that the self-adaption of CR and F is conducted as in jDE explained in Section III.A above before the trial solutions are generated using the crossover and mutation operations in XjDE in order to allow for the new CR and F values to be produced before the trial solutions are subjected to the new crossover-first operations.

D. Self-Adaptive DE Algorithms

Next, we present the two straightforward but novel self-adaptive modifications to the basic DE algorithm. The two novel algorithms are implemented as straightforward self-adaptive DE whereby the first is called DESACR implements a self-adaptive CR that is encoded into the chromosome and evolved and the F is randomized between 0 and 1 for each offspring generated. The self-adaptive CR is mutated using the standard DE genetic operations similar to how the actual design variables are mutated and is conducted first before conducting the standard crossover operation in order to capture the goodness of the newly-generated CR if a superior offspring is generated through this new CR. The self-adapted CR is then adopted by the new offspring and used in subsequent crossover operations should this offspring survive into the next generation. A lower and upper bound threshold of 0 and 1 respectively are set for the self-adaptive CR in order to maintain the CR within this range.

The second novel algorithm is called DESACRF and version of DE is also implemented in a straightforward fashion as described above for DESACR except that now both the CR and F are self-adapted in this manner. Again, both CR and F are mutated first and then used for the actual mutation and crossover operations for the actual design variables for the same reasons as described earlier. Again, lower and upper bound thresholds of 0 and 1 are set for both the self-adaptive CR and F.

The other two self-adaptive DE algorithm tested in this study are jDE [3] and SDE [17]. SDE implements a straightforward self-adaptation of the F parameter where each individual will optimize its own F value while jDE implements self-adaptation for both CR and F based on a new parameters, τ1 and τ2, which are thresholds that when the conditions are met, the F and CR values will undergo randomized changes.

E. *Experimental Setup*

Both the existing XDE and the proposed XjDE algorithms are tested using the CEC 2015 Global Optimization Competition test suite of 15 non-linear numerical optimization benchmark functions which is implemented in the C++ language [4].

TABLE I
CEC 2015 BENCHMARK FUNCTIONS

No.	Functions	*F**
1	Rotated High Conditioned Elliptic Function	100
2	Rotated Cigar Function	200
3	Shifted and Rotated Ackley's Function	300
4	Shifted and Rotated Rastrigin's Function	400
5	Shifted and Rotated Schwefel's Function	500
6	Hybrid Function 1 (*N*=3)	600
7	Hybrid Function 2 (*N*=4)	700
8	Hybrid Function 3(*N*=5)	800
9	Composition Function 1 (*N*=3)	900
10	Composition Function 2 (*N*=3)	1000
11	Composition Function 3 (*N*=5)	1100
12	Composition Function 4 (*N*=5)	1200
13	Composition Function 5 (*N*=5)	1300
14	Composition Function 6 (*N*=7)	1400
15	Composition Function 7 (*N*=10)	1500

All parameters were set according to the competition guidelines and tested using all four optimization dimensions set for the competition, which are 10, 30, 50 and 100 dimensions (D) representing the scalability of the algorithms in terms of their optimization performance. The maximum number of evaluations allowed was 10,000 * D.

The details of the test functions are as detailed in Table 1. All final results have been subtracted by the amount of the global optimum shift value as denoted by F*. The accuracy threshold is set at 1E-08 as per the competition rules. NP is set to 100 and MR is set to 0.5 in both XDE and XjDE.

III. RESULTS AND DISCUSSION

The results are presented in the following sections as follows: average best values found from each algorithm from the experiment using dimensions of 10 and 30 variables are presented in Subsection A. This is followed by the results obtained from using dimensions of 50 and 100 variables in Subsection B.

Since all the test functions are minimization problems with solutions centered around 0, the results are shown are the average of the lowest objective value returns over 51 repeated runs. The better performing algorithm is highlighted for its results for each of the respective 15 test functions. Figure 1 below shows a summary of the overall number of test functions won by each algorithm or drawn.

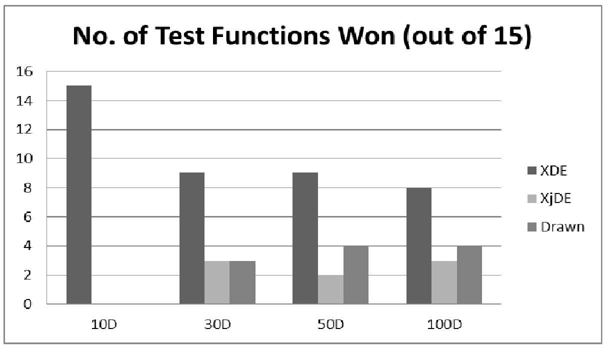

Fig. 1. Comparison of XDE vs. XjDE.

