International Journal of Computational Intelligence Systems, Vol. 6, No. 2 (March, 2013), 000-000

An Efficient Binary Differential Evolution with Parameter Adaptation

Dongli Jia *
School of Information and Electronic Engineering, Hebei University of Engineering, 056038, China

Xintao Duan
School of Computer and Information Engineering, Henan Normal University, 453000, China

Muhammad Khurram Khan
Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh, Saudi Arabia

Received 21 December 2010
Accepted 13 November 2012

Abstract

Differential Evolution (DE) has been applied to many scientific and engineering problems for its simplicity and efficiency. However, the standard DE cannot be used in a binary search space directly. This paper proposes an adaptive binary Differential Evolution algorithm, or ABDE, that has a similar framework as the standard DE but with an improved binary mutation strategy in which the best individual participates. To further enhance the search ability, the parameters of the ABDE are slightly disturbed in an adaptive manner. Experiments have been carried out by comparing ABDE with two binary DE variants, normDE and BDE, and the most used binary search technique, GA, on a set of 13 selected benchmark functions and the classical 0-1 knapsack problem. Results show that the ABDE performs better than, or at least comparable to, the other algorithms in terms of search ability, convergence speed, and solution accuracy.

Keywords: Computational Intelligence; Evolutionary Computation; Differential Evolution; Genetic Algorithm; Binary optimization.

1. Introduction

Differential Evolution (DE), which was first proposed over 1994-1996 by Storn and Price at Berkeley, is a simple yet powerful evolutionary algorithm [1-5]. It has been proved that DE is an accurate, reasonably fast, and robust optimizer for many optimization problems in real-world applications such as filter design, PID control, image segmentation, and other scientific and engineering problems [6-15]. The DE has a similar framework with Genetic Algorithm but only a few control variables. Therefore it is easy to use in optimizing problems.

Most researchers focus their attention on DE in continuous optimization applications and a lot of improved variants of DE have been presented recently such as Self-adaptive Differential Evolution (SaDE)[16], Fuzzy Adaptive Differential Evolution (FADE)[17], Adaptive DE with Optional Archive(JADE)[18], jDE [19, 20] etc. However, many problems are set in a discrete search space where the standard DE cannot be implemented directly [21, 22]. Besides, for easy of hardware-implementation, continuous optimization problems are usually solved in a binary number space. However, unfortunately, only a few researches pay their attentions to the binary DE algorithm. Gong and Tuson proposed a binary Differential Evolution (BDE) algorithm in [23] where the continuous difference between two individuals in standard DE is represented by a hamming distance in the binary search space. Similar with binary particle swarm optimization (BPSO)[24], Engelbrecht and Pampara also proposed a

* Corresponding author, present affiliation: School of Information and Electronic Engineering, Hebei University of Engineering, China. Email: jwdsli@gmail.com.

Co-published by Atlantis Press and Taylor & Francis
Copyright: the authors

normalized binary Differential Evolution algorithm(normDE) in CEC 07 [25].

In this paper, we proposed a novel binary DE algorithm with parameters adaptation (ABDE) based on DE/best/1/bin strategy of the standard DE for both continuous and discrete problems optimization. Different from the other two binary DE variants, BDE and normDE, the scaling factor F was transformed into a binary string in the proposed ABDE algorithm. Compared with Genetic Algorithm (GA), BDE and normDE, the proposed ABDE algorithm shows a better optimization performance on a set of test problems.

The remainder of this paper is organized as follows. Section 2 describes the classic DE procedure and some basic concepts. In Section 3, the new proposed ABDE algorithm is elaborated with detailed explanations. Simulations are presented in Section 4 for the comparison and analysis. At the end, Section 5 concludes the findings in the paper.

2. Differential Evolution

In standard Differential Evolution, the individual (also called a solution to the problem) is first generated in a continuous search space randomly. Then it experienced the iteration of mutation, crossover, and selection steps until the termination criterion is met.

Suppose that there is a minimization problem $f(X)$.

$$\begin{aligned} &\min f(X),\ \ X=[x_1,x_2,...,x_D], \\ &s.t.\ \ x_i \in [a_i,b_i] \end{aligned} \tag{1}$$

where X is the decision vector consisting of D variables. a_i and b_i are the lower and upper boundaries of x_i, respectively.

