Multi-population Differential Evolution with Adaptive Parameter Control for Global Optimization

Wei-jie Yu and Jun Zhang (Corresponding Author)
Department of Computer Science, Sun Yat-sen University
Key Laboratory of Digital Life, Ministry of Education
Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P.R. China
junzhang@ieee.org

ABSTRACT
Differential evolution (DE) is one of the most successful evolutionary algorithms (EAs) for global numerical optimization. Like other EAs, maintaining population diversity is important for DE to escape from local optima and locate a near-global optimum. Using a multi-population algorithm is a representative method to avoid early loss of population diversity. In this paper, we propose a multi-population DE algorithm (MPDE) which manipulates multiple sub-populations. Different sub-populations in MPDE exchange information via a novel mutation operation instead of migration used in most multi-population EAs. The mutation operation is helpful to balance the fast convergence and population diversity of the proposed algorithm. Moreover, the performance of MPDE is further improved by an adaptive parameter control scheme designed based on the multi-population approach. Each sub-population in MPDE evolves with its own set of control parameters, and a learning strategy is used to adaptively adjust the parameter values. A set of benchmark functions is used to test the proposed MPDE algorithm. The experimental results show that MPDE performs better than, or at least comparably, to the classical single population DE with fixed parameter values and three existing state-of-the-art DE variants.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods*

General Terms
Algorithms

Keywords
Differential evolution, multi-population, adaptive parameter control, global optimization

1. INTRODUCTION
Differential evolution (DE), first proposed by Storn and Price [1][2], is a simple and efficient global optimizer over continuous spaces. It has been successfully applied to a variety of practical problems from diverse domains [3]. However, like other evolutionary algorithms (EAs), the DE algorithm suffers from the problem of premature convergence, i.e., the population loses diversity too early and is trapped in local optima of the objective function. This challenge is particularly hard when the dimension of problems is high and there are a lot of local optima.

In this paper, we propose a multi-population approach for the DE variant known as DE/best/1 which uses the best solution information to guide the search. The DE/best/1 strategy has a fast convergence rate but easily suffers from premature convergence due to early loss of population diversity. Our proposed multi-population approach is helpful for diversifying the population and alleviating the problem of premature convergence. Specifically, in our proposed multi-population DE (MPDE), the entire population is divided into multiple sub-populations which evolve on their own. The size and number of the sub-populations are predefined and kept unchanged after initialization. During the evolutionary process, each sub-population can exchange information with any other sub-populations. Most of the multi-population EAs use migration as a means of communication between different sub-populations. However, the performance of these algorithms is sensitive to the choice of control parameters such as migration size and rate. Instead of using migration, sub-populations in MPDE communicate with each other by means of a novel mutation operation, which involves a best vector and a difference vector. The best vector is selected from the corresponding sub-population instead of the entire population, which can balance the fast convergence and population diversity. On the other hand, the difference vector is generated by two vectors selected from the entire population. Therefore, the difference vector may contain information from two different sub-populations and can be used as a medium of information exchange.

In order to further improve the robustness and efficiency of the proposed algorithm, a learning strategy is designed to adaptively control the DE parameters. Since the MPDE manipulates several sub-populations, each sub-population is associated with its own control parameters F and CR. During the evolutionary process, the sub-population with good parameter values is more likely to generate more promising solutions. In each generation, the parameter values in the poor sub-populations are optimized by learning from the parameter values in the good sub-populations. Therefore, the MPDE can evolve the parameters for each sub-population to appropriate values for different evolutionary stages and different optimization problems.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

The rest of this paper is organized as follows. Section 2 reviews the DE algorithm and the related works on multi-population approaches and parameter control methods for DE. Section 3 describes the proposed MPDE algorithm in detail. In Section 4, the MPDE is compared with the classical DE and other existing state-of-the-art DEs on a suite of benchmark problems. Finally, Section 5 draws the conclusions.

