Adaptive Differential Evolution with *p*-Best Crossover for Continuous Global Optimization

Sk Minhazul Islam, Saurav Ghosh, Subhrajit Roy, and Swagatam Das

Dept. of Electronics and Telecommunication Engg.,
Jadavpur University, Kolkata 700 032, India
skminha.isl@gmail.com, saurav_online@yahoo.in,
roy.subhrajit20@gmail.com, swagatamdas19@yahoo.co.in

Abstract. Differential Evolution (DE) is arguably one of the most powerful stochastic real parameter optimization algorithms in current use. DE operates through the similar computational steps as employed by a standard Evolutionary Algorithm (EA). However, unlike the traditional EAs, the DE-variants perturb the current-generation population members with the scaled differences of randomly selected and distinct population members. Therefore, no separate probability distribution has to be used, which makes the scheme self-organizing in this respect. Its performance, however, is still quite dependent on the setting of control parameters such as the mutation factor and the crossover probability according to both experimental studies and theoretical analyses. Our aim is to design a DE algorithm with control parameters such as the scale factor and the crossover constants adapting themselves to different problem landscapes avoiding any user intervention. Further to improve the convergence performance an innovative crossover mechanism is proposed here.

Keywords: Differential Evolution, Numerical Optimization, Parameter Adaptation, *p*-best Crossover.

1 Introduction

Scientists and engineers from all disciplines often have to deal with the classical problem of global optimization where the main target is to determine a set of model parameters or state-variables that provide the globally minimum or maximum value of a predefined cost or objective function, or a set of optimal tradeoff values in the case of two or more conflicting objectives. In this article we consider bound-constrained single-objective optimization problems that involve D decision variables represented in a vector like $\vec{X} = [x_1, x_2, x_3, ..., x_D]^T$ and a scalar objective function (or fitness function) to judge the quality of the solution that we have achieved. Locating the global optimal solutions becomes very challenging for single-objective problems arising in many practical situations, due to the presence of high dimensionality, strong epitasis, ill-conditioning, and multimodality of the objective function. Differential Evolution [1]-[5],[8] has emerged as a very competitive optimizer for continuous search spaces, exhibiting remarkable performances in several competitions held under the IEEE Congress on Evolutionary Computation (CEC). A detail survey of the DE

B.K. Panigrahi et al. (Eds.): SEMCCO 2010, LNCS 6466, pp. 119–128, 2010.

family of algorithms can be found in [25].In this paper we propose a simple yet very powerful adaptive DE variant depicted as ADEpBX which implements a modified version of the mutation scheme "DE/target-to-best/1/bin" along with an innovative type of crossover scheme named as *p*-best crossover. The detailed description of the algorithm is discussed in section 3.

2 Classical DE

DE is a simple real-coded evolutionary algorithm. In this section we describe the basic operations of DE and introduce necessary notations and terminologies which facilitate the explanation of our adaptive DE algorithm later.

2.1 Initialization of the Parameter Vectors

DE searches for a global optimum point in a *D*-dimensional continuous hyperspace. It begins with a randomly initiated population of *Np* (population number) *D* dimensional real-valued parameter vectors. Each vector, also known as *genome/chromosome*, forms a candidate solution to the multi-dimensional optimization problem. We shall denote subsequent generations in DE by $G = 0,1...,G_{\max}$ .Here we adopt the following notation for representing the *i*-th vector of the population at the current generation:

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, ..., x_{D,i,G}]. \quad (1)$$

For each parameter of the problem, there may be a certain range within which the value of the parameter should lie for better search results. The initial population (at $G = 0$) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds:

$$\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, ..., x_{D,\min}\} \text{ and } \vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, ..., x_{D,\max}\}.$$

Hence we may initialize the *j*-th component of the *i*-th vector as:

$$x_{j,i,0} = x_{j,\min} + rand_{i,j}(0,1) \cdot (x_{j,\max} - x_{j,\min}) \quad (2)$$

Here rand is a uniformly distributed random number lying between 0 and 1 (actually $0 \le rand_{i,j}(0,1) < 1$) and is instantiated independently for each component of the *i*-th vector.

