Modifications of Differential Evolution with Composite Trial Vector Generation Strategies

Josef Tvrdík

Department of Computer Science, University of Ostrava,
30. dubna 22, 701 03 Ostrava, Czech Republic
josef.tvrdik@osu.cz

Abstract. Differential evolution algorithm with composite trial vector generation strategies and control parameters has been proposed recently. The performance of this algorithm is claimed to be better or competitive in comparison with the state-of-the-art variants of differential evolution. When we attempted to implement the algorithm according to the published description, several modified variants appear to follow the description of the algorithm. These variants of the algorithm were compared experimentally in benchmark problems. One of newly proposed variants outperforms the other variants significantly, including the variant used by the authors of the algorithm in their published experimental comparison.

1 Introduction

Differential evolution (DE) was proposed by Storn and Price [9] as a global optimizer for unconstrained continuous optimization problems with a real-value objective function. The search space (domain) S is specified by lower (a_j) and upper (b_j) limits of each component j, $S = \prod_{j=1}^{D}[a_j, b_j]$, $a_j < b_j$, $j = 1, 2, \ldots, D$, D is the dimension of the problem. The global minimum point x^*, satisfying condition $f(x^*) \leq f(x)$ for $\forall x \in S$ is the solution of the problem.

DE algorithm has become one of the evolutionary algorithms most frequently used for solving the continuous global optimization problems in recent years [7]. Comprehensive summarizations of the up-to date results in DE are presented by Neri and Tirronen [5] and by Das and Suganthan [3].

Algorithm of DE works with a population of individuals (*NP* points in domain *S*) that are considered as candidates of solution. Parameter *NP* is called the size of the population. The population is developed iteratively by using evolutionary operators of selection, mutation, and crossover. Each iteration corresponds to an evolutionary generation. Let us denote two subsequent generations by *P* and *Q*. Applications of evolutionary operators in the old generation *P* create a new generation *Q*. After completing the new generation *Q*, the *Q* becomes the old generation for next iteration. The basic structure of DE algorithm is shown in Algorithm 1.

The trial vector y is generated (line 5 in Algorithm 1) by crossover of two parent vectors, the current (target) vector x_i and a mutant vector v. The mutant vector v is obtained by a mutation. Several kinds of mutation have been proposed in last years. Three kinds of mutation used in algorithms compared in this study are described below.

Algorithm 1. Differential evolution

```
1:  generate an initial population $P = (x_1, x_2, \ldots, x_{NP})$, $x_i \in S$ distributed uniformly
2:  evaluate $f(x_i),\ i = 1, 2, \ldots, NP$
3:  while stopping condition not reached do
4:    for $i = 1$ to $NP$ do
5:      generate a trial vector $y$
6:      evaluate $f(y)$
7:      if $f(y) \le f(x_i)$ then
8:        insert $y$ into new generation $Q$
9:      else
10:       insert $x_i$ into new generation $Q$
11:     end if
12:   end for
13:   $P := Q$
14: end while
```

Symbols r_1, r_2, r_3, r_4, and r_5 denote mutually distinct points taken randomly from the current generation P, not coinciding with the target point x_i, $F > 0$ is an input control parameter, and U is uniformly distributed random value between 0 and 1.

- *rand/1*

$$v = r_1 + F\,(r_2 - r_3)\,. \tag{1}$$

- *rand/2.*

$$v = r_1 + F\,(r_2 - r_3) + F\,(r_4 - r_5)\,. \tag{2}$$

- *current-to-rand/1*

$$y = x_i + U\,(r_1 - x_i) + F\,(r_2 - r_3). \tag{3}$$

The current-to-rand/1/ mutation generates directly a trial point y because it includes so called arithmetic crossover represented by the second term on the right side of (3).

The crossover operator constructs the trial vector y from current individual x_i and the mutant vector v. Two types of crossover were proposed in [9]. One of them is binomial crossover which generates a new trial vector y by using the following rule

$$y_j = \begin{cases} v_j & \text{if} \quad U_j \le CR \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > CR \quad \text{and} \quad j \ne l\,, \end{cases} \tag{4}$$

where l is a randomly chosen integer from $\{1, 2, \ldots, D\}$, and $U_1, U_2, \ldots, U_D$ are independent random variables uniformly distributed in $[0,\ 1)$. $CR \in [0, 1]$ is a control parameter influencing the number of elements to be exchanged by the crossover. Eq. (4) ensures that at least one element of x_i is changed, even if $CR = 0$.

