

An Improved Self-adaptive Control Parameter of Differential Evolution for Global Optimization

Liyuan Jia¹, Wenyin Gong², and Hongbin Wu¹

¹ Department of Computer Science,
Hunan City University, Yiyang Hunan 413000, P.R. China,
jia_4211003@126.com

² School of Computer Science,
China University of Geosciences, Wuhan 430074, P.R. China
cug11100304@yahoo.com.cn

Abstract. Differential evolution (DE), a fast and robust evolutionary algorithm for global optimization, has been widely used in many areas. However, the success of DE for solving different problems mainly depends on properly choosing the control parameter values. On the other hand, DE is good at exploring the search space and locating the region of global minimum, but it is slow at exploiting the solution. In order to alleviate these drawbacks of DE, this paper proposes an improved self-adaptive control parameter of DE, referred to as ISADE, for global numerical optimization. The proposed approach employs the individual fitness information to adapt the parameter settings. Hence, it can exploit the information of the individual and generate the promising offspring efficiently. To verify the viability of the proposed ISADE, 10 high-dimensional benchmark problems are chosen from literature. Experiment results indicate that this approach is efficient and effective. It is proved that this approach performs better than the original DE in terms of the convergence rate and the quality of the final solutions. Moreover, ISADE obtains faster convergence than the original self-adaptive control parameter of DE (SADE).

Keywords: Multiobjective optimization, Immune algorithm, Similar individuals, Evolutionary algorithm.

1 Introduction

Differential Evolution (DE) [1] is a simple yet powerful population-based, direct search algorithm with the generation-and-test feature for global optimization problems using real-valued parameters. DE uses the distance and direction information from the current population to guide the further search. It won the third place at the first International Contest on Evolutionary Computation on a real-valued function test-suite [2]. Among DE's advantages are its simple structure, ease of use, speed and robustness. Price and Storn [1] gave the working principle of DE with single scheme. Later on, they suggested ten different schemes of DE [2], [3]. However, DE has been shown to have certain weaknesses, especially if the global optimum should be located using a limited number of fitness function evaluations (NFFE). In addition, DE is good at exploring the search space and locating the region of global minimum, but it is slow at exploitation of the

solution [4]. Moreover, the parameters of DE are problem dependent and the choice of them is often critical for the performance of DE [5].

In order to remedy these drawbacks of DE, in this paper, we propose an improved self-adaptive control parameter of DE, referred to as ISADE, for global optimization. Our work is an extension of the self-adaptive control parameter of DE (SADE) [6]. First, the self-adaptive control parameter is adopted to alleviate to choose appropriately control parameter values for different problems. Second, our approach employs the individual fitness information to adapt the parameter settings. And hence, it can exploit the information of the individual and generate the promising offspring efficiently. To verify the viability of the proposed ISADE, 10 high-dimensional benchmark problems are chosen from literature. Experimental results indicate that our approach is efficient and effective. Our approach performs better than the original DE in terms of both the convergence rate and the quality of the final solutions. Moreover, ISADE obtains faster convergence rate than the original self-adaptive control parameter of DE (SADE).

The rest of this paper is organized as follows. Section 2 briefly introduces the DE algorithm and the related work in control parameter of DE. In Section 3 presents our proposed approach in detail. Followed by the experimental results and the analysis shown in Section 4. In the last section, Section 5, we denote the conclusion and our future work.

2 Differential Evolution

Consider the following optimization problem:

$$\text{Minimize } f(X), \quad X \in S \quad (1)$$

where $S \subseteq \mathbb{R}^D$ is a compact set, and D is the dimension of the decision variables.

The DE algorithm [1] is a simple EA that creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness. This is a rather greedy selection scheme that often outperforms the traditional EAs. In addition, DE is a simple yet powerful population-based, direct search algorithm with the generation-and-test feature for globally optimizing functions using real-valued parameters. Among DE's advantages are its simple structure, ease of use, speed and robustness. Due to these advantages, it has many real-world applications, such as data mining [7], [8], pattern recognition, digital filter design, neural network training, etc. [3], [9].

The DE algorithm in pseudo-code is shown in Algorithm 1. D is the number of decision variables, NP is the size of the parent population P ; F is the mutation scaling factor; CR is the probability of crossover operator; U^i is the offspring; $\text{rndint}(1, D)$ is a uniformly distributed random integer number between 1 and D ; $X_j^{r_1}$ is the j -th variable of solution X^{r_1} ; and $\text{rnd}_j[0, 1)$ is a uniformly distributed random real number in $[0, 1)$. Many schemes of creation of a candidate are possible. We use the DE/rand/1/bin scheme (see lines 6 - 13 of Algorithm 1) described in Algorithm 1 (more details on DE/rand/1/bin and other DE schemes can be found in [2] and [3]).