2. DE ALGORITHM AND RELATED WORKS

2.1 Differential evolution (DE) algorithm

DE is a population-based stochastic algorithm designed for global numerical optimization. Similar to other EAs, DE searches for a global optimum in the feasible solution space with a population of parameter vectors $\{\mathrm{x}_i^g = [x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g], i = 1,2,..., NP\}$, where g denotes the current generation, D is the dimension of the search space, and NP is the population size. In generation g=0, the jth component of the ith vector can be initialized as

$$x_{i,j}^0 = x_{\min,j} + \mathrm{rand}(0,1) \cdot (x_{\max,j} - x_{\min,j}) \tag{1}$$

where rand(0,1) is a uniform random number on the interval [0,1], and $x_{\min,j}$, $x_{\max,j}$ are the prescribed minimum and maximum bounds of the jth dimension, respectively. After initialization, DE enters an evolutionary process which includes mutation, crossover, and selection operations.

Mutation: In each generation g, the mutation operation is applied to each individual x_i^g (also called target vector) to create its corresponding mutant vector v_i^g. The five most frequently used mutation strategies are listed as follows.

- DE/rand/1:

$$\mathrm{v}_i^g = \mathrm{x}_{r1}^g + F \cdot (\mathrm{x}_{r2}^g - \mathrm{x}_{r3}^g) \tag{2}$$

- DE/target-to-best/1:

$$\mathrm{v}_i^g = \mathrm{x}_i^g + F \cdot (\mathrm{x}_{\mathrm{best}}^g - \mathrm{x}_i^g) + F \cdot (\mathrm{x}_{r1}^g - \mathrm{x}_{r2}^g) \tag{3}$$

- DE/best/1:

$$\mathrm{v}_i^g = \mathrm{x}_{\mathrm{best}}^g + F \cdot (\mathrm{x}_{r1}^g - \mathrm{x}_{r2}^g) \tag{4}$$

- DE/best/2:

$$\mathrm{v}_i^g = \mathrm{x}_{\mathrm{best}}^g + F \cdot (\mathrm{x}_{r1}^g - \mathrm{x}_{r2}^g) + F \cdot (\mathrm{x}_{r3}^g - \mathrm{x}_{r4}^g) \tag{5}$$

- DE/rand/2:

$$\mathrm{v}_i^g = \mathrm{x}_{r1}^g + F \cdot (\mathrm{x}_{r2}^g - \mathrm{x}_{r3}^g) + F \cdot (\mathrm{x}_{r4}^g - \mathrm{x}_{r5}^g) \tag{6}$$

It can be seen that the mutant vector v_i^g is generated by combing a base vector with one or two scaled difference vectors. In the above equations, the indices r_1, r_2, r_3, r_4, and r_5 are distinct integers randomly selected from the range [1, NP], and all are different from the index i. $\mathrm{x}_{\mathrm{best}}^g$ is the vector with the best fitness value in the current generation. The factor F is a positive control parameter for amplifying the difference vectors.

Crossover: In order to enhance population diversity, a crossover operation exchanges some components of the mutant vector v_i^g with the target vector x_i^g to generate a trial vector u_i^g. The process can be expressed as

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, \text{ if } \mathrm{rand}(0,1) \le CR \text{ or } j = j_{\mathrm{rand}} \\ x_{i,j}^g, \text{ otherwise} \end{cases} \tag{7}$$

where rand(0,1) is a uniformly distributed random number as before. j_{rand} is an integer randomly generated from the range [1, D], which is used to ensure the trial vector has at least one component different from the target vector. The crossover probability CR is another control parameter, which determines the fraction of vector components inherited from the mutant vector.

Selection: To decide whether the target or the trial vector can survive to the next generation, the selection operation is finally performed. For a minimization problem, the vector with the lower fitness value enters the next generation, which can be expressed as follows:

$$\mathrm{x}_i^{g+1} = \begin{cases} \mathrm{u}_i^g, \text{ if } f(\mathrm{u}_i^g) \le f(\mathrm{x}_i^g) \\ \mathrm{x}_i^g, \text{ otherwise} \end{cases} \tag{8}$$

where $f(\mathrm{x})$ is the objective function for the minimization problem.

2.2 Multi-population approaches for DE

Various multi-population approaches for DE have been designed to solve different kinds of optimization problems. Most of these approaches maintain population diversity via information exchange between different sub-populations. Tasoulis *et al*. [4] parallelized DE in a virtual parallel environment so as to improve performance. In order to promote information sharing, the best individuals from each sub-population are allowed to migrate to other sub-populations based on a ring topology. Another migration scheme for multi-population was proposed in [5]. The authors suggested substituting the oldest individual of the target sub-population instead of a randomly chosen one. In [6], a multi-populated DE (MDE) with a regrouping scheme was presented to solve constrained optimization problems. To provide full information exchange between sub-populations, the individual selection in the mutation phase is from all sub-populations.