Mutation according to (1), (2), or (3) could cause that a new trial point y moves out of the domain S. In such a case, the values of $y_j \notin [a_j, b_j]$ is turned over into S by using transformation $y_j = 2 \times a_j - y_j$ or $y_j = 2 \times b_j - y_j$ for the violated component.

2 DE with Composite Trial Vector Generation Strategies

DE algorithm with composite trial vector generation strategies and control parameters, in abbreviation *composite DE*, was presented by Wang et al. [12] and compared with the state-of-the-art DE variants considered best performing, namely *jDE* [1], *EPSDE* [4], *SaDE* [8], and *JADE* [13]. From the results In benchmark tests [10], *composite DE* outperformed all of these algorithms except *EPSDE*. In comparison with *EPSDE*, *composite DE* was competitive.

The *composite DE* combines three well-studied trial-vector strategies with three control parameter settings in a random way to generate trial vectors. The strategies are: rand/1/bin, rand/2/bin, and current-to-rand/1/. All the strategies are carried out with a pair of F and CR values randomly chosen from a parameter pool. It results in having three candidates to a trial vector in each iteration step (line 5 in Algoritmus 1) that compete by tournament. Thus, three function evaluations are needed in each step. The vector with the least function value of those three candidates is then used as a trial vector. The parameter pool used in [12] contains the following pairs of control parameters: $[F = 1.0,\ CR = 0.1]$, $[F = 1.0,\ CR = 0.9]$, and $[F = 0.8,\ CR = 0.2]$.

After the first reading of the paper [12], the *composite DE* algorithm was implemented as described above. This variant of the algorithm is labeled by *CoDE* hereafter.

However, the following part of the paper [12] induced doubts about the right form of the algorithm (the symbols in the text within quotation marks are changed to be compatible with the symbols used in this paper): "*After mutation, the current-to-rand/1 strategy uses the rotation-invariant arithmetic crossover rather than the binomial crossover, to generate the trial vector [2,6]. As a result, this strategy is rotation-invariant and suitable for rotated problems. The arithmetic crossover in this strategy linearly combines the mutant vector with the target vector as follows:*

$$y = x_i + U\,(v - x_i) \tag{5}$$

where U is a uniformly distributed random number between 0 and 1. Note that for the arithmetic crossover the crossover control parameter CR is not needed". It is not quite clear if this text is meant only as an explanation how the current-to-rand/1 works or if the current-to-rand/1 defined in (3) is used for generation of a mutant vector v and then followed by the arithmetic crossover according to (5). If we decide for the latter explanation and substitute y from (3) instead of v into (5), it results in generating the trial vector according to

$$y = x_i + U_1\,U_2\,(r_1 - x_i) + U_2\,F\,(r_2 - r_3), \tag{6}$$

where U_1 and U_2 are independent random variables uniformly distributed in $[0, 1]$, U_1 corresponding to (1) and U_2 corresponding to (5). However, the expected value of the multiplicative coefficient at $(r_1 - x_i)$ is $E(U_1\,U_2) = E(U_1)\,E(U_2) = 1/4$ and the distribution of $U_1\,U_2$ has positive skewness, which means that smaller values of the multiplicative coefficient are more frequent. It results in generating the trial point y preferably in smaller distance from x_i compared to (3). This version of *composite DE* is denoted by *CoDE0* hereafter.

The uncertainty about the correct form of *composite DE* algorithm was consulted via e-mail with Y. Wang, the first author of [12]. From his response follows that no other crossover in current-to-rand/1 strategy is applied and the (5) is just to explain why the current-to-rand/1 strategy is rotation-invariant. He also wrote, that the current-to-rand/1 strategy can be considered as two-step procedure:

1) generation of the mutant vector v by rand/1 mutation (1),

2) arithmetic crossover according to (5).