2.2 Mutation with Difference Vectors

After initialization, DE creates a *donor* vector $\vec{V}_{i,G}$ corresponding to each population member or target vector $\vec{X}_{i,G}$ in the current generation through mutation. It is the method of creating this donor vector, which differentiates between the various DE schemes. The following are the two most frequent mutation strategies used in the literature:

1)"DE/rand/1"

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) \tag{3}$$

2) "DE/target-to-best/1"

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F \cdot (\vec{X}_{best,G} - \vec{X}_{i,G}) + F \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \tag{4}$$

The indices r_1^i, r_2^i, r_3^i, r_4^i, and r_5^i are distinct integers uniformly chosen from the set {1, 2, . . ., *Np*}/{i}.These indices are randomly generated once for each donor vector. The scaling factor *F* is a positive control parameter for scaling the difference vectors. $\vec{X}_{best,G}$ is the best individual vector with the best fitness (i.e. lowest objective function value for minimization problem) in the population at generation *G*.

2.3 Crossover

To enhance the potential diversity of the population, a crossover operation comes into play after generating the donor vector through mutation. The donor vector exchanges its components with the target vector $\vec{X}_{i,G}$ under this operation to form the *trial* vector $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, ..., u_{D,i,G}]$.The scheme may be outlined as:

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } (rand_{i,j}(0,1) \le Cr \text{ or } j = j_{rand} \\ x_{j,i,G}, & \text{otherwise,} \end{cases} \tag{5}$$

Here, $rand_{i,j}(0,1)$ is a uniformly distributed random number, for each *j*-th component of the *i*-th parameter vector. $j_{rand} \in \{1,2,....,D\}$is a randomly chosen index, which ensures that $\vec{U}_{i,G}$ gets at least one component from $\vec{V}_{i,G}$. These *F*, *Cr* and population number are the control parameters of the basic DE.

2.4 Selection

The next step of the algorithm calls for *selection* to determine whether the target or the trial vector survives to the next generation i.e. at $G = G+1$. The selection operation is described as:

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G}, & if \quad f(\vec{U}_{i,G}) \le f(\vec{X}_{i,G}) \\ \vec{X}_{i,G}, & if \quad f(\vec{U}_{i,G}) \ge f(\vec{X}_{i,G}) \end{cases} \tag{6}$$

where $f(\vec{X})$ is the objective function to be minimized.

3 ADEpBX

In this section, we propose a new DE algorithm, ADEpBX, which implements an innovative mutation strategy and controls *F* and *Cr* in an adaptive [9]-[20] manner. This algorithm also proposes an innovative type of crossover scheme.

3.1 New Mutation Strategy: -DE/target-to-poprandbest /bin/1

DE/target-to-best/1 is one of the widely used mutation strategies in DE where the new donor vector used to perturb each population member, is created using any two randomly selected member of the population as well as the best vector of the current generation. The notation of this strategy has been shown earlier.

ADEpBX uses a newly proposed mutation strategy named as *DE/target-to-poprandbest /bin/1*. *DE/target-to-poprandbest/1* is actually a modified version of DE/target-to-best/1.

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F \cdot (\vec{X}_{poprandbest,G} - \vec{X}_{i,G} + \vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}). \tag{7}$$

$X_{(poprandbest)}$ is the best of q % vectors randomly chosen from the current population. $X_{(r1,g)}$ and $X_{(r2,g)}$ are vectors chosen randomly from the current population.