Instead of exchanging information between different sub-populations, some multi-population approaches use other mechanisms to maintain population diversity. Mendes and Mohais [7] proposed a multi-population DE algorithm, called DynDE, to solve dynamic optimization problems. To maintain population diversity, DynDE reinitializes a sub-population if the distance between the best individuals of this population and that of another population is within a range. Brest *et al*. [8] investigated a self-adaptive DE (jDE) where a multi-population method with aging mechanism is used for dynamic optimization. In their algorithm, no information is exchanged between the sub-populations.

2.3 Parameter Control Methods for DE

Control parameters in DE have significant effects on the performance of the algorithm [9][10]. However, there is no fixed parameter setting that can achieve the best performance for all types of problems. Therefore, various parameter control methods have been proposed for DE to dynamically adjust the parameter values. These methods are capable of enhancing the robustness of the DE algorithm. According to the classification by Eiben *et al*. [11], parameter adaptation methods can be classified into three categories as follows.

1) *Deterministic parameter control*

This kind of method simply uses some deterministic rules to change the parameter values, without exploiting any information from the evolution. In [12], Das *et al*. proposed two schemes to control the scale factor F of DE. The first one decreases the value of F based on a linear rule, and the second one generates the value of F in a random way. Since the linear rule in the first scheme is based on the current number and the predefined maximum number of generations, it is actually determined before running the algorithm.

2) *Adaptive parameter control*

Strategies for adaptive parameter control dynamically adjust the parameter values by using some form of feedback from the search, which can adapt to different evolutionary states. In [13], a fuzzy logic control approach was proposed to adapt the DE parameters F and CR. The fuzzy controllers incorporate the function values and individuals of the successful generations as their inputs, and the outputs are the values of F and CR. In [14], the value of the parameter F is adaptively adjusted based on the minimum and maximum objective function values over the individuals in each generation.

3) *Self-adaptive parameter control*

For this method, each individual in the population maintains its own set of parameter values, which are encoded into the chromosome and optimized through the evolutionary process. Brest *et al*. [15] introduced a self-adaptive approach for the control parameters F and CR. Each individual in the population is associated with parameter values F_i and CR_i. In each generation, new values for F_i and CR_i are randomly generated in their corresponding ranges with probabilities τ_1 and τ_2, respectively. Qin *et al*. [16] proposed a self-adaptive DE (SaDE) algorithm, in which the trial vector generation strategies as well as the control parameters are self-adapted by learning from the previous experiences. Zhang and Sanderson [17] introduced a new adaptive DE called JADE. The control parameters for each individual in JADE are updated based on their historical record of success. In the PLADE recently proposed by Zhan and Zhang [18], a learning strategy inspired by particle swarm optimization (PSO) is used to adaptively adjust the DE parameters.

3. MULTI-POPULATION DIFFERENTIAL EVOLUTION (MPDE)

There exist several DE variants due to the various mutation strategies described in Section 2. For global numerical optimization, the experimental studies in [19] indicate that the DE/best/1 with binomial crossover is the most competitive approach among eight DE variants. In this section, we focus on the DE/best/1 algorithm and introduce a multi-population version of this algorithm.

3.1 Multi-population Approach

In the multi-population approach, the entire population is divided into a predefined number of sub-populations. The size and population members of these sub-populations are kept unchanged during the algorithm's execution. Each sub-population can exchange information with any other sub-populations.

In most of the multi-population evolutionary algorithms, migration is used as a means of communication between sub-population. Different from these algorithms, sub-populations in our multi-population approach exchange information via the mutation operation. As mentioned earlier, a mutant vector in DE/best/1 is generated by combining a best vector with a scaled difference vector. In the multi-population approach, the best vector is selected from the sub-population with respect to the target vector instead of the entire population. Therefore, each sub-population is attracted by its own best vector, and the entire population is indeed guided by several locally best vectors instead of the single globally best vector. Such an approach thus benefits from a balance between fast convergence and population diversity.

On the other hand, the difference vector involved in the mutation operation can be generated not only by two vectors from the same sub-population, but also by two vectors from different sub-populations (i.e. the entire population). Since the difference vector may contain information from two different sub-populations, such a mutation operation can be used as a means of information exchange.

