Reevaluating Exponential Crossover in Differential Evolution

Ryoji Tanabe and Alex Fukunaga

Graduate School of Arts and Sciences, The University of Tokyo

Abstract. Exponential crossover in Differential Evolution (DE), which is similar to 1-point crossover in genetic algorithms, continues to be used today as a default crossover operator for DE. We demonstrate that exponential crossover exploits an unnatural feature of some widely used synthetic benchmarks such as the Rosenbrock function – dependencies between adjacent variables. We show that for standard DE as well as state-of-the-art adaptive DE, exponential crossover performs quite poorly on benchmarks without this artificial feature. We also show that shuffled exponential crossover, which removes this kind of search bias, significantly outperforms exponential crossover.

1 Introduction

Differential Evolution (DE) is an Evolutionary Algorithm (EA) that was primarily designed for real parameter optimization problems [16], and has been applied to many practical problems [14]. A DE population is represented as a set of real parameter vectors $\boldsymbol{x}_i = (x_1, ..., x_D)$, $i = 1, ..., N$, where D is the dimensionality of the target problem, and N is the population size. In each generation t, a mutant vector $\boldsymbol{v}_{i,t}$ is generated from an existing population member $\boldsymbol{x}_{i,t}$ by applying some mutation strategy. Then, the mutant vector $\boldsymbol{v}_{i,t}$, is crossed with the parent $\boldsymbol{x}_{i,t}$ in order to generate trial vector $\boldsymbol{u}_{i,t}$. After all of the trial vectors $\boldsymbol{u}_{i,t}, 0 \leq i \leq N$ have been generated, each individual $\boldsymbol{x}_{i,t}$ is compared with its corresponding trial vector $\boldsymbol{u}_{i,t}$, keeping the better vector in the population.

Algorithm 1: exponential crossover

$\boldsymbol{u}_{i,t} = \boldsymbol{x}_{i,t}$, j is randomly selected from $[1, D]$, $L = 1$;
repeat
  $u_{j,i,t} = v_{j,i,t}$, $j = (j + 1)$ modulo D, $L = L + 1$;
until rand$[0, 1) < CR$ *and* $L < D$;

The two most common type of crossover in DE are binomial crossover, analogous to uniform crossover in GA's, and exponential crossover, analogous to 1 or 2 point crossover in GA's [16]. Binomial crossover is implemented as follows: For each j $(j = 1, ..., D)$, if rand$[0, 1) \leq CR$ or $j = j_{rand}$, $u_{j,i,t} = v_{j,i,t}$.

Otherwise, $u_{j,i,t} = x_{j,i,t}$, where rand$[0, 1)$ denotes a uniformly selected random number from $[0, 1)$, and j_{rand} is a decision variable index which is uniformly randomly selected from $[1, D]$. $CR \in [0, 1]$ is the crossover rate. Exponential crossover is implemented as shown in Algorithm 1. The choice of crossover type determines the distribution of the number of variables that are inherited by children (trial vectors in DE terminology), as well as the contiguousness of the inherited variables. Although binomial crossover appears to be more frequently used in state-of-the-art DEs [2, 18, 23], a number of recent papers have reported successful usage of exponential crossover [3, 7, 9, 13, 24, 25].

This paper questions the continued usage of exponential crossover in DE. It has been long known in the GA community that traditional 1-point/2-point crossover introduces significant positional biases – interactions between genes that are positionally far from each other in a genome tend to be disrupted, while interactions between genes that are close to each other tend not to be disrupted [4]. This positional bias tends to result in undesirable search behavior in real-world problems, and classical 1-point/2-point crossover, while still introduced in textbooks, tends not to be used by experienced GA practitioners. Why then, is exponential crossover, which is quite similar to 1 or 2 point crossover, still used by the DE community?

We argue that exponential crossover in DE has been overrated because it successfully exploits unnatural dependencies between adjacent variables in widely used synthetic benchmarks. We show that if the benchmarks are altered to eliminate these unnatural dependencies, then exponential crossover performs very poorly. We also evaluate shuffled exponential crossover (SEC), a method for implementing exponential crossover without relying on arbitrary dependencies between adjacent variables, which was briefly suggested in [14] but to our knowledge has never been evaluated. Although exponential crossover has been one of the recommended crossover methods since the introduction of DE in 1995 [16], we believe that the use of exponential crossover needs to be carefully reconsidered in light of our experimental results.

