Applied Mathematical Sciences, Vol. 10, 2016, no. 32, 1603 - 1610
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ams.2016.6377

Fixed vs. Self-Adaptive Crossover-First Differential Evolution

Jason Teo*, Asni Tahir, Norhayati Daut, Nordaliela Mohd. Rusli and Norazlina Khamis

Faculty of Computing and Informatics, Universiti Malaysia Sabah
Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia
* Corresponding author

Abstract

Although the Differential Evolution (DE) algorithm is a powerful and commonly used stochastic evolutionary-based optimizer for solving non-linear, continuous optimization problems, it has a highly unconventional order of genetic operations when compared against canonical evolutionary-based optimizers whereby in DE, mutation is conducted first before crossover. This has led us to investigate both a fixed as well as self-adaptive crossover-first version of DE, of which the fixed version has yielded statistically significant improvements to its performance when solving two particular classes of continuous optimization problems. The self-adaptive version of this crossover-first DE was also observed to be producing optimization results which were superior than the conventional DE.

Mathematics Subject Classification: 90C26

Keywords: Evolutionary Optimization Algorithms; Differential Evolution; Global Non-Linear Optimization; Self-Adaptation; CEC 2005 Benchmark

1 Introduction

Evolutionary-based optimization algorithms (EAs) are stochastic optimization heuristics that implement an artificial Darwinian evolution process which iteratively improves upon candidate solutions through pseudo-biological processes

of genetic crossover, mutation and selection. Among these algorithms, the Differential Evolution (DE) algorithm [2] stands out as one of the most competitive and widely-adopted EAs [1] [1].

In mimicking the natural selection process, the vast majority of canonical EAs typically diversifies their genetic material through firstly a crossover operation, which is analogous to the biological crossing-over of chromosomes during meiosis, and secondly subjecting the crossovered chromosomes to sporadic mutational processes. In DE however, the mutation is conducted first before crossover making it highly unusual when compared against other EAs. This has prompted us to investigate whether reversing the order of genetic operations in DE could create beneficial effects to the DE algorithm in terms of its optimization performance. Specifically, we hypothesize that conducting crossover first before mutation in Differential Evolution can improve its performance in solving some classes of continuous optimization problems. Our novel modification is a very straightforward reversal operation which does not add any new genetic operators, specialized functions or sub-routines nor computational complexity to the DE algorithm. In this study, it is shown that this straightforward change to DE is improves its optimization accuracy especially for in the case of two particular classes that possess comparatively more complex search spaces that are significantly more irregular and highly non-uniform. The most improvement over the conventional DE were observed in the results which were obtained when attempting to solve the most difficult classes of benchmark test functions, i.e. those test functions which had expanded functions and hybrid composition functions (F13-F25) [3]. Non-parametric tests will show that these improvements were statistically significant.

The report from this study reads as follows. In the second section, explanation regarding the original and most basic version of DE is given. The third section presents a short summary of our simple approach to the implementation of our fixed version of Crossover-First Differential Evolution algorithm [4] which we denote as XDE-FM as well as the self-adaptive version which we denote as XDE-SM. The fourth section shows the experimental results that compares our proposed novel XDE algorithms to the conventional DE. The last section concludes with the major findings and future avenues of possible exploration.

2 Conventional Differential Evolution

The original and conventional DE represents a stochastic multiple solution-based optimization algorithm that performs best in the real number domain.

[1] A search conducted on Scopus database for articles with title/abstract/keywords "Differential Evolution" within the physical sciences category only for the period of Jan-Dec 2015 returned 2956 results.

DE performs its operations in the following order: 1. initialization of chromosomes; 2. mutation operation; 3. crossover operation; 4. selection operation; 5. repeat from step (2) until termination. The DE algorithm requires a number of user parameters to be manually determined and set prior to conducting the optimization process. They are 1. NP: the population size; 2. F: the scaling factor; and 3. CR: the crossover rate. For a more detailed exposition on this matter, the interested reader may read [2]. Given the following optimization problem:

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

where x_i is a chromosome with D number of variables, x^* is the global optimum and $\Omega \subseteq R^D$, the conventional DE algorithm will perform optimization on the chromosome $x = \{x_1, x_2, ..., x_D\}$, where $x_i^G = \{x_{i,1}^G, x_{i.2}^G, ..., x_{i,D}^G\}$ is the ith chromosome in the pool of available individuals of the Gth generation of the particular evolutionary run.

Of paramount importance, a new candidate chromosome is created as follows:

1. Mutation operation: for each individual x_i^G, a new chromosome is generated:

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

   where $r1, r2$, and $r3$ are chromosomes which are picked at random from $[1, NP]$ and $i \neq r1 \neq r2 \neq r3$.

2. Crossover operation: for each individual x_i^G, a trial chromosome is generated:

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

   where R_j is a uniform real number chosen at random $[0, 1]$ and j_{rand} is uniform integer number chosen at random $[1, D]$.