The main problem with DE/target-to-best/1/bin scheme is that this scheme uses the best vector to generate the mutated vectors. It promotes exploitation as all the population vectors are attracted towards the best position. As a result of such exploitative tendency the population can lose its global exploration abilities and can get trapped to some locally optimal point on the search space. In ADEpBX it has been aimed to maintain a balance between the two contradictory aspects of DE-exploration and exploitation. So some modifications have been introduced to overcome the limitations of fast but less reliable convergence performance of target-to-best scheme. Instead of using the best vector, the best of a dynamic group of q% vectors randomly chosen from the current population has been used to generate the donor vector. As a result the strategy proposed somewhat diminishes the greediness nature of the target-to-best scheme enhancing the population diversity and a proper trade-off between exploration and exploitation has been achieved to some extent.

3.2 Parameter Adaptation

Scale Factor adaptation: At every generation the scale factor F_i *of* each individual target vector is independently generated according to a Cauchy distribution with location parameter F_m and scale parameter 0.1i.e.

$$F_i = randci(F_m, 0.1) \tag{8}$$

The Cauchy distribution is implemented in the algorithm in such a way that the values of F_i for all target vectors lie between 0 and 1. Denote $F_{success}$ as the set of all successful scale factors of the current population generating better trial vectors that are likely to advance to the next generation and *mean(F)* as the mean of all the scale

factors associated with the target vectors of the current population. F_m of the Cauchy distribution is initialized to be 0.5 and then updated at the end of each generation in the following manner:-

Case 1: *mean (F)* < 0.85

$$F_m = (0.9 + 0.01 \cdot abs(randn)) \cdot F_m + 0.1 \cdot (1 + 0.01 \cdot abs(randn)) \cdot powermean(F_{success}) \quad (9)$$

Case 2: *mean (F)* > 0.85

$$F_m = (0.85 + 0.01 \cdot abs(randn)) \cdot F_m + 0.1 \cdot (1 + 0.01 \cdot abs(randn)) \cdot powermean(F_{success}) \quad (10)$$

$$powermean(F_{success}) = \sum x \in F_{success} (x^n / length(F_{success}))^{1/n} \quad (11)$$

Here the argument n is taken as 1.5.

Crossover probability adaptation: At every generation the crossover probability Cr_i of each individual vector *is* independently generated according to a normal distribution of mean Cr_m and standard deviation 0.1i.e.

$$Cr_i = randn_i(Cr_m, 0.1) \quad (12)$$

And then kept in [0, 1]. Denote Cr_{success} as the set of all successful crossover probabilities Cr_i's at the current generation. The mean Cr_m is initialized to be 0.7 and then updated at the end of each generation as

$$Cr_m = (0.9 + 0.001 \cdot abs(randn)) \cdot Cr_m + 0.1 \cdot (1 + 0.001 \cdot abs(randn)) \cdot powermean(Cr_{success}) \quad (13)$$

$$powermean(Cr_{success}) = \sum x \in Cr_{success} (x^n / length(Cr_{success}))^{1/n} \quad (14)$$

Here the argument n is taken as 1.5.

Explanation of parameter adaptation: In DE three most important parameters [6], [21] that affect the algorithm performance are the *Np*, *F* and *Cr*. Here *Np* has been kept fixed. The scale factor, *F,* actually improves the convergence speed while the crossover probability, *Cr*, is associated with the property and complexity of the problem. During the adaptation of F_m the usage of Power mean leads to higher value of F_m that accounts for larger perturbation to the target vectors, thus avoiding premature convergence at local optima. The essence of $F_{success}$ is that it memorizes the successful scale factors thus glorifying the chance of creating better donor vectors by extending it to the following generations. F_m is used as a location parameter of Cauchy distribution which rather diversifies the values of *F* as compared to the traditional normal distribution [7].For the adaption of Cr_m the usage of $Cr_{success}$ again records the successful *Cr* values, thus generates better individuals as offsprings which are more likely to survive. A normal distribution with mean Cr_m and standard deviation of 0.1 is used to generate the *Cr* values. The usage of power mean instead of arithmetic mean in adaptation of Cr_m leads to higher values of *Cr* which eliminates the implicit bias of *Cr* towards small values.