However, then we obtain the rule for generating the trial point as it follows:

$$y = x_i + U\,(r_1 - x_i) + U\,F\,(r_2 - r_3). \tag{7}$$

It is obvious that such explanation of the current-to-rand/1 strategy is not quite correct, (7) differs from (3) in the multiplicative coefficient at the second difference of the vectors. Meanwhile after a very careful reading of [12], a short passage hidden in the text was found in the paper [12]: "*In order to further improve the search ability of the rand/2/bin strategy, the first scaling factor F in the rand/2 mutation operator is randomly chosen from 0 to 1 in this paper*". It means that version of *composite DE* algorithm described in the paper [12] uses the rand/2 mutation in the form as follows:

$$v = r_1 + U\,(r_2 - r_3) + F\,(r_4 - r_5)\,, \tag{8}$$

and the current-to-rand/1 strategy generates the trial vector just according to (3). This "classic" version of composite DE which follows the description in [12] is labeled by *CoDE1* hereafter.

Through the inspection of Matlab source text of the *composite DE* algorithm and comments therein downloaded from Q. Zhang's home page[1] it was found that the current-to-rand/1 strategy is implemented in a slightly different way in the *composite DE* algorithm which is tested in the paper [12]. The points r_1, r_2, and r_3 for the mutation are chosen as a sample with replications and the current point x_i is not excluded from the sampling. They need not be mutually distinct. This approach seems to be strange, but the comment in the source code explains the reason: "*We found that using the following mechanism to choose the indices for mutation can improve the performance to certain degree*". This variant using rand/2 according to (8) and modified current-to-rand/1 strategy with replications described above is marked by *CoDE2* hereafter.

The last variant of *composite DE* for experimental comparison. labeled by *CoDE3* hereafter, was implemented with the improved rand/2 strategy according to (8) and the current-to-rand/1 strategy according to

$$y = x_i + U_1\,(r_1 - x_i) + U_2 F\,(r_2 - r_3), \tag{9}$$

where U_1 and U_2 are independent random variables uniformly distributed in $[0,\,1]$. Comparing with the current-to-rand/1 strategy (6) used in efficient *CoDE1* variant, the first scaling coefficient is again randomly distributed but uniformly with $E(U_1) = 1/2$ and the second scaling coefficient is randomized in the same way as in (6).

[1] `http://dces.essex.ac.uk/staff/qzhang/`

3 Experiments

Six well-known scalable test functions [7,9] were used as a benchmark, namely first DeJong (sphere model), Ackley, Griewank, Rastrigin, Rosenbrock, and Schwefel function. The first four of the test functions in their non-shifted form have the global minimum point in the center of domain S, $x^* = (0,0,\dots,0)$, which makes the search of the solution easier for many stochastic algorithms. That is why they were used in their shifted version. The shifted function is evaluated at the point $z = x - o$, $o \in S$, $o \neq (0,0,\dots,0)$. The shift o is generated randomly from uniform D-dimensional distribution before each run. The solution of the global minimization problem is then $x^* = o$ for all the shifted functions. The definition of the test functions and the range of search domains are presented in [11]. The test functions names or their self-explaining abbreviations are used as labels when reporting the results.

The six functions with the problem dimension of $D = 30$ were used as a benchmark in experimental comparison of tested DE variants. One hundred of independent runs was executed for each test problem and each *composite DE* variant in comparison. Population size was set to $NP = 30$ for all the problems, i.e. the same as in [12].

The minimum function value (f_{min}) in the final population and the number of objective function evaluations (*nfe*) needed for the search was recorded in each run. The run is terminated if the difference of function values in the current population is very small or if the number of objective function evaluations was over the given limit. Small difference of function values in the current population indicates that the points of the population are aggregated in a very small part of the search space and the population lost the ability to change its place considerably. The given maximum allowed number of objective function evaluations expresses our willingness to wait for the results. Such form of stopping condition is appropriate either for the benchmark tests or for the real-world applications. The run is finished if the following condition is reached:

$$f_{max} - f_{min} < 1 \times 10^{-6} \quad OR \quad nfe \geq 2 \times 10^4 \times D, \tag{10}$$

where f_{max} and f_{min} are maximum and minimum function values in the current generation, respectively. The same stopping condition is used in all the experiments.