Algorithm 1. DE algorithm with DE/rand/1/bin

```

1: Generate the initial population  $P$ 
2: Evaluate the fitness for each individual in  $P$ 
3: while The halting criterion is not satisfied do
4:   for  $i = 1$  to  $NP$  do
5:     Select uniform randomly  $r_1 \neq r_2 \neq r_3 \neq i$ 
6:      $j_{rand} = \text{rndint}(1, D)$ 
7:     for  $j = 1$  to  $D$  do
8:       if  $\text{rnd}_j[0, 1] > CR$  or  $j == j_{rand}$  then
9:          $U_j^i = X_j^{r_1} + F \times (X_j^{r_2} - X_j^{r_3})$ 
10:        else
11:           $U_j^i = X_j^i$ 
12:        end if
13:      end for
14:      Evaluate the offspring  $U^i$ 
15:      if  $U^i$  is better than  $P^i$  then
16:         $P^i = U^i$ 
17:      end if
18:    end for
19: end while

```

From Algorithm 1, we can see that there are only three control parameters in this algorithm. These are NP , F and CR . As for the terminal conditions, one can either fix the maximum NFFEs Max_NFFEs or the precision of a desired solution VTR (value to reach).

Since the parameters of DE are problem dependent and the choice of them is often critical for the performance of DE [5], [10]. Adapting the DE's control parameters is one possible improvement. Liu and Lampinen [10] proposed a Fuzzy Adaptive DE (FADE), which employs fuzzy logic controllers to adapt the mutation and crossover control parameters. Brest *et al.* [6] proposed self-adapting control parameter settings. Their proposed approach encodes the F and CR parameters into the chromosome and uses a self-adaptive control mechanism to change them. Salman *et al.* [11] proposed a self-adaptive DE (SDE) algorithm that eliminates the need for manual tuning of control parameters. In SDE, the mutation weighting factor F is self-adapted by a mutation strategy similar to the mutation operator of DE. Nobakhti and Wang [12] proposed a Randomized Adaptive Differential Evolution (RADE) method, where a simple randomized self-adaptive scheme was proposed for the mutation weighting factor F . Das *et al.* [13] proposed two variants of DE, DERSF and DETVSF, that use varying scale factors. They concluded that those variants outperform the original DE. Teo [14] presented a dynamic self-adaptive populations DE, where the population size is self-adapting. Through five De Jong's test functions, they showed that DE with self-adaptive populations produced highly competitive results. Brest and Maučec [15] proposed an improved DE method, where the population size is gradually reduced. They concluded that their approach improved efficiency and robustness of DE.

3 Our Approach: ISADE

As mentioned above, there are many works for adapting the control parameters of DE. Despite its simplicity, the SADE method [6] seems to have good performance for a various set of test problems. However, the performance of this self-adaptive control parameter strategy proposed in [6] can be further improved to obtain faster convergence rate. Based on this consideration, in this work, we propose an improved self-adaptive control parameter method for the DE algorithm. Inspired by the ideas in [16], in our approach the fitness information of each individual is considered to adapt the CR and F values. Thus, it can exploit the information of the individual and generate the promising offspring efficiently. Our approach, referred to as ISADE, is presented in detail as follows.

3.1 Improved Self-adaptive Control Parameter

Similar to [6], the individual is represented by a D -dimensional vector in the following way:

$$X^i = (X_1^i, X_2^i, \dots, X_j^i, \dots, X_D^i, CR^i, F^i)^T \quad (2)$$

where X_j^i is the j -th variable of the i -th individual; and the values of scaling factor F^i and crossover probability CR^i are initially randomly generated between 0 and 1.