3.3 *p*-Best Crossover Operation

The crossover operation used in ADEpBX named as *p*-best crossover where for each donor vector, a vector is randomly selected from the *p* top-ranking vectors (according to their objective function values) in the current population and then normal binomial crossover is performed as per eqn. (5) between the donor vector and the randomly selected *p*-best vector to generate the trial vector at the same index. Thus the offspring achieve a much high probability to advance to the subsequent generation due to injection of information of the *p*-best vector. The *p*-best crossover operation has served a splendid purpose in enhancing the convergence speed of the algorithm which is inevitable in single objective optimization as the number of function evaluations is limited.

4 Experimental Setup and Results

4.1 Benchmark Functions Used

We have used a test - bed of twenty-five well - known boundary-constrained benchmark functions to evaluate the performance of the new adaptive DE variant. These functions constituted the benchmark of CEC-2005 competition on single objective optimization.

4.2 Algorithms Compared

The results of ADEpBX on the above test bed have been compared to the following algorithms:

- DE/rand/1/bin [22]
- JADE with $c = 0.1$, $p = 0.05$ and optional external archive.[20]
- jDE with F_l=0.1,F_u=0.9 and $\tau 1$=$\tau 2$=0.1[16]
- Self Adaptive DE(SaDE)[23]
- DMS-PSO [24].

4.3 Simulation Strategies and Parameter Settings

Functions f_1 to f_{25} were tested in 30 dimensions (30D). The maximum number of function evaluations (FEs) was set to 3e+05. Two important parameters in ADEpBX are: q controls the greediness of the mutation strategy target-to-poprandbest and *p* determines the *p*-best crossover operation. Both these parameters are insensitive to different objective functions as evident from their roles. ADEpBX usually executes best with q set to $1/4^{th}$ of the population size which is kept as 100 for all the assigned problems and *p* is dynamically updated with generation (or iteration) according to the equation shown below.

$$p = \left\lceil \frac{Np}{2}\left(1 - \frac{iter}{iter\_\max}\right)\right\rceil + 1 \tag{15}$$

Here *Np* represents the population size and *iter*_max represents the maximum number of iterations allotted. The reduction routine of *p* favors exploration at the beginning of the search and exploitation during the later stages by gradually downsizing the elitist portion of the population, with a randomly selected member from where the component mixing of the donor vector is allowed for formation of the trial vector. For the contestant algorithms, we employ the best suited parametric set-up, chosen with guidelines from their respective literatures. All the algorithms are tested on a Pentium core 2 duo machine with 2 GB RAM and 2.23 GHz speed.

4.4 Results on Numerical Benchmarks

Tables 1 to 5 show the mean and the standard deviation of the best-of-run errors for 50 independent runs of each of the nine algorithms on twenty-five numerical benchmarks for $D = 30$ respectively. Note that the best-of-the-run error corresponds to absolute difference between the best-of-the-run value $f(\vec{x}_{best})$ and the actual optimum f^* of a particular objective function i.e. $\left|f(\vec{x}_{best}) - f^*\right|$.

Table 1. Mean and standard deviation of error values for function1-5

Algorithms	f_1	f_2	f_3	f_4	f_5
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	2.4536e-28 (3.4756e-28)	5.49e-08 (1.2e-07)	2.89e+05 (1.93e+05)	5.04e-01 (8.58e-01)	1.2719e+03 (2.83e+02)
JADE	1.3258se-54 (9.2436e-54)	2.5146e-26 (3.4269e-26)	4.7421e+04 (1.6213e+04)	5.1159e-07 (4.0194e-07)	3.2792e+02 (1.8494e+02)
jDE	3.8652e-29 (4.5732e-29)	7.5064e-06 (7.3804e-06)	2.2663e+05 (1.6085e+05)	2.7305e-01 (2.7305e-01)	1.1108e+03 (3.7238e+02)
SaDE	6.7843e-30 (2.3879e-30)	9.7191e-08 (4.8596e-07)	5.0521e+04 (1.5754e+05)	5.8160e-06 (1.4479e-05)	7.8803e+02 (1.2439e+03)
DMS-L-PSO	2.3462e-20 (5.6234e-20)	1.1757e-07 (6.5592e-08)	1.6343e-06 (3.9247e-06)	2.5487e+03 (3.0638e+02)	2.1858e+03 (8.2641e+02)
ADEpBX	**1.3429e-62** (**2.4352e-61**)	**1.9981e-26** (**2.4429e-26**)	4.0977e+04 (3.2699e+04)	**6.9268e-08** (**8.9742e-08**)	2.2057e+02 (1.6754e+02)