In the standard DE model, each candidate X can be coding as an individual in a continuous search space. All individuals form a population for further evolution. At each generation, individuals are evaluated to assess their quality and the best member is marked to track the evolution progress.

A. Initialization

A candidate solution to objective function $f(X)$ can be coding as

$$X_i^g = [x_{i1}^g, x_{i2}^g, ..., x_{iD}^g], i \in \{1,2,...,NP\} \tag{2}$$

where x_{ij}^g is the j-th component of the i-th individual in g-th generation. NP is the population size. D is the dimension of objective function $f(X)$. In initialization, all individuals are randomly generated with the uniform probability distribution.

B. Mutation

In standard DE, the mutation is executed in a continuous search space. For each individual vector X_i^g, a continuous difference is calculated between two randomly selected individuals. Then the difference is scaled by multiplied a scaling factor F. The mutation equation is

$$\begin{aligned} &V_i^g = X_{r3}^g + F \cdot (X_{r2}^g - X_{r1}^g), \\ &r_1, r_2, r_3 \in \{1,2,...,NP\}, F \in [0,1] \end{aligned} \tag{3}$$

where X_{r3}^g, X_{r2}^g and X_{r1}^g are selected individuals from the population but different from the running individual X_i^g. The scaling factor F is closely related to the convergence speed.

C. Crossover

In the crossover scheme, the components of the individuals exchange their position according to the following equation:

$$u_{ij}^g = \begin{cases} v_{ij}^g, & rand(j) \le CR, \text{or } j = rnbr(i) \\ x_{ij}^g, & rand(j) > CR \text{ or } j \ne rnbr(i) \end{cases}, \quad j \in \{1,2,...,D\} \tag{4}$$

where u_{ij}^g is a component of the candidate child U_i^g, and the v_{ij}^g is a component of V_i^g. $rnbr(i)$ is a randomly chosen index $\in \{1,2,...,D\}$. CR is the crossover factor. In an adaptive DE algorithm, F and CR usually vary with the evolution process.

D. Selection

The candidate child U_i^g and the individual X_i^g compete in their fitness. The winner will survive for the next generation.

$$X_i^{g+1} = \begin{cases} U_i^g, & \text{if } f(X_i^g) \ge f(U_i^g) \\ X_i^g, & \text{otherwise} \end{cases} \tag{5}$$

The DE iterates above B, C and D steps until the termination criterion is met.

Copyright: the authors

Different mutation and crossover schemes can form different DE strategies. Generally, they are defined as $DE/x/y/z$. The x specifies the individual who will take part in the mutation, y is the number of difference vectors, and z denotes the crossover scheme.

3. Discrete Differential Evolution with parameters adaptation

This section proposes a novel discrete DE with parameters adaptation for the discrete space optimization. The basic difference between the discrete DE algorithm and the standard DE lies in the mutation strategy. In standard DE, mutation is implemented based on the scaled difference between two selected individuals in a continuous search space. In the proposed binary DE algorithm, the mutation of ith individual can be defined as:

$$\begin{aligned} V_i^g &= X_{best}^g \wedge D \\ D_j &= (\text{if } rand_j < F) \,\&\, (x_{2j}^g \wedge x_{1j}^g) \end{aligned} \tag{6}$$

where, '^' is the xor operator and '&' is the and operator. D_j is the scalded difference between two individuals. The scaling factor F indicates the accepted probability of the component difference between x_{1j}^g and x_{2j}^g in two selected individuals X_2^g and X_1^g.

Our adaptation scheme is designed based on the DE/best/1/bin strategy, so the X_{best}^g will always be the best member in the population.

An instance of the binary mutation is illustrated in Fig. 1.

In the proposed binary DE, the crossover and the selection strategies are the same as them in the standard DE.