The solution of the problem found by a DE variant was considered acceptable if $f_{min} - f(x^*) < 1 \times 10^{-4}$. If an acceptable solution is found in a run, the number of objective function evaluations needed for its finding was also returned from the run. This number of the function evaluations is denoted by *nfe_near* in results.

The reliability rate R of the search is assessed by the count of acceptable solutions obtained in 100 runs. The number of objective function evaluations (*nfe*) and the reliability rate R are fundamental experimental characteristics of the efficiency of the search.

4 Results

Two basic characteristics are needed to assess algorithm's performance. The first one is the number of function evaluations (*nfe*) needed for the termination of the search. The second one is the reliability rate of the search (R) expressed by the number of runs giving an acceptable solution out of total 100 runs. The values of these characteristics are shown in Table 1 for all the tested *composite DE* variants and benchmark problems. The algorithms except *CoDE3* perform with high reliability, $R \geq 98$. The reliability of *CoDE3* is much lower in three benchmark problems, even no acceptable solution was found in Rosenbrock problem. The least values of *nfe* for each problem are underlined, only the the variants with $R \geq 98$ are taken into account into this *nfe* comparison.

Table 1. Average number of function evaluations and values of reliability rate of *composite DE* variants for all the benchmark problems in experimental tests

	Ackley		DeJong1		Griewank		Rastrigin		Rosenbrock		Schwefel	
Alg	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*	*nfe*	*R*
CoDE	189764	100	98703	100	152818	100	207422	100	507140	100	165572	100
CoDE0	$\underline{46002}$	100	23920	100	$\underline{35018}$	100	131766	100	$\underline{259676}$	98	73691	100
CoDE1	162412	100	85365	100	132715	100	198209	100	467466	100	144816	100
CoDE2	122313	100	65035	100	100560	100	169936	100	322318	100	117908	100
CoDE3	27113	23	$\underline{20296}$	100	26779	57	$\underline{121764}$	99	594310	0	$\underline{62238}$	100

Computational costs of the algorithms (expressed by *nfe*) are also compared visually using the boxplots in Figure 1. The same scale of vertical axis is used for all the problems in order to make inter-problem comparison easier. It is evident from the Figure 1 that the variants differ in their computational costs substantially. It was also confirmed statistically by one-way analysis of variance (ANOVA) carried out for the variants with almost full reliability of the search, i.e. *CoDE3* was excluded from ANOVA as the variant with low reliability in three out of six benchmark problems. ANOVA tests rejected the null hypotheses on the equivalence of expected *nfe* values in all the test problems. Tukey-Kramer multiple comparison reveals the significant differences among the variants. In all the problems, the increasing sequence of variants with respect to *nfe* is the same:

$$\mathit{CoDE0} \prec \mathit{CoDE2} \prec \mathit{CoDE1} \prec \mathit{CoDE}$$

with significant difference between each consequent pairs of variants at 5 % significance level.

Another simple characteristic of the search process is the success rate. The success rate is defined as the percentage of iterations, when the trial point is better than current point of the population (condition in Line 7 in Algorithm 1 is satisfied) with respect to the total number of function evaluations *nfe*. The values of the success rate are shown in Table 2. Higher success rate means that the results of previous process are more exploited. However, too high success rate causes the suppression of exploration, which can result in premature convergence. High success rate of *CoDE3* likely causes its bad reliability in solving Rosenbrock problem.

The average number of the function evaluations necessary to find an acceptable solution gives us a view to the convergence of algorithms in the early and the last stage