A. *XDE vs XjDE: Results from 10D & 30D*

TABLE II
LEFT: RESULTS FOR 10D; RIGHT: RESULTS FOR 30D

D = 10 Variables		*D = 30 Variables*	
XDE	**XjDE**	**XDE**	**XjDE**
1.34E+07	4.13E+07	**6.19E+04**	1.96E+05
2.18E+05	1.02E+07	5.49E+03	**1.69E+03**
2.03E+01	2.05E+01	**2.01E+01**	2.02E+01
7.40E+01	1.43E+02	**4.39E+00**	1.09E+01
2.92E+03	4.56E+03	**6.54E+01**	3.68E+02
2.98E+06	6.13E+06	**7.88E+01**	1.10E+02

1.23E+01	1.68E+01	6.17E-01	**2.66E-01**
9.15E+05	1.44E+06	6.64E+00	**1.36E+00**
1.04E+02	1.05E+02	**1.00E+02**	**1.00E+02**
1.35E+06	2.06E+06	**2.24E+02**	2.30E+02
6.81E+02	9.02E+02	**1.72E+02**	1.73E+02
1.07E+02	1.08E+02	**1.02E+02**	**1.02E+02**
1.06E+02	1.18E+02	**2.68E+01**	2.77E+01
3.34E+04	3.44E+04	**5.79E+03**	6.32E+03
1.92E-02	1.60E+01	**0.00E+00**	**0.00E+00**

For the test setting with the number of optimization dimensions set to 10 variables, XDE outperformed XjDE for all 15 test functions as shown in Table II (left). However, when the dimensionality of the test problems were increased to 30 variables, XjDE could be seen to perform at par or better than XDE in 6 out of the 15 test problems as shown in Table II (right).

B. XDE vs XjDE: Results from 50D & 100D

For the test setting with the number of optimization dimensions set to 50 variables, similar to the results obtained from the 30 variable setting, XjDE could be seen to perform at par or better than XDE in 6 out of the 15 test problems as shown in Table III (left). XjDE's performance against XDE improved to 7 out of the 15 test problems when the optimization setting was at its most challenging dimension of 100 variables as shown in Table III (right).

TABLE III
LEFT: RESULTS FOR 50D; RIGHT: RESULTS FOR 100D

D = 50 Variables		*D = 100 Variables*	
XDE	**XjDE**	**XDE**	**XjDE**
5.47E+07	1.95E+08	**1.72E+08**	4.44E+08
8.67E+03	**7.86E+03**	5.32E+03	**3.44E+03**
2.11E+01	**2.11E+01**	**2.13E+01**	**2.13E+01**
3.43E+02	3.46E+02	8.53E+02	**8.52E+02**
1.26E+04	1.28E+04	**2.98E+04**	3.00E+04
2.07E+06	1.22E+07	**1.77E+07**	1.05E+08
4.58E+01	4.69E+01	**1.37E+02**	1.39E+02
5.29E+05	3.55E+06	**6.74E+06**	4.80E+07
1.05E+02	**1.05E+02**	**1.08E+02**	**1.08E+02**
4.53E+04	**4.01E+04**	**3.99E+03**	1.17E+04
4.34E+127	8.43E+127	**5.77E+02**	6.62E+02
1.09E+02	**1.09E+02**	1.19E+02	**1.18E+02**
2.16E+02	2.20E+02	**4.62E+02**	4.64E+02
6.65E+04	6.76E+04	**1.09E+05**	**1.09E+05**
0.00E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**

The self-adaptive tuning mechanism present in XjDE for automatically adjusting the CR and F parameters was probably the main reason why it was able to improve its performance in the higher dimensional settings. As the number of dimensions increases, DE is known to be less effective in terms of its performance due to the static nature of the CR and F. Since XjDE was able to automatically tune these parameters during the run, hence it was able to adjust these parameters to more optimum values in order to improve its search procedure in the tests involving 30, 50 and 100 dimensions. Hence, this scalability analysis has shown the promise of adopting a self-adaptive approach to optimizing the DE algorithm for solving global optimization problems with a high number of variables.