3.2 Adaptive Parameter Control Method

In order to enhance the robustness of the proposed algorithm, we further designed an adaptive parameter control scheme based on the multi-population approach. During the evolutionary process, each sub-population maintains its own set of control parameters, and a learning strategy is used to optimize these parameters for each sub-population.

The learning strategy for adaptive parameter control in this paper is inspired by the PSO learning strategy proposed by Zhan and Zhang [18]. Each sub-population s is associated with its own control parameters F_s and CR_s, $s = 1, 2, ..., NS$, where NS is the number of sub-populations. After all sub-populations have finished their evolutionary operations in the current generation, the one which has generated the most number of successful vectors is chosen as the best sub-population. The successful vectors are defined as the vectors which have their solution quality improved under the control of their own control parameters. Then, the parameter values F_s and CR_s in the remaining sub-populations are adjusted by approaching those in the best sub-population.

Table 1. Benchmark functions used in this paper

Name	Test function	*D*	*S*	f_{min}
Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^2$	30	$[-100, 100]^D$	0
Rosenbrock	$f_2(x)=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	$[-30, 30]^D$	0
Schwefel	$f_3(x)=\sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]^D$	-12569.5
Rastrigin	$f_4(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	30	$[-5.12, 5.12]^D$	0
Ackley	$f_5(x)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i)+20+e$	30	$[-32, 32]^D$	0
Griewank	$f_6(x)=\frac{1}{4000}\sum_{i=1}^{D} x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-600, 600]^D$	0
Penalized	$f_7(x)=\frac{\pi}{D}\{10\sin^2(\pi y_i)+\sum_{i-1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_D-1)^2\}+\sum_{i=1}^{D} u(x_i,10,100,4)$, $y_i=1+\frac{1}{4}(x_i+1)$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a,\\ 0, & -a\le x_i<a,\\ k(-x_i-a)^m, & x_i<-a.\end{cases}$	30	$[-50, 50]^D$	0
Penalized	$f_8(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin 2(3\pi x_{i+1})]+(x_D-1)^2[1+\sin^2(2\pi x_D)]\}+\sum_{i=1}^{D} u(x_i,5,100,4)$	30	$[-50, 50]^D$	0

4. EXPERIMENTAL STUDIES

In this section, the performance of our proposed MPDE algorithm is evaluated on a set of benchmark functions listed in Table 1 [20]. All these functions are high-dimensional problems with dimensions D=30. Function f_1 is unimodal. Function f_2 is the Rosenbrock function which is unimodal for $D \le 3$, but may become multimodal when the dimension is high [21]. Functions f_3-f_8 are multimodal where the number of local minima increases exponentially with the problem dimension. Such functions appear to the most difficult class of problems for many optimization algorithms.

Two groups of experiments are carried out. The first group compares the MPDE with the classical DEs. Since MPDE is based on the DE/best/1 strategy, three classical DE/best/1 algorithms with different parameter settings are used. All the classical DE/best/1 set F as 0.5 as suggested in most of the literatures [2][10], and set CR as 0.1, 0.5, and 0.9 respectively. In the second groups of experiments, we further compare the MPDE with three existing state-of-the-art DE variants, i.e., JADE [17], jDE [15], and PLADE [18]. We use the parameter settings of JADE, jDE, and PLADE according to their original papers.

For a fair comparison, all DE algorithms use a population size of 100, and our proposed MPDE divides the entire population into 10 sub-populations. Moreover, each algorithm is run 50 times independently and the results are averaged. For clarity, the results of the best algorithms are marked in **boldface**.

4.1 Comparison with Classical DE

Table 2 summarizes the experimental results of the classical DE/best/1 algorithms and the proposed MPDE algorithm. MPDE performs significantly better than the other three classical DE/best/1 on 7 (f_2-f_8) out of the 8 benchmark functions. In addition, it can be observed that the performance of classical DE/best/1 is very sensitive to the parameter settings. The DE/best/1 (CR = 0.1) is able to find the near-global optimum on most of the multimodal functions (f_3-f_8), but it performs the worst on the unimodal functions f_1 and f_2. In contrast, the DE/best/1 (CR = 0.9) obtains the best performance on f_1, but it fails to locate the near-global optimum on all the multimodal functions except f_6. The performance of MPDE is less dependent on the optimization problems. It is capable of obtaining the near-global optimum on both unimodal and multimodal functions. This is because the MPDE can maintain population diversity to avoid premature convergence, and its adaptive parameter control strategy is helpful to improve robustness.