To further improve the optimization performance, we introduce a parameter adaptation scheme into the binary DE. First, each individual in the population is assigned an F and a CR value as depicted in Fig 2.

x_{1j}	1	1	0	1	1
x_{2j}	0	1	1	1	0
$rand_j < F$	1	0	1	1	0
$x_{best,j}$	0	1	1	0	1
v_j	1	1	0	0	1

Fig. 1 An instance of binary mutation

X_1^g	F_1^g	CR_1^g
X_2^g	F_2^g	CR_2^g
⋮	⋮	⋮
X_{NP}^g	F_{NP}^g	CR_{NP}^g

Fig. 2 Adaptation encoding format

In the initialization phase, all the F and CR are originally set to be 0.65 and 0.25, respectively. Then in every evolutionary generation, the F and CR of each individual are slightly disturbed based on the following equation.

$$\begin{aligned} F_i^{g+1} &= \begin{cases} normrnd(mF, 0.05), & \text{if } rand_i < 0.05 \\ F_i^g, & \text{otherwise} \end{cases} \\ CR_i^{g+1} &= \begin{cases} normrnd(mCR, 0.05), & \text{if } rand_i < 0.05 \\ CR_i^g, & \text{otherwise} \end{cases} \end{aligned} \tag{7}$$

where, mF and mCR are the mean F and mean CR values, respectively. $normrnd(\mathrm{g})$ indicates a normal distribution with a mean of mF or mCR and a variance of 0.05.

In selection step, the individual with a better fitness will survive the next generation and the associated F and CR will be also propagated to the next generation.

The whole procedure of the adaptive DE based on the DE/best/1/bin strategy is described below:

- Step1. Initialization
  - Step1.1. Set the population size NP, and the stop criteria;
  - Step1.2. Initialize all individuals randomly; set the F and CR to be 0.65 and 0.25, respectively.
  - Step1.3. Evaluate $f(X)$ over all individuals.
- Step2. Iteration
  - Step2.1. Execute mutation according to equation (6) and adjusting F and CR for each individual;
  - Step2.2. Execute crossover according to equation (4);
  - Step2.3. Execute selection according to equation (5);

Copyright: the authors

- Step3. If the stopping criteria are met, output the best solution. Otherwise, jump to step2.

4. Experiments

4.1. *Algorithms for comparison*

The most-used binary evolutionary algorithm, GA, and two binary DE variants, normDE and BDE, are compared over the test benchmarks. In order to demonstrate the effects of our mutation strategy on improving the performance of ABDE, the ABDE without parameters adaptation (ABDE_W) was also simulated.

normDE:

normDE is originally proposed by Engelbrecht and Pampara in CEC 07[25]. Similarly with BPSO [24], the normDE uses the floating-point DE individuals to generate a binary DE bit string. First, each component of each individual is linearly scaled to the range of [0, 1].

$$x_{ij}^{'}(g) = (x_{ij}(g) + |x_i^{\min}(g)|)/(|x_i^{\min}(g)| + x_i^{\max}(g)) \tag{8}$$

where $x_{ij}(g)$ is the *j*-th component of *i*-th individual in *g*-th generation. $x_i^{\max}$ and $x_i^{\min}$ are the largest and the smallest component values of the *i*-th individual. The binary bit string then generated using

$$y_{ij}(g) = \begin{cases} 0, & \text{if } x_{ij}^{'}(g) < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

The bit string is used by the fitness function to determine its quality and in turn associated with floating-point representation of the individual.

BDE:

BDE is originally proposed by Gong and Tuson in [23]. There are two mutation operators introduced in BDE. One operator considers each decision variable as a single dimension and the other operator considers all decision variables as a single dimension. In our experiment, we only consider the latter one. Firstly, evaluate the distance d between X_2^g and X_3^g over all variables. Then, scale the distance d' with F, $d^{'} = F \cdot D$. Finally, round the scaled distance to a suitable integer number ($\in [0, D]$) to apply to another apply binary string (i.e. X_1^g).

$$D(Mutant, X_1^g) = \begin{cases} (\text{int})d^{'} + 1, & \text{if } rand < d^{'} - (\text{int})d^{'} \\ (\text{int})d^{'}, & \text{otherwise} \end{cases} \tag{10}$$

GA:

GA is the most-used binary evolutionary algorithm and we can find a lot of paper related to GA. In our comparison, we use the standard GA and Dejong parameter settings as introduced in [26].

4.2. *Benchmark functions*

Experiments are carried out over a set of 13 widely used benchmark functions described in appendix 1. These functions are taken from [27] by Yao et al each with different characteristics. Most of them are also tested in other binary evolution algorithm such as in [25, 23]. Among these benchmarks, F1-F4 are unimodal functions, F5 is the Rosenbrock function which is unimodal function for D=2 and multimodal function for D>3[28], F6 is a step function, F7 is a noisy quartic function, F8-F13 are multimodal function.