C. Scalability Analysis of DE vs Self-Adaptive DEs

TABLE IV
AVERAGE BEST RESULTS FOR 10D

DE	DESACR	DESACRF	SDE	jDE
0.00E+00	4.10E+07	3.45E+06	3.38E+05	4.80E-04
0.00E+00	1.00E+07	1.82E+05	8.00E+03	**0.00E+00**
2.01E+01	2.05E+01	2.00E+01	2.01E+01	**1.98E+01**
2.46E+01	1.43E+02	1.41E+01	8.88E+00	**3.87E+00**
8.38E+02	4.56E+03	6.05E+02	3.06E+02	**2.25E+02**
7.42E+00	6.13E+06	3.48E+04	2.07E+03	1.39E+01
5.10E-01	1.68E+01	2.01E+00	4.37E-01	**3.14E-01**
2.12E-01	1.44E+06	2.92E+04	6.89E+02	2.24E-01
1.00E+02	1.05E+02	1.00E+02	1.00E+02	1.00E+02
2.17E+02	2.06E+06	6.75E+03	4.31E+02	2.17E+02
2.59E+02	9.02E+02	1.60E+02	1.83E+02	**1.43E+02**
1.02E+02	1.08E+02	1.03E+02	1.03E+02	**1.02E+02**
3.14E+01	1.18E+02	3.30E+01	2.77E+01	**2.73E+01**
6.24E+03	3.44E+04	**3.44E+03**	4.32E+03	5.58E+03

0.00E+00	1.60E+01	1.63E-02	**0.00E+00**	**0.00E+00**

The results for all algorithms tested across the 15 test problems of the CEC 2015 benchmark test suite are shown in Tables 4-7. Each algorithm is tested across four dimensions (D), where D = 10, 30, 50, and 100 as per the competition's guidelines. Their average best solutions found over 51 repeated runs are shown in the tables respectively.

Across all dimensions, the original DE algorithm outperformed both DESACR and DESACRF except for 50D where DESACR performed better in 8 problems compared to 7 for DE respectively. When comparing DESACR against DESACRF, DESACR outperformed DESACRF across all problems in all settings for D. DE and SDE were tied at 7 wins each with 1 draw for 10D while DE outperformed SDE in 30D and 100D. SDE outperformed DE in 50D with 8 wins to 6 with 1 draw.

TABLE V
AVERAGE BEST RESULTS FOR 30D

DE	DESACR	DESACRF	SDE	jDE
3.47E+04	4.10E+07	1.30E+07	4.10E+07	4.29E+04
9.70E-08	1.00E+07	2.18E+05	1.00E+07	1.79E-01
2.09E+01	2.05E+01	**2.03E+01**	2.05E+01	2.04E+01
1.79E+02	1.43E+02	7.40E+01	1.43E+02	**4.25E+01**
6.71E+03	4.56E+03	2.92E+03	4.56E+03	**2.29E+03**
1.45E+03	6.13E+06	2.98E+06	6.13E+06	3.38E+03
5.28E+00	1.68E+01	1.23E+01	1.68E+01	6.71E+00
6.50E+02	1.44E+06	9.15E+05	1.44E+06	**3.53E+02**
1.03E+02	1.05E+02	1.04E+02	1.05E+02	**1.02E+02**
4.10E+02	2.06E+06	1.35E+06	2.06E+06	1.18E+03
4.68E+02	9.02E+02	6.81E+02	9.02E+02	4.78E+02
1.07E+02	1.08E+02	1.07E+02	1.08E+02	**1.06E+02**
1.22E+02	1.18E+02	1.06E+02	1.18E+02	**1.05E+02**
3.32E+04	3.44E+04	3.34E+04	3.44E+04	3.38E+04
0.00E+00	1.60E+01	1.92E-02	1.60E+01	**0.00E+00**

Finally, jDE performed the best amongst all the algorithms when comparing against the original DE where it outperformed DE in all settings except 30D where it was equal to DE with 7 wins each and 1 draw. jDE outperformed DE most greatly in 50D where it won 12, lost 2 and drew 1.

However, it is also interesting to note that the original version of DE although being quite dated is still among the best performers after jDE.