3. Selection operation: the child chromosome is compared with the parent chromosome for survival to the next optimization generation:

$$x_i^{G+1} = \begin{cases} u_i^{G+1} & \text{if } f(u_i^{G+1}) \leq f(x_i^G). \\ x_i^G & \text{otherwise.} \end{cases} \tag{4}$$

3 XDE: Crossover-First Differential Evolution

In this part of the report, we present the modifications made to our proposed DE algorithm which we call XDE as compared to the basic Differential Evolution algorithm. We test two different methods of implementing our proposed algorithm: in subsection 3.1, we describe the version which implements a fixed mutation scheme whereas in subsection 3.2, we describe the version which implements a self-adaptive mutation scheme.

3.1 XDE-FM: XDE with Fixed Mutation

This section presents the explanation on how we modify the conventional DE algorithm. It is worth noting that our novel approach does not introduce any new genetic operators, specialized sub-routines nor diversity-enhancing sub-routines. The only difference in this proposed approach is simply reversing the genetic operations between the existing mutation operation and the crossover operation currently present in the original DE during the generation of the trial chromosome.

In XDE-FM, a new chromosome is generated as follows:

1. Crossover operation: for each individual x_i^G, a new chromosome is generated:

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

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

2. Mutation operation: for each individual x_i^G, a trial chromosome is generated:

$$u_{i,j}^{G+1} = \begin{cases} x_{R2,j}^G + F \cdot (x_{R3,j}^G - x_{R4,j}^G) & \text{if } R_j = 1. \\ v_{i,j}^{G+1} & \text{otherwise.} \end{cases} \tag{6}$$

   where $r2, r3$, and $r4$ are individuals chosen at random from $[1, NP]$, R_j is random Boolean value and $i \neq r1 \neq r2 \neq r3 \neq r4$.

3. Selection operation: the child chromosome is compared with the parent chromosome for survival to the next optimization generation:

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

For our novel implementation, the new XDE algorithm performs more normally as an evolutionary algorithm. This is the case because it first conducts crossing-over of the target chromosome with a randomly chosen chromosome from the pool of candidate solutions as in Eq. 5. After that, only then does it take this newly generated individual and conducts the mutation operation on it by utilizing the normal DE mutation routine (this is described in Section 3 by Eq. 3). Here, the difference between two chromosomes chosen at random, which is first scaled according to the parameter F, is then added to a third chromosome, also chosen at random. This is as presented in Eq. 6 where a new chromosome is generated. Moreover, there is a 50-50 chance for a certain gene present in the chromosome to mutate or not. No changes were made to the survivor selection operation.

3.2 XDE-SM: XDE with Self-Adaptive Mutation

Here we explain the second approach to our Crossover-First Differential Evolution algorithm which is now implemented to self-adapt the mutation rate rather than having it fixed as in XDE-FM. The only difference is in Eq. 9 where a new mutation rate is generated anew before it is used in the actual mutation of the decision variables of the trial chromosome.

In XDE-SM, a new chromosome is generated as follows:

1. Crossover operation: for each individual x_i^G, a new chromosome is generated:

$$v_{i,j}^{G+1} = \begin{cases} x_{r1,j}^G & \text{if } R_j \leq CR \text{ or } j = j_{rand}. \\ x_{i,j}^G & \text{otherwise.} \end{cases} \tag{8}$$

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

2. Mutation operation: for each individual x_i^G, a new chromosome is generated:

   Perform self-adaptation of the mutation rate (MR) using the standard DE mutation scheme:

$$MR_{i,j}^{G+1} = MR_{R2,j}^G + F \cdot (MR_{R3,j}^G - MR_{R4,j}^G) \tag{9}$$

$$u_{i,j}^{G+1} = \begin{cases} x_{R2,j}^{G} + F \cdot (x_{R3,j}^{G} - x_{R4,j}^{G}) & \text{if } R_j \leq MR_{i,j}^{G+1}. \\ v_{i,j}^{G+1} & \text{otherwise.} \end{cases} \tag{10}$$

where $r2, r3$, and $r4$ are individuals chosen at random from $[1, NP]$, R_j is a uniform real number chosen at random $[0, 1]$ and $i \neq r1 \neq r2 \neq r3 \neq r4$. MR denotes the mutation rate that is being self-adapted.

3. Selection operation: the child chromosome is compared with the parent chromosome for survival to the next optimization generation:

$$x_i^{G+1} = \begin{cases} u_i^{G+1}, & \text{if } f(u_i^{G+1}) \leq f(x_i^{G}). \\ x_i^{G}, & \text{otherwise.} \end{cases} \tag{11}$$

In XDE-SM, the motivation for investigating a self-adaptive approach is to answer the question of whether implementing the self-adaptation method would be beneficial for self-adapting the mutation rate in XDE-SM. In Eq. 9, a new mutation rate is generated for each trial vector before the trial vector is generated. This is so that if the newly generated mutation rate is used to mutate the values of the decision variables and is able to generate a fitter trial vector, then the newly generated mutation rate would be kept along with the values of the decision variables in the trial vector during the survivor selection process. Thus, it self-adapts and optimizes the mutation rate automatically as part of the evolving chromosome in XDE-SM. The method for generating a new mutation rate in the self-adaptive scheme employs the same DE scheme for mutating the values of the decision variables in the trial vector. In other words, the mutation rate in XDE-SM is mutated first before the actual mutation of the trial vector is conducted since the mutation of the trial vector requires the new mutation rate to be available first before this process can be carried out, as described in Eq. 10. The mutation rate is randomly initialized uniformly in the range [0.5,0.9] during the initialization stage. The self-adapting values of the mutation rates are also constrained to be within the range of 0.5 to 0.9 during the self-adaptation process should newly generated values exceed this range.