4.3. *Parameter settings and numerical results*

All simulations are carried out over 13 benchmark functions. The parameters of the normDE are set as recommended in [25]. For BDE, the F and CR values used in [23] are problem dependent. In our simulation, we use the median values which the authors of BDE have used. We fixed the parameters of BDE to F=0.3, CR=0.6 for all benchmark functions. For GA, the parameters are set to two point crossover with rate of 0.6, bit flip mutation with rate of 0.001, and roulette wheel selection as recommended by Dejong in [26]. The parameters of ABDE are set as discussed above. The F and CR values are fixed to 0.65 and 0.25 for ABDE_W, respectively. For all binary DE variants and GA, the population size are set to be 60 and the max generation 1000 for all 13 benchmark functions at dimension D=10（every dimension is represented by a length of 20 binary bit string）.

The averaged and standard deviations (within parentheses) of the best values for 25 independent runs of each algorithm are presented in Table 1. The best solutions and its standard deviation have been shown in boldface.

Copyright: the authors

Table 1. Mean best values and its standard deviation at D=10

	GA	normDE	BDE	ABDE_W	ABDE	Statistical Significance
F1	**9.09E-08 (0.00E+00)**	1.11E+00 (3.32E-01)	2.33E+01 (4.55E+01)	3.38E-02 (1.30E-01)	1.29E-01 (4.98E-01)	+
F2	**9.54E-05 (0.00E+00)**	2.12E-01 (2.58E-02)	1.74E-01 (9.74E-02)	6.91E-04 1.42(E-03)	7.49E-04 (2.02E-03)	+
F3	1.49E+02 (1.18E+02)	1.27E+02 (4.90E+01)	8.74E+01 (7.59E+01)	3.08E-01 (9.99E-01)	**2.60E-01 (9.84E-01)**	+
F4	**9.68E-01 (9.38E-01)**	1.79E+00 (2.28E-01)	1.49E+01 (7.80E+00)	2.27E+00 (2.75E+00)	2.48E+00 (3.39E+00)	+
F5	8.45E+00 (7.86E+00)	8.26E+01 (2.51E+01)	1.35E+04 (5.77E+04)	4.52E+00 (5.06E+00)	**3.43E+00 (2.67E+00)**	+
F6	1.48E+00 (5.03E+00)	5.20E-01 (5.10E-01)	1.19E+01 (2.11E+01)	4.00E-02 (2.00E-01)	**0.00E+00 (0.00E+00)**	+
F7	3.69E-02 (1.76E-02)	1.27E-02 (3.59E-03)	6.66E-03 (2.54E-03)	3.22E-03 (1.52E-03)	**3.02E-03 (1.45E-03)**	+
F8	4.99E+02 (2.22E+02)	1.65E+02 (8.79E+01)	5.94E+01 (7.59E+01)	**2.14E-03 (6.15E-03)**	9.69E-03 (2.87E-02)	+
F9	1.29E+01 (5.03E+00)	1.19E+01 (2.32E+00)	6.13E+00 (2.82E+00)	1.41E-01 (3.38E-01)	**5.37E-02 (1.98E-01)**	+
F10	**1.22E-04 (0.00E+00)**	6.48E-01 (1.31E-01)	1.92E+00 (1.20E+00)	5.31E-02 (2.31E-01)	5.75E-02 (2.45E-01)	+
F11	3.86E-01 (6.34E-01)	6.14E-01 (1.14E-01)	6.09E-01 (2.54E-01)	**2.30E-02 (1.51E-02)**	2.33E-02 (2.51E-02)	+
F12	6.49E-01 (1.39E+00)	1.84E-02 (8.05E-03)	3.14E+02 (1.08E+03)	**1.75E-02 (6.48E-02)**	2.56E-02 (1.09E-01)	+
F13	2.40E-02 (2.73E-02	6.83E-02 1.70E-02)	3.89E+02 (1.24E+03)	**2.44E-04 (9.77E-04)**	2.70E-04 (1.07E-03)	+

From Table 1, it can be seen that our strategy is superior to GA and the other two binary DE variants. In 9 out of 13 cases in Table 1, the ABDE or ABDE_W achieves the best error values. If we make a deep insight into the results, we will find that, in many cases, the ABDE_W reaches the best results. However, in some other cases, the ABDE is superior to the ABDE_W because of its parameters adaptation strategy.