TABLE VI
AVERAGE BEST RESULTS FOR 50D

DE	DESACR	DESACRF	SDE	jDE
3.23E+06	1.60E+08	3.70E+07	2.60E+08	**4.67E+05**
2.71E+02	1.50E+07	1.64E+05	1.83E+04	8.02E+02
2.13E+01	2.07E+01	**2.04E+01**	2.09E+01	2.06E+01
8.20E+02	2.96E+02	1.38E+02	3.12E+02	**8.41E+01**
3.00E+04	8.49E+03	**4.86E+03**	1.10E+04	4.96E+03
6.47E+05	2.20E+07	7.74E+06	1.90E+07	**2.99E+04**
1.31E+02	6.72E+01	5.00E+01	5.04E+01	**4.12E+01**
1.32E+05	1.20E+07	7.17E+06	7.82E+06	**8.26E+03**
1.08E+02	1.08E+02	1.07E+02	1.05E+02	**1.04E+02**
2.05E+03	1.00E+07	3.15E+06	1.46E+06	**1.83E+03**
8.25E+02	7.00E+130	1.00E+130	2.00E+128	1.00E+122
1.18E+02	1.10E+02	1.09E+02	1.11E+02	**1.08E+02**
4.64E+02	2.10E+02	**1.89E+02**	2.16E+02	1.98E+02
1.09E+05	6.66E+04	**6.09E+04**	6.45E+04	6.68E+04
0.00E+00	2.45E+01	8.40E-09	**0.00E+00**	**0.00E+00**

IV. CONCLUSION

In this performance and scalability comparison of two crossover-first algorithms, the fixed XDE algorithm was compared against the self-adaptive XjDE algorithm over 15 challenging numerical optimization problems. It was shown that although the XjDE algorithm performed worse than the XDE algorithm in all 15 test functions for the 10-dimensional problem, XjDE was able to improve its performance for the higher dimensional problems of 30, 50 and 100 variables. XjDE was able to perform at par or better than XDE in 6 out of the 15 test problems in the 30 and 50 variable setting. XjDE returned its best performance with 7 test problems at par or better than XDE for the largest setting of 100 dimensions. Therefore, this study has shown that as the dimensionality of the problem grows, the self-adaptive XjDE algorithm with the crossover-first approach has some merits in terms of solving challenging non-linear numerical optimization problems with complex landscapes.

This study has made a systematic scalability comparison of fixed versus four self-adaptive DE algorithms using 15 benchmark test problems. The fixed parameter and original version of Differential Evolution (DE) was compared against DESACR, DESACRF, SDE and jDE representing four different schemes of self-adapting the various parameter values of DE. Observable differences were found between the fixed and self-adaptive DE algorithms as the number of optimization dimensions were increased.

TABLE VII
AVERAGE BEST RESULTS FOR 100D

DE	DESACR	DESACRF	SDE	jDE
3.23E+06	**0.00E+00**	3.40E+08	**0.00E+00**	2.19E+06
2.71E+02	1.50E+07	2.37E+04	4.82E+03	2.06E+03
2.13E+01	2.09E+01	**2.06E+01**	2.12E+01	2.08E+01
8.20E+02	7.85E+02	7.12E+02	8.73E+02	**1.84E+02**
3.00E+04	2.13E+04	1.57E+04	2.82E+04	**1.27E+04**
6.47E+05	1.90E+08	5.60E+07	1.50E+08	**1.48E+05**
1.31E+02	1.92E+02	1.56E+02	1.61E+02	**1.23E+02**
1.32E+05	7.70E+07	3.40E+07	7.30E+07	**4.87E+04**
1.08E+02	1.15E+02	1.13E+02	1.10E+02	**1.06E+02**
2.05E+03	5.60E+07	1.70E+07	8.95E+05	5.27E+03
8.25E+02	3.05E+03	2.87E+03	3.54E+03	1.64E+03
1.18E+02	1.17E+02	1.17E+02	1.21E+02	**1.16E+02**
4.64E+02	4.36E+02	**4.13E+02**	4.61E+02	4.34E+02
1.09E+05	1.39E+05	1.12E+05	**1.09E+05**	1.10E+05
0.00E+00	5.67E+04	4.27E+01	4.15E-01	4.00E-01

It was found that the jDE algorithm performed the best among all the self-adaptive variants where it outperformed DE in all settings except 50D. SDE came second in the comparison with DESACR coming third and the worst being DESACRF. Across all dimensions, it was found that a self-adaptive parameter adaptation scheme was more beneficial to the DE algorithm compared to a fixed parameter setting scheme, particularly for larger number of optimization dimensions.