Table 2. Mean and standard deviation of error values for function 6-10

Algorithms	f_6	f_7	f_8	f_9	f_{10}
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	3.7764e+00 (2.71e+00)	9.6575e-01 (9.14e-02)	2.0941e+01 (6.25e-02)	5.3742e-14 (6.7342e-14)	6.1643e+01 (4.56e+01)
JADE	5.6094e+00 (1.9445e+01)	6.9598e-03 (4.4845e-03)	2.0929e+01 (2.6628e-02)	5.9272e-22 (8.9543e-22)	3.0313e+01 (8.3551e+00)
jDE	1.1196e+01 (1.3987e+00)	9.8597e-03 (3.4824e-03)	2.093`e+01 (2.5067e-02)	8.3264e-16 (2.3645e-15)	5.2547e+01 (4.4660e+00)
SaDE	2.1248e+001 (1.3413e+01)	8.2727e-03 (1.1445e-02)	2.0140e+01 (5.7258e-02)	2.2737e-15 (1.1369e-14)	3.5758e+01 (6.0809e+00)
DMS-L-PSO	4.7840e-01 (1.3222e+00)	6.9990e-03 (4.5371e-03)	**2.0000e+01** (**2.3029e-04**)	1.7591e+01 (3.0222e+00)	3.7410e+01 (5.2883e+00)
ADEpBX	**3.9873e-01** (**1.0815e+00**)	**6.6472e-03** (**9.0313e-03**)	**2.0000e+01** (**6.7185e-06**)	1.0342e+01 (3.2346e+00)	**2.8890e+01** (**8.9159e+00**)

Table 3. Mean and standard deviation of error values for function 11-15

Algorithms	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	3.2648e+01 (1.0954e+01)	8.4331e+04 (6.2535e+04	4.5130e+00 (2.2662e+00	1.3347e+01 (3.4764e-01)	4.8432e+02 (2.1467e+0
JADE	2.6456e+01 (1.9169e+01)	2.6978e+04 (6.8003e+03	1.6285e+00 (4.8739e-02)	1.2771e+01 (2.2057e-01)	2.8884e+02 (9.0503e+0
jDE	3.1370e+01 (2.3952e+00)	3.8376e+04 (6.5374e+03	1.8568e+00 (1.0313e-01)	1.3545e+01 (9.9402e-02)	2.9642e+02 (1.8711e+0
SaDE	2.6562e+01 (1.1275e+00)	8.7345e+02 (9.3383e+02	1.2070e+00 (1.3420e-01)	1.2760e+01 (2.5936e-01)	3.2775e+02 (9.6450e+0
DMS-L-PSO	2.7278e+01 (1.5739e+00)	2.5359e+02 (2.8883e+02)	2.3595e+00 (5.2823e-01)	1.2961e+01 (4.1146e-01)	3.4400e+02 (5.0662e+01)
ADEpBX	**1.7590e+01 (6.0615e+00)**	9.5793e+04 (8.133e+03)	**1.1051e+00 (5.6060e-02)**	**1.2429e+01 (3.4320e-01)**	**2.8553e+02 (9.7542e+0**