Table 2. Comparison between MPDE and the classical DEs

Fun.	Gen.	MPDE Mean (Std Dev)	DE (*CR*=0.1) Mean (Std Dev)	DE (*CR*=0.5) Mean (Std Dev)	DE (*CR*=0.9) Mean (Std Dev)
f_1	1500	8.62E-43 (2.82E-42)	3.63E-34 (2.85E-34)	4.44E-100 (1.20E–99)	**2.41E-174 (0.00E+00)**
f_2	20000	**3.76E-30 (8.43E-30)**	3.08E+01 (2.06E+01)	1.04E+00 (1.75E+00)	1.51E+00 (1.94E+00)
f_3	9000	**-12567.12 (1.66E+01)**	-12232.31 (2.06E+02)	-10307.72 (4.35E+02)	-8316.36 (7.07E+02)
f_4	5000	**0.00E+00 (0.00E+00)**	2.79E-01 (4.47E-01)	2.49E+01 (7.14E+00)	7.16E+01 (1.86E+01)
f_5	2000	**4.14E-15 (0.00E+00)**	7.77E-15 (8.67E-16)	1.76E-01 (4.05E-01)	4.72E+00 (1.38E+00)
f_6	3000	**0.00E+00 (0.00E+00)**	2.96E-04 (1.45E-03)	5.42E-03 (7.99E-03)	2.55E-02 (2.25E-02)
f_7	1500	**1.57E-32 (1.64E-47)**	**1.57E-32 (1.64E-47)**	7.28E-02 (2.01E-01)	1.93E+00 (2.50E+00)
f_8	1500	**1.35E-32 (8.21E-48)**	**1.35E-32 (8.21E-48)**	1.11E-01 (4.87E-01)	2.67E+00 (4.03E+00)

4.2 Comparison with State-of-the-art DE

The performance of MPDE is further compared with three other state-of-the-art DEs, namely, JADE [17], jDE [15], and PLADE [18]. The experimental results averaged over 50 runs are listed in Table 3. All these compared algorithms are able to locate a near-global optimum on most of the benchmark functions, since they all use some strategies to enhance the algorithms' robustness. However, the JADE, jDE, and PLADE are all trapped in local optima sometimes on the Rosenbrock function f_2, while our proposed MPDE can obtain the near-global optimum in every run. For the JADE, it performs the best on the Sphere function f_1, but it is sometimes trapped by the Rosenbrock function f_2 and the Griewank function f_6. The jDE and PLADE do the best on the Schwefel function f_3, followed by our proposed MPDE. For the other functions, MPDE has the capability to obtain near-global optimum with higher mean solution accuracy than the other compared algorithms. Overall, our proposed MPDE algorithm achieves the best performance on 6 (f_2, f_4-f_8) out of the 8 benchmark functions when compared with the other three improved DE variants.

5. CONCLUSION

In this paper, a multi-population DE with adaptive parameter control, MPDE, has been developed. In MPDE, the entire population is divided into multiple sub-populations. A novel mutation operation is designed and used as a means of communication between different sub-populations. Due to the multi-population approach, the diversity of the overall population can be preserved. In addition, each sub-population is associated with its own control parameters F and CR. During the evolutionary process, poor sub-populations learn the parameter values from the best sub-population in the current generation. Thus, the control parameters for each sub-population can be adjusted to better values, and the DE evolutionary operations become more effective and efficient. The performance of the MPDE algorithm has been validated over a set of 8 benchmark functions. Experimental results show that the multi-population approach is helpful to maintain population diversity. MPDE not only performs better than the classical DEs with single population and fixed parameter values, but also is competitive when compared with other state-of-the-art DEs.