For future work, it would be interesting to extend the crossover-first approach to multi-objective DE algorithms as well as to test the XDE and XjDE algorithms in solving real-world practical problems such as in evolutionary robotics and evolutionary game-playing applications.

ACKNOWLEDGMENT

This work was supported in part by the ERGS research project ref: ERG0043 & FRG0435 granted by the Ministry of Education, Malaysia.

REFERENCES

[1] J. Teo, M.H.A. Hijazi, H.K. Lau, S. Fattah and A. Baharum, “Crossover-First Differential Evolution for Improved Global Optimization in Non-Uniform Search Landscapes,” Progress in Artificial Intelligence, 2015, 3(3-4): 129-134.

[2] R. Storn, K. Price, “Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces”, Technical Report TR-95-012, 1995, International Computer Science Institute, Berkeley.

[3] U. Chakraborty. Advances in Differential Evolution, 2008, Springer.

[4] S. Das, P.N. Suganthan. Differential Evolution: A Survey of the State-of-the-art. IEEE Transactions on Evolutionary Computation, 2011, 15(1):831–836.

[5] J. Brest, V. Zumer, M. Maucec, “Self-adaptive differential evolution algorithm in constrained real-parameter optimization”. IEEE Congress on Evolutionary Computation, 2006, pp. 215-222, IEEE.

[6] J. Brest, B. Boskovic, S. Greine, V. Zumer, M. Maucec, “Performance comparison of self-adaptive and adaptive differential evolution algorithms”, 2007, 11(7): 617-629.

[7] A. Zamuda, J. Brest, B. Boskovic, V. Zumer, "Large Scale Global Optimization using Differential Evolution with Self-adaptation and Cooperative Co-evolution," IEEE World Congress on Computational Intelligence, 2008, pp. 3718-3725.

[8] J. Teo. “Exploring dynamic self-adaptive populations in differential evolution”. Soft Computing, 2006, 10(8):673-686.

[9] J.J. Liang, B.Y. Qu, P.N. Suganthan, Q. Chen. “Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization”. Technical Report 201411A, 2014, Zhengzhou University, China.

[10] S.M. Guo, J. Tsai, C.C. Yang, P.H. Hsu. "A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set." In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 1003-1010. IEEE, 2015.

[11] C. Yu, L.C. Kelley, and Y. Tan. "Dynamic search fireworks algorithm with covariance mutation for solving the CEC 2015 learning based competition problems." In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 1106-1112. IEEE, 2015.

[12] K.M. Sallam, R.A. Sarker, D.L. Essam, S.M. Elsayed. "Neurodynamic differential evolution algorithm and solving CEC2015 competition problems." In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 1033-1040. IEEE, 2015.

[13] M.R. Tanweer, S. Suresh, N. Sundararajan. "Improved SRPSO algorithm for solving CEC 2015 computationally expensive numerical optimization problems." In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 1943-1949. IEEE, 2015.

[14] J.L. Rueda, I. Erlich. "Testing MVMO on learning-based real-parameter single objective benchmark optimization problems." In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 1025-1032. IEEE, 2015.

[15] N. Awad, M.Z. Ali, R.G. Reynolds. "A differential evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization." In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 1098-1105. IEEE, 2015.

[16] Y. Zhang, M. Zhou, Z. Jiang, and J. Liu. "A multi-agent genetic algorithm for big optimization problems." In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 703-707. IEEE, 2015.

[17] M. Omran, A. Salman, A. Engelbrecht. “Self-adaptive Differential Evolution”. In Computational Intelligence and Security, Hao, Y. et al. (eds.), 2005, 192-199, Springer.

[18] Dhanalakshmy, D.M., Pranav, P., Jeyakumar, G. “A Survey On Adaptation Strategies For Mutation And Crossover Rates Of Differential Evolution Algorithm”. International Journal on Advanced Science, Engineering and Information Technology, 2016, 6 (5), pp. 613-623.

[19] Ong, J.H., Teo, J. “A Time-Critical Investigation Of Parameter Tuning In Differential Evolution For Non-Linear Global Optimization”. International Journal on Advanced Science, Engineering and Information Technology, 2016, 6 (4), pp. 426-436.