Table 4. Mean and standard deviation of error values for function 16-20

Algorithms	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	2.8228e+02 (1.1328e+01)	3.0942e+02 (1.5698e+01	9.1342e+02 (8.4336e-01)	9.1984e+02 (1.2190e+00	9.1317e+02 (1.1642e+00
JADE	7.4383e+01 (3.9432e+01)	8.4619e+01 (3.5763e+01	8.1679e+02 (1.6523e-01)	8.1644e+02 (1.5419e-01)	8.1697e+02 (1.7231e-01)
jDE	1.2854e+02 (4.0730e+01)	1.6189e+02 (4.7251e+01	8.6111e+02 (1.8705e+00	8.4801e+02 (3.1790e+00	8.5466e+02 (9.54e-01)
SaDE	1.379e+02 (1.702e+01)	1.509e+03 (9.363e+02)	9.544e+02 (3.438e+01)	8.458e+02 (6.215e+01)	2.040e+03 (8.768e+02)
DMS-L-PSO	1.1950e+02 (1.2068e+02)	1.4519e+02 (7.3247e+01)	9.1053e+02 (1.5761e+00)	9.1060e+02 (1.3383e+00)	9.0189e+02 (3.0719e+01)
ADEpBX	**7.2307e+01 (3.8872e+01)**	**8.2328e+01 (3.9757e+01**	**8.1626e+02 (1.5209e-01)**	**8.1625e+02 (1.5340e-01)**	**8.1642e+02 (1.6990e-01)**

Table 5. Mean and standard deviation of error values for function 21-25

Algorithms	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)	Mean(Std)
DE/rand/1/bin	5.8135e+02 (2.6247e+01)	9.6425e+02 (1.1439e+01	6.2131e+02 (3.0647e+01	3.14e34+02 (3.2249e+01	9.8643e+02 (2.1775e+01
JADE	8.5838e+02 (1.1013e+00)	5.0762e+02 (1.6550e+00	8.6558e+02 (6.6102e-01)	2.1141e+02 (1.5664e+01	2.1135e+02 (3.5588e+00
jDE	8.6002e+02 (1.1361e+00)	5.0340e+02 (2.9115e+00	6.1835e+02 (4.5481e+00	2.1081e+02 (2.8842e+00	9.761e+02 (2.409e+01)
SaDE	1.730e+03 (5.118e+02)	1.582e+03 (4.252e+02)	5.506e+02 (2.489e+01)	**2.0000e+02 (1.2455e-04)**	5.0012e+02 (5.683e-02)
DMS-L-PSO	5.0134e+02 (1.0132e-04)	9.2154e+02 (9.841e+01)	5.344e+02 (4.001e-01)	**2.0000e+02 (5.0123e-04)**	9.889e+02 (3.015e+01)
ADEpBX	**5.0000e+02 (0)**	**5.0021e+02 (4.5755e-01)**	**5.3416e+02 (7.8384e-04)**	2.0985e+02 (3.5882e+00	**2.0962e+002 (3.6271e+00**

4.5 Analysis of Results

Tables 1-5 indicate that out of 25 in 19 cases ADEpBX ranked first considering the mean error. It is to be noted that ADEpBX has shown superiority in case of rotated, hybrid as well as functions with noise in fitness. So function rotation, incorporating multiplicative noise in them as well as composition of functions does not hamper the performance of the proposed algorithm significantly. SADE, JADE and DMS-PSO

remained as the toughest competitors for ADEpBX though the latter has yielded better results in majority of the cases.

5 Conclusion

Over past one decade of research DE has reached an impressive state. Nowadays an extensive research work is going on in designing various DE algorithms to optimize large-scale high-dimensional problems (D=1000, D=500). ADEpBX has been tested in the CEC 2010 high dimensional benchmark functions where it has been observed to give promising results in case of non-separable functions. The parameter adaptation and the mutation scheme which are the main features in ADEpBX can be quite useful in constraint, multi-objective, multi-modal, large-scale and dynamic optimization where our initial studies have shown exciting results, there are still many open questions in incorporating parameter adaptation schemes to evolutionary optimization.