In the final columns of Table 1, a two-tailed T test was employed to show the statistical significance level of the difference of the means of two best algorithms. A sign '+' indicates the t value of 24 degrees of freedom is significant at a 0.05 level of significance, a sign ' • ' stand for the difference is not statistically significant.

4.4. *Convergence performance*

Convergence speed is very important for evolutionary optimization algorithms [6, 25]. To demonstrate the convergence performance of the proposed ABDE algorithm, we compared it with the binary DE variants and GA over four benchmark functions each with different characteristics. F3 is unimodal function, F5 is the Rosenbrock function, F7 is a noisy quartic function, and F8 is multimodal function. The convergence graphs of the algorithms over test problems are shown in Fig 3. Please note that the convergence graphs are the mean best function values of 25 independent runs at each generation. From the Fig 3, it can be seen that our algorithm not only has the fastest convergence speed, but it obtains the best solutions to the four test functions.

4.5. *Effects of the parameters adaptation*

To show the effects of the parameters adaptation scheme, we recorded the variation tendency of F and CR values in the evolution processes over F7-F11 benchmark functions and illustrated them in Fig. 4 and Fig. 5.

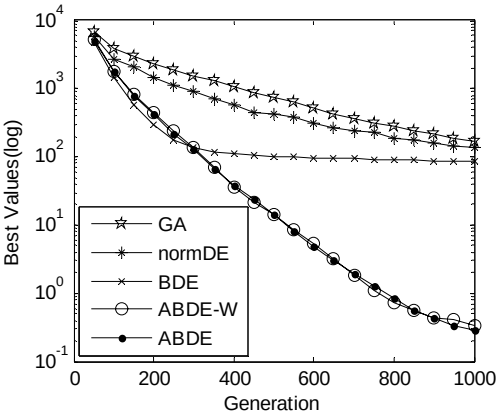
From Fig. 4 and Fig. 5, it can be seen that the variant tendency of F and CR are problem dependent. With the progress of the evolution, the F and CR tend to become smaller almost for all these test functions.

4.6. *Test on the 0-1 knapsack problem*

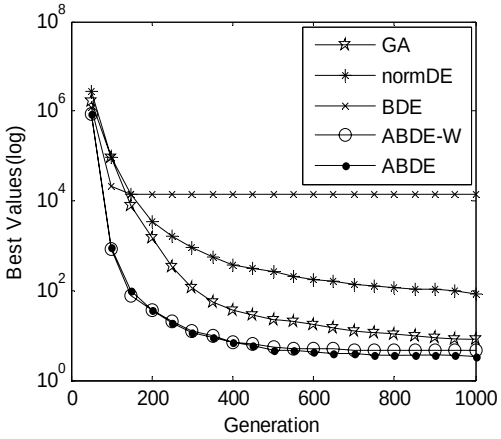
The 0-1 knapsack problem (KP) is one of the classical NP-hard problems with binary decision variables. The

Copyright: the authors

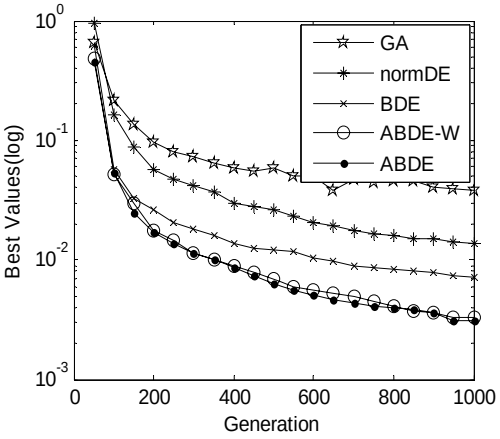
0-1 KP problem is generally descried as: given a set of N items, each item i has a profit p_i and a weight w_i, the problem is how to choose a subset of the items to maximize the overall profit [29].