In our proposed improved self-adaptive parameter control, the fitness information of each individual is considered. It is introduced as follows:

$$F^i = \begin{cases} 0.1 + (F^i - 0.1) \times \frac{f(X^i) - f_{min}}{f_{avg} - f_{min}}, & \text{rnd}[0, 1] < \tau_1 \text{ and } f(X^i) < f_{avg} \\ \text{rnd}[0.1, 1], & \text{rnd}[0, 1] < \tau_1 \text{ and } f(X^i) \geq f_{avg} \\ F^i, & \text{otherwise} \end{cases} \quad (3)$$

$$CR^i = \begin{cases} CR^i \times \frac{f(X^i) - f_{min}}{f_{avg} - f_{min}}, & \text{rnd}[0, 1] < \tau_2 \text{ and } f(X^i) < f_{avg} \\ \text{rnd}[0, 1], & \text{rnd}[0, 1] < \tau_2 \text{ and } f(X^i) \geq f_{avg} \\ CR^i, & \text{otherwise} \end{cases} \quad (4)$$

where $\text{rnd}[a, b]$ is the uniform random variable between a and b . τ_1 and τ_2 indicate probabilities to adjust factors F^i and CR^i . f_{avg} and f_{min} are the average fitness and the minimal fitness of the current population, respectively.

The main idea of the modification is that when the fitness of the individual is lower than the average fitness in the current population, its CR and F values are shrunken occasionally. The offspring generated by the CR and F is located in the neighborhood of its parent. And hence, the offspring can exploit the useful information of the good parent and accelerate the convergence rate. Compared with SADE [6], our method does not add any additional parameters. To maintain the diversity of the population [17], the lower bound of F^i is 0.1, which is the same as used in SADE.

3.2 Handling the Boundary Constraint of Variables

After using the DE/rand/1/bin scheme to generate a new solution, if one or more of the variables in the new solution are outside their boundaries, i.e., $X_j \notin [l_j, u_j]$, the following repair rule is applied:

$$X_j = \begin{cases} l_j + \text{rnd}_j[0, 1] \times (u_j - l_j) & \text{if } X_j < l_j \\ u_j - \text{rnd}_j[0, 1] \times (u_j - l_j) & \text{if } X_j > u_j \end{cases} \quad (5)$$

where $\text{rnd}_j[0, 1]$ is the uniform random variable from $[0,1]$ in each dimension j . And l_j and u_j are the lower bound and upper bound of the j -th variable, respectively.

4 Experimental Results

In order to evaluate the performance of our method 10 benchmark functions ($f_{01} - f_{10}$) were used. The test functions are described in Table 1. All of the functions are minimization problems with $D = 30$. Functions $f_{01} - f_{04}$ are high-dimensional and unimodal problems. Also function f_{04} is a noisy quartic function, where $\text{random}[0,1)$ is a uniformly distributed random variable in $[0,1)$. Functions $f_{05} - f_{10}$ are high-dimensional and multimodal problems with many local minima, where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms.

Table 1. The 10 benchmark functions used in our experimental study, where D is the number of variables, “optimal” is the minimum value of the function, and $S \subseteq \mathbb{R}^D$. A detail description of all functions can be found in [18].

Test Functions	D	S	optimal
$f_{01} = \sum_{i=1}^D x_i^2$	30	$[-100, 100]^D$	0
$f_{02} = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	30	$[-10, 10]^D$	0
$f_{03} = \sum_{i=1}^{D-1} (x_i + 0.5)^2$	30	$[-100, 100]^D$	0
$f_{04} = \sum_{i=1}^D x_i^4 + \text{random}[0,1)$	30	$[-1.28, 1.28]^D$	0
$f_{05} = \sum_{i=1}^D (-x_i \sin(\sqrt{ x_i }))$	30	$[-500, 500]^D$	-12569.48662
$f_{06} = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	30	$[-5.12, 5.12]^D$	0
$f_{07} = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)) + 20 + \exp(1)$	30	$[-32, 32]^D$	0
$f_{08} = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^D$	0
$f_{09} = \frac{\pi}{6} \{10 \sin^2(\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 \cdot [1 + 10 \sin^2(\pi y_{i+1})] + (y_D - 1)^2\}$ $+ \sum_{i=1}^D u(x_i, 10, 100, 4)$	30	$[-50, 50]^D$	0
$f_{10} = \frac{1}{10} \{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2 [1 + \sin^2(2\pi x_D)]\}$ $+ \sum_{i=1}^D u(x_i, 5, 100, 4)$	30	$[-50, 50]^D$	0

Table 2. Best error values of DE, SADE, and ISADE on all test functions with $D = 30$ after 300,000 NFFEs. Where “Mean” indicates the mean best error values found in the last generation; “Std Dev” stands for the standard deviation.