References

[1] Storn, R., Price, K.: Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization 11(4), 341–359 (1997)
[2] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization, 1st edn. Springer, New York (2005)
[3] Joshi, R., Sanderson, A.C.: Minimal representation multisensory fusion using differential evolution. IEEE Trans. Syst., Man Cybern. 29(1), 63–76 (1999)
[4] Zhang, J., Avasarala, V., Subbu, R.: Evolutionary optimization of transition probability matrices for credit decision-making. Eur. J. Oper. Res. (to be published)
[5] Zhang, J., Avasarala, V., Sanderson, A.C., Mullen, T.: Differential evolution for discrete optimization: An experimental study on combinatorial auction problems. In: Proc. IEEE World Congr. Comput. Intell., Hong Kong, China, pp. 2794–2800 (June 2008)
[6] Gamperle, R., Muller, S.D., Koumoutsakos, P.: A parameter study for differential evolution. In: Proc. Advances Intell. Syst., Fuzzy Syst., Evol. Comput., Crete, Greece, pp. 293–298 (2002)
[7] Zhang, J., Sanderson, A.C.: An approximate Gaussian model of differential evolution with spherical fitness functions. In: Proc. IEEE Congr. Evol. Comput., Singapore, pp. 2220–2228 (September 2007)
[8] Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.A.: A comparative study of differential evolution variants for global optimization. In: Proc. Genetic Evol. Comput. Conf., Seattle, WA, pp. 485–492 (July 2006)
[9] Abbass, H.A.: The self-adaptive pareto differential evolution algorithm. In: Proc. IEEE Congr. Evol. Comput., Honolulu, HI, vol. 1, pp. 831–836 (May 2002)
[10] Teo, J.: Exploring dynamic self-adaptive populations in differential evolution. Soft Comput.: Fusion Found., Methodologies Applicat. 10(8), 673–686 (2006)
[11] Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Comput.: Fusion Found., Methodologies Applicat. 9(6), 448–462 (2005)
[12] Xue, F., Sanderson, A.C., Bonissone, P.P., Graves, R.J.: Fuzzy logic controlled multiobjective differential evolution. In: Proc. IEEE Int. Conf. Fuzzy Syst., Reno, NV, pp. 720–725 (June 2005)

[13] Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proc. IEEE Congr. Evol. Comput., vol. 2, pp. 1785–1791 (September 2005)
[14] Huang, V.L., Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In: Proc. IEEE Congr. Evol. Comput., pp. 17–24 (July 2006)
[15] Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. 10(6), 646–657 (2006)
[16] Brest, J., Zumer, V., Maucec, M.S.: Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: Proc. IEEE Congr. Evol. Comput., Vancouver, BC, pp. 215–222 (July 2006)
[17] Brest, J., Boskovic, B., Greiner, S., Zumer, V., Maucec, M.S.: Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Comput.: Fusion Found., Methodologies Applicat. 11(7), 617–629 (2007)
[18] Yang, Z., Tang, K., Yao, X.: Self-adaptive differential evolution with neighborhood search. In: Proc. IEEE Congr. Evol. Comput., Hong Kong, China, pp. 1110–1116 (June 2008)
[19] Angeline, P.J.: Adaptive and self-adaptive evolutionary computations. In: Computational Intelligence: A Dynamic Systems Perspective, pp. 152–163 (1995)
[20] Zhang, J., Sanderson, A.C.: JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In: Proc. IEEE Congr. Evol. Comput., Singapore, pp. 2251–2258 (September 2007)
[21] Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3(2), 124–141 (1999)
[22] Ronkkonen, J., Kukkonen, S., Price, K.: Real-parameter optimization using Differential Evolution. In: IEEE CEC 2005 (2005)
[23] Qin, A.K., Suganthan, P.N.: Self-adaptive Differential Evolution for numerical optimization. In: IEEE CEC 2005 (2005)
[24] Liang, J.J., Suganthan, P.N.: Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search. In: IEEE CEC 2005 (2005)
[25] Das, S., Suganthan, P.N.: Differential Evolution: A Survey of the State-of-the-art. IEEE Trans. Evol. Comput. (2010)