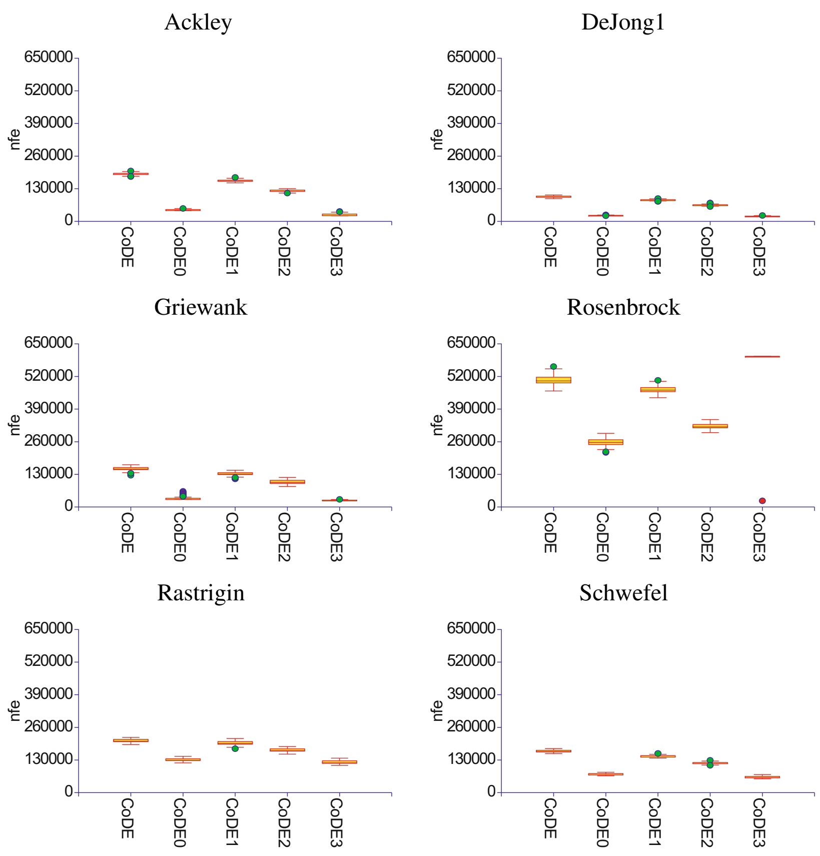

Fig. 1. Comparison of computational costs expressed by *nfe*

of the search. The average values given as the percentage of *nfe* are shown in Table 3. The number of function evaluations needed to find an acceptable solution is specific rather to the problem than variant.

Statistical comparison of the success rates of the DE strategies used in the search process is helpful for the explanation of the different performance of various *composite DE* variants in the experimental tests. The success of a given strategy means that the strategy generates the trial vector with the minimum function value among three strategies and simultaneously the function value of the trial point is less than $f(x_i)$. The counts of the success found by each strategy were recorded in the experiments, the values of counts in 5 by 3 contingency tables are evaluated cumulatively in 100 runs for each benchmark problem separately. The contingency tables were analyzed by the χ^2 tests. Null hypotheses (independence of *composite DE* variant and kind of strategy) were rejected in all the problems at significance level of $\alpha = 0.001$. The sources of dependence were assessed by standardized residuals. The standardized residuals are

Table 2. Values of success rate of *composite DE* variants in all the benchmark problems

	Success Rate (% of *nfe*)						
	ack	dej	gri	ras	ros	schw	Avg
CoDE	5.4	6.0	5.3	3.6	3.3	4.8	4.7
CoDE0	16.1	18.5	16.7	4.7	18.5	8.2	13.8
CoDE1	6.2	6.9	6.0	3.8	3.7	5.4	5.3
CoDE2	7.0	7.7	6.8	3.9	5.5	5.8	6.1
CoDE3	20.0	22.2	21.6	4.7	22.4	9.4	16.7

Table 3. Average number of the function evaluations needed to find an acceptable solution as percentage of *nfe*

	nfe_near (% of *nfe*)					
	ack	dej	gri	ras	ros	schw
CoDE	77	72	82	87	89	83
CoDE0	79	75	82	95	95	92
CoDE1	77	72	82	88	88	84
CoDE2	77	72	82	89	92	85
CoDE3	80	78	84	96	–	93
Average	78	74	82	91	91	87

asymptotically normally distributed, $N(0,1)$. The patterns of sources of dependence are shown in Table 4 for each problem. The significance of the standardized residual is marked schematically by sign symbols in the appropriate cell of contingency table. The symbol "$+++$" denotes significantly positive value of the standardized residual (i.e. this strategy is successful more frequently than expected under independence), the symbol "$---$" denotes significantly lower frequency of success at the level of significance $\alpha = 0.001$, the symbol "$++$" means significant positive value at $\alpha = 0.01$, If the value of the standardized residual is not significant, the cell is empty.