F	DE			SADE			ISADE		
	Mean	Std Dev	SR	Mean	Std Dev	SR	Mean	Std Dev	SR
f_{01}	2.03E-33	5.48E-33	50	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50
f_{02}	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50
f_{03}	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50
f_{04}	3.54E-03	9.40E-04	50	3.31E-03	9.00E-04	50	3.20E-03	7.25E-04	50
f_{05}	3.81E+02	6.51E+02	3	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50
f_{06}	1.54E+01	1.05E+01	0	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50
f_{07}	4.14E-15	0.00E+00	50	4.14E-15	0.00E+00	50	4.14E-15	0.00E+00	50
f_{08}	1.87E-03	4.03E-03	40	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50
f_{09}	1.04E-02	6.02E-02	48	1.57E-32	0.00E+00	50	1.57E-32	0.00E+00	50
f_{10}	3.89E-03	1.93E-02	48	1.35E-32	0.00E+00	50	1.35E-32	0.00E+00	50

4.1 Experimental Setup

For all experiments, we use the following parameters unless a change is mentioned.

- Population size: $NP = 100$;
- $\tau_1 = \tau_2 = 0.1$;
- DE scheme: DE/rand/1/bin;
- Value to reach: $VTR = 10^{-8}$, except for f_{04} of $VTR = 10^{-2}$;
- Maximum NFFEs: $\text{Max_NFFEs} = 300,000$.

Moreover, in our experiments, each function is optimized over 50 independent runs. We also use the same set of initial random populations to evaluate different algorithms in a similar way done in [4]. All the algorithms are implemented in standard C++.

4.2 Performance Criteria

Five performance criteria are selected from the literature [19], [20] to evaluate the performance of the algorithms. These criteria are described as follows.

- **Error** [20]: The error of a solution X is defined as $f(X) - f(X^*)$, where X^* is the global optimum of the function. The minimum error is recorded when the Max_NFFEs is reached in 50 runs. Also the average and standard deviation of the error values are calculated.
- **NFFEs** [20]: The number of fitness function evaluations (NFFEs) is also recorded when the VTR is reached. The average and standard deviation of the NFFEs values are calculated.
- **Number of successful runs (SR)** [20]: The number of successful runs is recorded when the VTR is reached before the max_NFFEs condition terminates the trial.
- **Convergence graphs** [20]: The convergence graphs show the mean error performance of the total runs, in the respective experiments.

- **Acceleration rate (AR)** [19]: This criterion is used to compare the convergence speeds between ISADE and other algorithms. It is defined as follows: $AR = \frac{NFFE_{s_{other}}}{NFFE_{s_{ISADE}}}$, where $AR > 1$ indicates ISADE is faster than its competitor.

4.3 General Performance of ISADE

In order to show the superiority of our proposed ISADE approach, we compare it with the original DE algorithm and the SADE algorithm. The parameters used for DE, SADE, and ISADE are the same as described in Section 4.1. The control parameters of DE are set as $F = 0.5$ and $CR = 0.9$. All functions are conducted for 50 independent runs. Table 2 shows the best error values of DE, SADE, and ISADE on all test functions. The average and standard deviation of NFFEs are shown in Table 3. The best results are highlighted in **Bold** face. In addition, some representative convergence graphs of DE, SADE, and ISADE are shown in Fig. 1.

When compared with DE: From Table 2 it can be seen that ISADE can obtain the optimal or close-to-optimal solutions for all test functions. And the standard deviations for all functions are very small, which means that our proposed ISADE method is very robust. It can solve these functions over all 50 runs. However, for DE it fails to solve two functions (f_{05} and f_{06}). For three functions (f_{08} , f_{09} , and f_{10}), DE locates the local minima some times. Table 3 shows that ISADE is able to achieve faster convergence rate for all test functions. Additionally, it is apparent that ISADE obtains higher convergence velocity than DE according to Fig. 1.

When compared with SADE: Table 2 shows that both SADE and ISADE can solve these functions over 50 runs. Nevertheless, from Table 3 and Fig. 1 we can see that ISADE converges faster than SADE. It needs less NFFEs to reach the *VTR* for all test functions.

In summary, our proposed ISADE approach can achieve better performance than DE in terms of both the convergence rate and the quality of the final solutions. Both SADE

Table 3. NFFEs Required to obtain accuracy levels less than *VTR*. “NA” indicates the accuracy level is not obtained after 300, 000 NFFEs. Where “1 vs 3” means “DE vs ISADE” and “2 vs 3” indicates “SADE vs ISADE”.