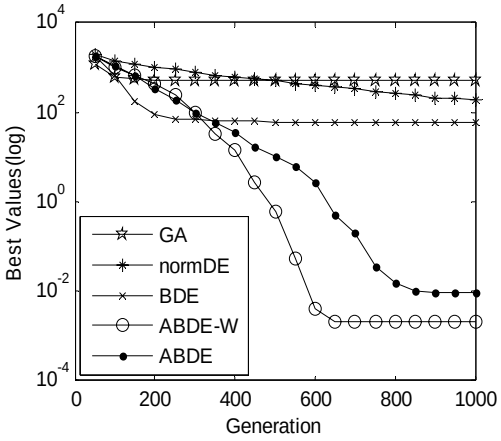
Function F3



Function F5



Function F7



Function F8

Fig. 3 Convergence performance of the proposed algorithm

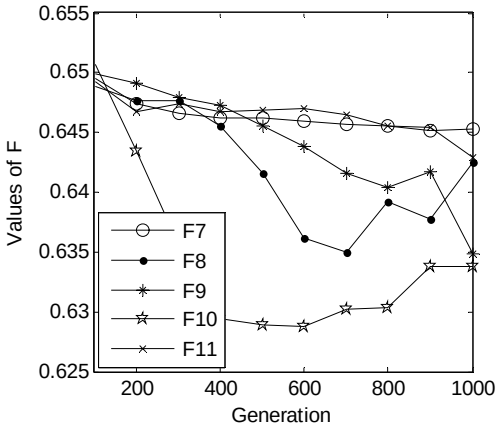


Fig. 4 Variant tendency of F

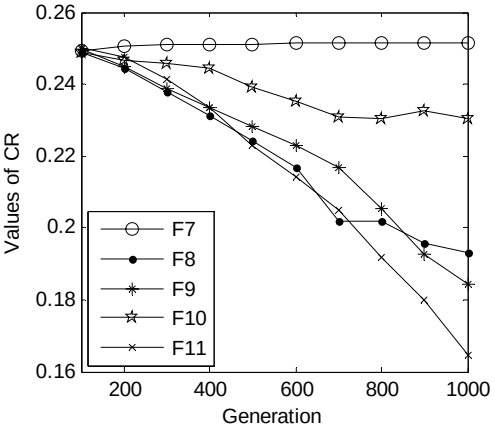


Fig. 5. Variant tendency of CR

Copyright: the authors

In the test, we assume that the number of items is $N = 100$, all the weight w_i and profit p_i are integer values uniformly random distributed in [1,200]. The settings of the algorithms are the same as mentioned in section 4.3.

The items for test are:

W = {17 94 50 139 136 172 155 176 6 142 21 113 166 120 167 184 86 51 41 173 62 143 166 16 95 107 58 77 19 22 48 197 102 79 1 25 125 184 190 193 46 38 126 71 151 46 71 197 127 13 120 150 84 166 25 48 68 12 191 150 76 12 196 141 200 34 16 169 53 120 97 43 75 128 133 103 199 108 185 186 89 96 81 192 32 137 11 191 62 7 139 200 35 65 98 100 181 76 141 43}

P = {172 46 108 100 190 146 164 19 73 160 26 199 32 22 63 154 173 3 143 180 196 113 113 72 162 97 14 5 95 37 91 86 27 159 139 140 118 20 14 101 83 134 62 129 42 155 180 116 62 93 130 9 2 130 92 83 194 82 95 103 187 118 124 47 164 170 159 17 5 167 151 137 130 16 168 196 158 120 142 21 33 65 77 148 59 125 48 72 111 22 100 30 80 187 40 153 58 7 158 106}

The results are summarized in Table 2.

Table 2 Solutions to 0-1 knapsack problem

Algorithm	Best	Mean Best	Std. Dev.
GA	7090	6834	1.79E+02
normDE	7220	7154	3.68E+01
BDE	7274	7147	6.01E+01
ABDE	7388	7388	0.00E+00

From Table 2, it can be seen that not only the ABDE obtains the best solutions to 0-1 KP problem, but it is the most stable algorithm with the minimum standard deviation.