Table 4 shows almost the same patterns of strategy's success for *CoDE0* and *CoDE3* variants. In the both variants, rand/2 strategy has significantly higher success than the other strategies in all the test problems. Notice, the rand/2 strategy in *CoDE0* differs from the rand/2 strategy applied in *CoDE3*. *CoDE0* and *CoDE3* were the most efficient variants in all the test problems except Rosenbrock, see Figure 1. The patterns of *CoDE0* and *CoDE3* are different from the patterns of the other variants. Despite the similarity of success patterns, there is a big difference in the performance of *CoDE0* and *CoDE3* variants. While the reliability rate of *CoDE0* is almost 100, the reliability of *CoDE3* is much less in three test problems. It seems that *CoDE3* is too greedy, likely due to the using the version of current-to-rand/1 according to (9).

Table 4. Comparison of success rate of strategies – significance of standardized residuals

	rand1	rand2	currtorand	rand1	rand2	currtorand
		ackley			dejong1	
CoDE	+++	−−−	+++	+++	−−−	+++
CoDE0	−−−	+++	−−−	−−−	+++	−−−
CoDE1	+++	−−−	+++	+++	−−−	+++
CoDE2	++	−−−	+++	++	−−−	++
CoDE3	−−−	+++	−−−	−−−	+++	−−−
		griewank			rosenbrock	
CoDE	+++	−−−	++	+++	−−−	+++
CoDE0	−−−	+++	−−−	−−−	+++	−−−
CoDE1	+++	−−−	+++	+++	−−−	+++
CoDE2	+++	−−−	+++	+++	−−−	+++
CoDE3	−−−	+++	−−−	−−−	+++	−−−
		rastrigin			schwefel	
CoDE	+++	−−−	+++	+++	−−−	+++
CoDE0	−−−	+++	−−−	−−−	+++	−−−
CoDE1	+++	−−−	+++	+++	−−−	+++
CoDE2		−−−	++	++	−−−	++
CoDE3	−−−	+++	−−−	−−−	+++	−−−

5 Conclusion

The experimental comparison of *composite* DE variants verifies significantly different performance. The results impeaches the role of rationality in the design of well-performing stochastic global optimizers. Best performing *CoDE0* variant was proposed due to misunderstanding of *composite DE* description in [12]. Second best *CoDE2* variant uses a strange modification of the current-to-rand/1 strategy (random points are sampled with replications) described only in the source code of the algorithm. Any rational explanation of using the sampling with replications in this strategy can be hardly found. However, this variant is significantly more efficient than the classic *CoDE1* variant corresponding to the description of algorithm in [12]. The worst performing variant *CoDE3* was designed taking into account all the previous results with a rational effort to increase the algorithm's efficiency. This variant is very efficient in some problems but also appears unreliable in some other problems. However, the best performing *CoDE0* variant is a promising modification of composite DE algorithm and it should be tested in hard benchmark problems, e.g. those defined in [10] in order to verify its superiority in more thorough and convincing way. The current-to-rand/1 strategy and the choice of distribution in randomizing its multiplicative coefficients is another topic for next research.

Acknowledgement. This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and partly supported by University of Ostrava, the project SGS13/PrF/2012. The autor also thanks to Y. Wang for his e-mail discussion on the composite DE algorithm.

References

1. Brest, J., Greiner, S., Boškovič, 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, 646–657 (2006)
2. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a neighborhood-based mutation operator. IEEE Transactions on Evolutionary Computation 13, 526–553 (2009)
3. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15, 27–54 (2011)
4. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing 11, 1679–1696 (2011)
5. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33, 61–106 (2010)
6. Price, K.V.: An introduction to differential evolution. In: New Ideas in Optimization, pp. 293–298. McGraw-Hill, London (1999)
7. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer (2005)
8. Qin, A., Huang, V., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13, 398–417 (2009)
9. Storn, R., Price, K.V.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization 11, 341–359 (1997)
10. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization (2005), `http://www.ntu.edu.sg/home/epnsugan/`
11. Tvrdík, J.: A comparison of control-parameter-free algorithms for single-objective optimization. In: Matousek, R. (ed.) 16th International Conference on Soft Computing Mendel 2010, pp. 71–77 (2010)
12. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation 15, 55–66 (2011)
13. Zhang, J., Sanderson, A.C.: JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13, 945–958 (2009)