Table 3. Comparison between MPDE and other state-of-the-art DEs

Fun.	Gen.	MPDE Mean (Std Dev)	JADE Mean (Std Dev)	jDE Mean (Std Dev)	PLADE Mean (Std Dev)
f_1	1500	8.62E-43 (2.82E-42)	**2.09E-58 (1.46E-57)**	3.05E-31 (2.83E–31)	1.75E-29 (5.09E-29)
f_2	20000	**3.76E-30 (8.43E-30)**	7.97E-01 (1.61E+00)	2.39E-01 (9.47E-01)	7.97E-02 (5.58E-01)
f_3	9000	-12567.12 (1.66E+01)	-12498.40 (1.20E+02)	**-12569.49 (7.28E-12)**	**-12569.49 (7.28E-12)**
f_4	5000	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_5	2000	**4.14E-15 (0.00E+00)**	6.34E-15 (1.74E-15)	4.28E-15 (6.96E-16)	4.71E-15 (1.30E-15)
f_6	3000	**0.00E+00 (0.00E+00)**	1.53E-03 (3.37E-03)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_7	1500	**1.57E-32 (1.64E-47)**	7.60E-30 (5.36E-29)	3.93E-32 (4.62E-32)	3.06E-30 (1.05E-29)
f_8	1500	**1.35E-32 (8.21E-48)**	**1.35E-32 (8.29E-48)**	2.84E-31 (6.08E-31)	4.69E-30 (9.11E-30)

6. ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) No. 61070004, and by NSFC Joint Fund with Guangdong under Key Project U0835002. The authors are with the Key Lab of Digital Life, MoE, China, and the Key Lab. of Softw. Tech, Edu. Depart of Guangdong Province. The corresponding author is Jun Zhang, email: junzhang@ieee.org.

7. REFERENCES

[1] R. M. Storn and K. V. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," TR-95-012, 1995 [Online]. Available: http://icsi.berkeley.edu/~storn/litera.html.

[2] R. M. Storn and K. V. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, pp. 341–359, 1997.

[3] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.

[4] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, “Parallel differential evolution,” in *Proc. Congr. Evol. Comput.*, 2004.

[5] K. N. Kozlov and A. M. Samsonov, “A new migration scheme for parallel differential evolution”, in *Proc. Fifth International Conference on Bioinformatics of Genome Regulation and Structure (BGRS 06)*, Novosibirsk, Russia, July 16-22, 2006.

[6] M. F. Tasgetiren and P. N. Suganthan, “A multi-populated differential evolution algorithm for solving constrained optimization problem,” in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2006, pp. 340–354.

[7] R. Mendes and A. S. Mohais, “DynDE: A differential evolution for dynamic optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, 2005, pp. 2808–2815.

[8] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, “Dynamic optimization using self-adaptive differential evolution,” in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 415–422.

[9] R. Gämperle, S. D. Müller, and P. Koumoutsakos, “A parameter study for differential evolution,” in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, A. Grmela and N. E. Mastorakis, Eds. Interlaken, Switzerland: WSEAS Press, 2002, pp. 293–298.

[10] J. Rönkkönen, S. Kukkonen, and K. V. Price, “Real-parameter optimization with differential evolution,” in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 506–513.

[11] A. E. Eiben, R. Hinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Jul. 1999.

[12] S. Das, A. Konar, and U. K. Chakraborty, “Two improved differential evolution schemes for faster global search,” in *Proc. Genetic Evol. Comput. Conf.*, 2005, pp. 991-998.

[13] J. Liu and J. Lampinen, “A fuzzy adaptive differential evolution algorithm,” *Soft Comput. – A Fusion of Foundation, Methodologies and Applications*, vol. 9, no. 6, pp. 448–462, Apr. 2005.

[14] M. M. Ali and A. Törn, “Population set based global optimization algorithms: Some modifications and numerical studies,” *Comput. Oper. Res.*, vol. 31, no. 10, pp. 1703–1725, 2004.

[15] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[17] J. Q. Zhang and A. C. Sanderson, “JADE: Adaptive differential evolution with optional external archive,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[18] Z.-H. Zhan and J. Zhang, “Self-adaptive differential evolution based on PSO learning strategy,” in *Proc. Genet. Evol. Comput. Conf.*, 2010, pp. 39-47.

[19] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, “A comparative study of differential evolution variants for global optimization,” in *Proc. Genet. Evol. Comput. Conf.*, 2006, pp. 485–492.

[20] X. Yao, Y. Liu, and G. M. Lin, “Evolutionary programming made faster,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82-102, Jul. 1999.

[21] Y.-W. Shang and Y.-H. Qiu, “A note on the extended rosenbrock function,” *Evol. Comput.*, vol. 14, no. 1, pp. 119–126, 2006.