F	DE		SADE		ISADE		AR 1 vs 3	AR 2 vs 3
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev		
f_{01}	6.92E+04	1.55E+03	5.43E+04	1.03E+03	4.90E+04	9.73E+02	1.41	1.11
f_{02}	1.04E+05	1.98E+03	7.56E+04	1.44E+03	6.46E+04	1.12E+03	1.62	1.17
f_{03}	2.54E+04	1.86E+03	2.06E+04	7.84E+02	1.89E+04	7.43E+02	1.35	1.09
f_{04}	1.07E+05	3.16E+04	1.11E+05	2.01E+04	1.02E+05	2.34E+04	1.04	1.09
f_{05}	2.72E+05	1.55E+04	8.94E+04	2.27E+03	7.77E+04	1.98E+03	3.50	1.15
f_{06}	NA	NA	1.14E+05	3.59E+03	8.62E+04	2.61E+03	NA	1.32
f_{07}	1.07E+05	1.85E+03	8.26E+04	1.73E+03	7.36E+04	1.22E+03	1.45	1.12
f_{08}	7.22E+04	2.79E+03	5.74E+04	2.84E+03	5.13E+04	1.97E+03	1.41	1.12
f_{09}	6.32E+04	3.34E+03	5.00E+04	1.26E+03	4.65E+04	1.25E+03	1.36	1.07
f_{10}	8.03E+04	2.09E+04	6.03E+04	1.79E+03	5.58E+04	1.49E+03	1.44	1.08