5. Conclusions

DE is a recently developed simple yet powerful evolutionary algorithm. Due to ease of implementation, DE has been applied to many scientific and engineering problems. However, the standard DE can not be used in a binary search space directly. In this paper, we proposed a simple but effective binary DE algorithm, ABDE, that has a similar framework with the standard DE and an improved binary mutation strategy. To adapt to different problems, we control the parameters of the ABDE in an adaptive manner in the evolution process. Comparisons have been carried out on a set of 13 selected benchmarks and the classical 0-1 knapsack problem. Results show that the ABDE is superior to the other binary DE variants, normDE and BDE, and the most-used binary search algorithm, GA, in terms of the convergence speed, search ability, and robustness.

In our future study, we will apply the proposed ABDE algorithm to more real life problems. We will also compare the ABDE with more binary search technique to further verify its effectiveness.

Acknowledgement

This paper is partially supported by the National Nature Science Foundation of China (NO. U1204606).

References

1. S. Das and P. N. Suganthan, "Differential evolution - a survey of the state-of-the-art", IEEE Transactions on Evolutionary Computation, 15(1)(2011) 4-31.
2. S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation, 13(3) (2009) 526-52
3. K. Price, R. Storn, and J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, 1st ed. New York: Springer-Verlag,. 2005.
4. R Storn, K Price, Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11(4) (1997) 341-359.
5. R. Storn and K. Price, Minimizing the real functions of the ICEC'96 contest by differential evolution, Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 1996, pp.842-844.
6. S. Dasgupta, A. Biswas, S. Das and A. Abraham, Modeling and analysis of the population dynamics of differential evolution algorithm, AI Communications - The European Journal on Artificial Intelligence, IOS Press, Netherlands, 22(1) (2009) 1 – 20
7. S. Das, S. Sil, Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm, Information Sciences, 180(8) (2010) 1237-1256
8. R. Joshi, A. C. Sanderson, Minimal representation multisensory fusion using differential evolution, IEEE Transaction on . Systems, Man and Cybern. Part A, 29(1) (1999) 63–76.
9. A.C. Nearchou, A differential evolution approach for the common due date early/tardy job scheduling problem, Computers & Operations Research, 2008,35(4), (2008) 1329-1343
10. Q.K. Pan, L. Wang, B. Qian, A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems, Computers & Operations Research, 36(8) (2009) 2498-2511.
11. B. Qian, L. Wang, D.X. Huang, W.L. Wang, X. Wang, An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers , Computers & Operations Research, 36(1) (2009) 209-233.
12. M.F. Tasgetiren, Q.K. Pan, Y.C. Liang, A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence

Copyright: the authors

dependent setup times, Computers & Operations Research, 36(6) (2009)1900-1915.
13. Y. Wang, B. Li, T. Weise, Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems, Information Sciences, 180(12) (2010) 2405-2420
14. J. Zhang, V. Avasarala, and R. Subbu, Evolutionary optimization of transition probability matrices for credit decision-making, European Journal of Operational Research, 20(2) (2010) 557-567.
15. M. Zhang, W. Luo, X.F. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences, 178(15) (2008) 3043-3074
16. A. K. Qin and P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in Proceedings of IEEE Congress on Evolutionary Computation, vol. 2. 2005, pp. 1785–1791.
17. J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Comput.: Fusion Found., Methodologies Applicat., 9(6) (2005) 448–462.
18. J.Q. Zhang, A. C. Sanderson, JADE: Adaptive Differential Evolution with Optional External Archive, IEEE Transactions on Evolutionary Computation, 13(5) (2009) 945-958.
19. J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, Selfadapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation, 10(6) (2006) 646–657.
20. J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, V. Zumer, High-dimensional Real-parameter Optimization Using Self-adaptive Differential Evolution Algorithm with Population Size Reduction, 2008 IEEE World Congress on Computational Intelligence, 2008, pp. 2032-2039.
21. X. He, Q. Zhang, N. Sun, Y. Dong, Feature selection with discrete binary differential evolution. International Conference on Artificial Intelligence and Computational Intelligence. (2009), pp.327-330.
22. S. Dutta, D. Datta, A Binary-Real-Coded Differential Evolution for Unit Commitment Problem, International Journal of Electrical Power & Energy Systems, 42(1) (2012) 517-524.
23. T. Gong and A. Tuson, Differential Evolution of Binary Encoding, Soft Computing in Industrial Application, vol. ASC39, 2007, pp.251-262
24. J.Kennedy and R.Eberhart, A Discrete Binary Version of the Particle Swarm Algorithm, Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, 1997, pp.4104-4109
25. A.P. Engelbrecht, G. Pampara, Binary Differential Evolution Strategies[A], IEEE Congress on Evolutionary Computation[C], Singapore, 2007. 1942 – 1947
26. K.A. Dejong, W.M. Spears, An analysis of the interacting roles of population size and crossover in Genetic Algorithms. Proceedings of the first workshop on parallel problem solving from nature. Springer-Verlag, Berlin, 1990. pp.38-47
27. X. Yao, Y. Liu, and G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation., 3(2) (1999) 82–102.
28. Y. W. Shang and Y.H. Qiu, A note on the extended rosenbrock function, Evol. Comput., vol. 14, no. 1, pp. 119–126, 2006.
29. S. Martello, D. Pisinger, P. Toth, New trends in exact algorithm for 0-1 knapsack problem. European journal of operation research, 123(2), (2000) 325-332