4 Methods

To conduct comparisons of XDE-FM and XDE-SM against the original DE algorithm, we have chosen to use the 25 test functions described in the CEC2005 global optimization competition [3]. In this benchmark suite of test problems, there are 25 minimization problems. These problems contain very different and diverse function properties. There are simple unimodal functions followed

by straightforward multimodal functions. There are also more complicated functions such as the expanded functions and the very complex, non-uniform hybrid composition functions.

5 Results and Discussion

Table 1: Average Best Solutions Over 50 Optimization Runs

Function	XDE-FM	XDE-SM	DE
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**
F3	3.40E+05	4.07E+02	**0.00E+00**
F4	**0.00E+00**	**0.00E+00**	**0.00E+00**
F5	6.07E+01	2.65E+00	**1.10E-04**
F6	7.47E+00	4.37E+00	**2.40E-01**
F7	**3.98E-02**	2.05E-01	3.46E-01
F8	2.04E+01	2.04E+01	**2.03E+01**
F9	**0.00E+00**	**0.00E+00**	1.88E+01
F10	**1.60E+01**	2.34E+01	2.64E+01
F11	8.98E+00	**8.71E+00**	8.82E+00
F12	**3.76E+02**	7.83E+02	9.47E+03
F13	**6.69E-01**	1.31E+00	2.02E+00
F14	**3.53E+00**	3.78E+00	3.61E+00
F15	**2.55E+02**	3.04E+02	3.51E+02
F16	**1.10E+02**	1.44E+02	1.48E+02
F17	**1.33E+02**	1.59E+02	1.64E+02
F18	**6.36E+02**	7.76E+02	8.26E+02
F19	**6.31E+02**	7.64E+02	8.26E+02
F20	**6.25E+02**	7.49E+02	8.28E+02
F21	**6.42E+02**	9.74E+02	1.02E+03
F22	7.72E+02	7.36E+02	**6.07E+02**
F23	**7.27E+02**	9.40E+02	1.09E+03
F24	4.09E+02	**4.08E+02**	**4.08E+02**
F25	4.09E+02	**4.08E+02**	**4.08E+02**

The results obtained from 50 independent repeated runs of each algorithm over the 25 benchmark continuous test functions are shown in Table 1. XDE-FM with 17 wins showed the most number of best outcomes when tested using these 25 test functions. Next, the conventional DE with 10 wins was ranked second. Finally, XDE-SM had 7 wins and was ranked third. The results from this study show that the novel crossover-first DE is useful for improving the accuracy of DE's optimization. In particular, XDE-FM performed better than the original DE in both of the expanded functions (F13 & F14) and in 8 out of the 11 hybrid composition functions (F15-F21 & F23). These two classes of test problems represent the most challenging functions to solve in the CEC2005 test suite. These are the test functions with very irregular, non-uniform search landscapes. Also, in general, the fixed mutation approach performed better than the self-adaptive mutation approach in the proposed crossover-first DE algorithm. Nonetheless, XDE-SM still performed better than the basic DE in 14 out of the 25 test problems and only worse in 6 with 5 ties. As such, even though the fixed mutation outperformed the self-adaptive mutation, the

crossover-first scheme was still able to improve upon the basic DE in both implementations of the novel DE algorithm.

6 Conclusion

A novel approach for conducting genetic operations in the Differential Evolution (DE) algorithm was proposed in this study where crossover precedes mutation, thereby making DE more conventional as an evolutionary algorithm compared to its original version. Without the introduction of any additional specialized methods and/or functions for processing historical performance, population resizing, strategy aggregation, surrogate models and/or new diversity-generating operators, we have maintained the simplicity factor within the proposed novel crossover-first DE. Testing over 25 continouous optimization problems from the CEC2005 suite of benchmark tests has shown that this simple and straightforward change in the order of operations between crossover and mutation was able to enhance the optimization accuracy of DE.

References

[1] S. Das and P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation*, **15** (2011), no. 1, 4-31. http://dx.doi.org/10.1109/tevc.2010.2059031

[2] K.V. Price, R.M. Storn and J.A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. http://dx.doi.org/10.1007/3-540-31306-0

[3] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. Chen, A. Auger and S. Tiwari, *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*, Technical Report, Nanyang Technological University, Singapore, May 2005 and KanGAL Report 2005005, IIT Kanpur, India, 2005.

[4] 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*, **3** (2015), no. 3, 129-134. http://dx.doi.org/10.1007/s13748-015-0061-1

Received: March 17, 2016; Published: April 30, 2016