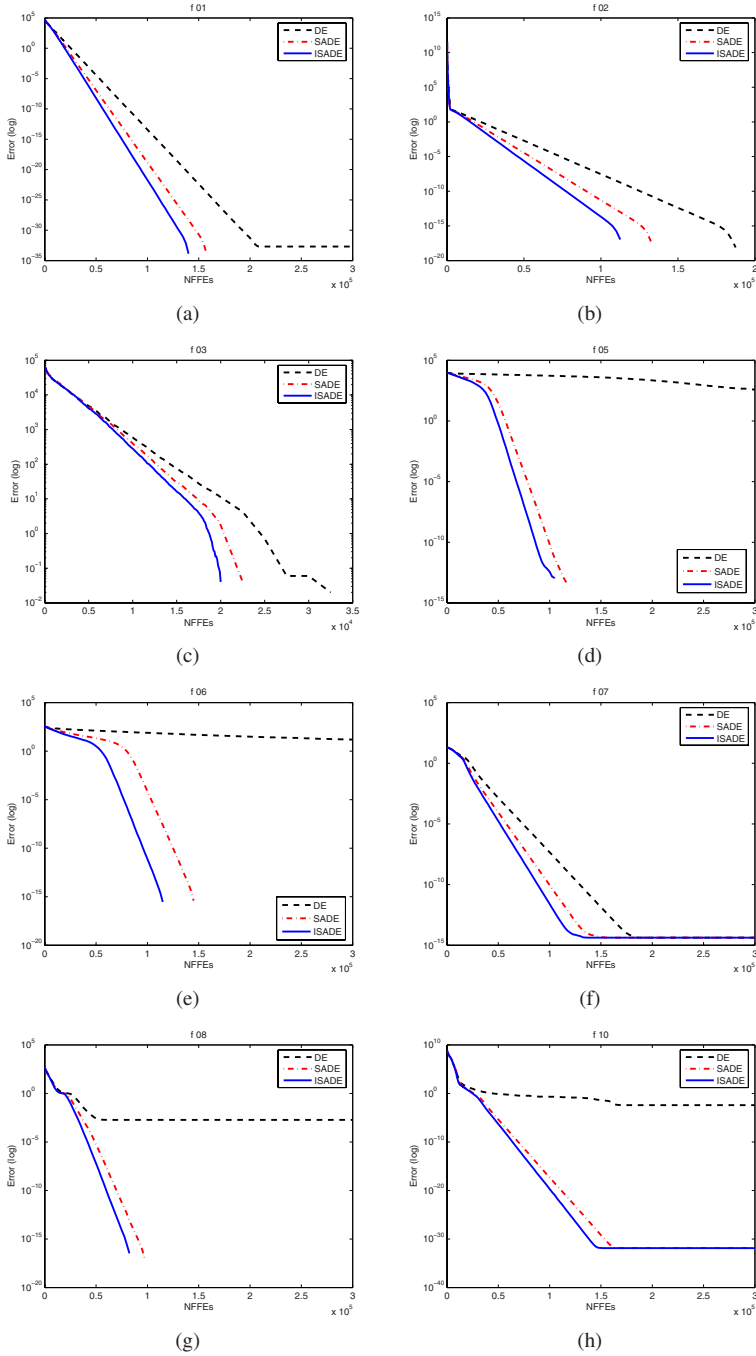


Fig. 1. Mean error curves of DE, SADE, and ISADE for 8 selected functions. (a) f_{01} . (b) f_{02} . (c) f_{03} . (d) f_{05} . (e) f_{06} . (f) f_{07} . (g) f_{08} . (h) f_{10} . Log-scale of Y axis is to make the comparison more clearly for all functions.

and ISADE obtain similar results with respect to the best error values. While ISADE is able to converge faster than SADE. The experimental results confirm that our proposed improved self-adaptive control parameter can exploit the fitness information of each individual efficiently. It can enhance the performance of the original DE algorithm.

5 Conclusion and Future Work

In this paper, we propose an improved self-adaptive control parameter for the DE algorithm. Our proposed method is an extension of SADE. In our approach the fitness information of each individual is used to adapt the CR and F values of DE. In this manner, our proposed approach can exploit the useful information of its parent, and hence, it can accelerate the convergence rate compared with the original SADE method.

In order to verify the viability of our proposed ISADE method, 10 high-dimensional benchmark functions are chosen from literature. Moreover, five performance criteria are employed to compare the performance of ISADE with those of DE and SADE. Experimental results indicate that ISADE can achieve better performance than DE in terms of both the convergence rate and the quality of the final solutions. Both SADE and ISADE obtain similar results with respect to the best error values. While ISADE is able to converge faster than SADE.

Our future work will test our approach on more various functions. In addition, using ISADE for the constrained optimization is another direction in our future work.

Acknowledgments

The authors would like to thank Prof. Brest for providing the SADE code.

References

1. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11(4), 341–359 (1997)
2. Storn, R., Price, K.: Home page of differential evolution (2008)
3. Price, K., Storn, R., Lampinen, J.: *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin (2005)
4. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation* 12(1), 107–125 (2008)
5. Gaperle, R., Muler, S., Koumoutsakos, P.: A parameter study for differential evolution. In: *Proc. WSEAS Int. Conf. Advances Intell. Syst., Fuzzy Syst., Evol. Comput.*, pp. 293–298 (2002)
6. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation* 10(6), 646–657 (2006)
7. Alatas, B., Akin, E., Karci, A.: Modenar: Multi-objective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing* 8(1), 646–656 (2008)
8. Das, S., Abraham, A., Konar, A.: Automatic clustering using an improved differential evolution algorithm. *IEEE Transaction on Systems Man and Cybernetics: Part A* 38(1), 218–237 (2008)

9. Chakraborty, U.: *Advances in Differential Evolution*. Springer, Berlin (2008)
10. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. *Soft Comput.* 9(6), 448–462 (2005)
11. Salman, A., Engelbrecht, A.P., Omran, M.G.H.: Empirical analysis of self-adaptive differential evolution. *European Journal of Operational Research* 183(2), 785–804 (2007)
12. Nobakhti, A., Wang, H.: A simple self-adaptive differential evolution algorithm with application on the ALSTOM gasifier. *Appl. Soft Comput* 8(1), 350–370 (2008)
13. Das, S., Konar, A., Chakraborty, U.K.: Two improved differential evolution schemes for faster global search. In: Beyer, H.G., O'Reilly, U.M. (eds.) *Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29*, pp. 991–998. ACM, New York (2005)
14. Teo, J.: Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.* 10(8), 673–686 (2006)
15. Brest, J., Maučec, M.S.: Population size reduction for the differential evolution algorithm. *Appl. Intell.* 29(3), 228–247 (2008)
16. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics* 24(4), 656–667 (1994)
17. Zaharie, D.: Critical values for the control parameters of differential evolution algorithms. In: Matoušek, R., Ošmera, P. (eds.) *MENDEL 2002, 8th International Mendel Conference on Soft Computing, June 5-7*, pp. 62–67 (2002)
18. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 3(2), 82–102 (1999)
19. Rahnamayan, S., Tizhoosh, H., Salama, M.: Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation* 12(1), 64–79 (2008)
20. Suganthan, P., Hansen, N., Liang, J.: Problem definitions and evaluation criteria for the cec2005 special session on real-parameter optimization (2005)