Appendix A.

1. . $F1(X)=\sum_{i=1}^{D}x_i^2\ ;\ -100\le x_i\le 100\ ;\ F1^*(X)=F1(0,\cdots,0)=0$.
2. $F2(X)=\sum_{i=1}^{D}|x_i|+\prod_{i=1}^{D}x_i\ ;\ -10\le x_i\le 10\ ;\ F2^*(X)=F2(0,\cdots,0)=0$.
3. $F3(X)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2\ ;\ -100\le x_i\le 100\ ;\ F3^*(X)=F3(0,\cdots,0)=0$.
4. $F4(X)=\max|x_i|\ ;\ -100\le x_i\le 100\ ;\ F4^*(X)=F4(0,\cdots,0)=0$.
5. $F5(X)=\sum_{i=1}^{D-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right];\ -30\le x_i\le 30\ ;\ F5^*(X)=F5(1,\cdots,1)=0$.
6. $F6(X)=\sum_{i=1}^{D}\left(\lfloor x_i+0.5\rfloor\right)^2\ ;\ -100\le x_i\le 100\ ;\ F6^*(X)=F6(x_1,\cdots,x_D)=0\ ;\ -\frac{1}{2}\le x_i<\frac{1}{2}$.
7. $F7(X)=\sum_{i=1}^{D}ix_i^4+rand[0,1)\ ;\ -1.28\le x_i\le 1.28\ ;\ F7^*(X)=F7(0,\cdots,0)=0$.

Copyright: the authors

8. $F8(X)=\sum_{i=1}^{D}-x_i\sin\left(\sqrt{|x_i|}\right)+D\cdot 418.98288727243369$; $-500\le x_i\le 500$;
$F8^*(X)=F8(0,\cdots,0)=0$.

9. $F9(X)=\sum_{i=1}^{D}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$; $-5.12\le x_i\le 5.12$; $F9^*(X)=F9(0,\cdots,0)=0$.

10. $F10(X)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)+20+e$; $-32\le x_i\le 32$;
$F10^*(X)=F10(0,\cdots,0)=0$.

11. $F11(X)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$; $-600\le x_i\le 600$; $F11^*(X)=F11(0,\cdots,0)=0$.

12. $F12(X)=\frac{\pi}{D}\left\{10\sin^2\left(\pi y_1\right)+\sum_{i=1}^{D-1}\left(x_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+\left(y_D-1\right)^2\right\}+\sum_{i=1}^{D}u\left(x_i,10,100,4\right)$;
$-50\le x_i\le 50$; $F12^*(X)=F12(-1,\cdots,-1)=0$.

where, $y_i=1+\frac{1}{4}\left(x_i+1\right)$ and $u\left(x_i,a,k,m\right)=\begin{cases}k\left(x_i-a\right)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$

13. $F13(X)=0.1\left\{\sin^2\left(3\pi x_1\right)+\sum_{i=1}^{D-1}\left(x_i-1\right)^2\cdot\left[1+\sin^2\left(3\pi x_{i+1}\right)\right]\right\}$
$+\left(x_D-1\right)^2\left\{1+\sin^2\left(2\pi x_n\right)\right\}+\sum_{i=1}^{D}u\left(x_i,5,100,4\right)$; $-50\le x_i\le 50$;
$F13^*(X)=F13(1,\cdots,1,1)=0$.

Copyright: the authors