2 Adjacent Functions: A Common But Unnatural Class of Benchmarks

In black-box optimization, synthetic benchmarks (e.g., the 13 classical functions [22], CEC benchmarks [17, 20], GECCO BBOB [5], and SOCO benchmarks [8]) are often used by EA researchers as proxies for performance on real-world problem instances. Although synthetic benchmark suites are designed in order to include representatives of many class of real-world problems (e.g. unimodal/multimodal/separable/nonseparable), previous work has pointed out that benchmark suites can some pitfalls in using synthetic benchmarks to evaluate EA's [11, 15, 20, 21]. One specific issue is the presences of exploitable problem characteristics that do not arise in real-world problems [11].

One such "exploitable problem characteristic" found in some widely used, nonseparable synthetic benchmarks is *unnatural dependencies between adjacent*

Table 1: Nonseparable Benchmark Functions

Name	Definitions	Search Range	Properties
Rosenbrock	$f(\boldsymbol{z}) = \sum_{i=1}^{D-1}(100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2)$	$[-30, 30]^D$	Multimodal
Schwefels 1.2	$f(\boldsymbol{z}) = \sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2$	$[-100, 100]^D$	Unimodal
Block-rotated Ellipsoid [1]	$f(\boldsymbol{z}) = \sum_{i=1}^{D-1}\sum_{j=1}^{2}\left(\alpha^{j-1}(\boldsymbol{R}_i \cdot (z_i, z_{i+1}))\right)^2$	$[-5, 5]^D$	Unimodal

variables – variables interacting (exclusively) with other variables that happen to have similar variable indices. For example, in the Rosenbrock function, a canonical, nonseparable function that has been widely used as an EA benchmark, each term in the summation depends on adjacent variables z_i and z_{i+1} (Table 1). However, there is no particular reason that adjacent variables should have such dependencies in real-world, black-box optimization problems, and such dependencies are an artifact of synthetic benchmarks.[1]

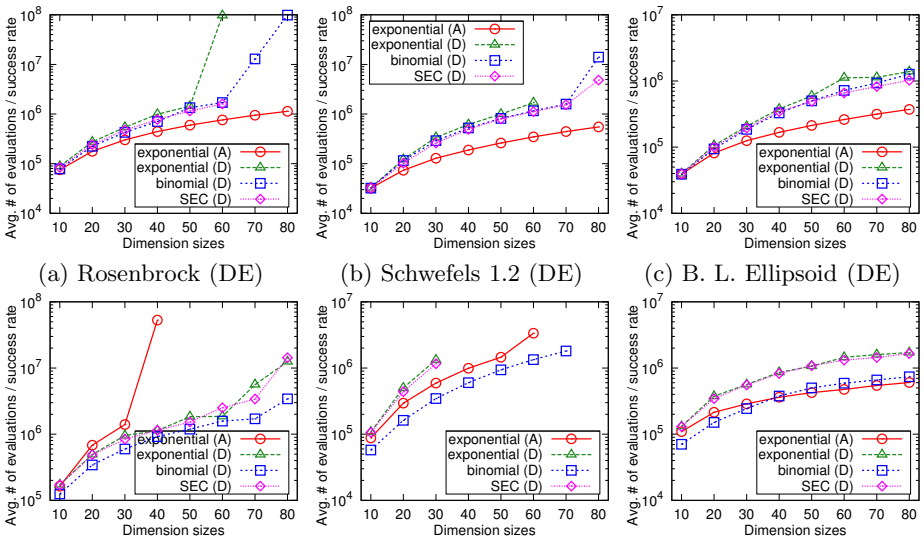
This unnatural problem structure can be easily eliminated using the randomization procedure described in [20]. First, the permutation vector $\boldsymbol{P}$ ($\boldsymbol{P} = P_1, ..., P_D$) is initialized to a random permutation at the beginning of the DE run. During the DE run, whenever a trial vector $\boldsymbol{x}$ is evaluated, we permute $\boldsymbol{x}$ using $\boldsymbol{P}$, resulting in the permuted trial vector $\boldsymbol{x'} = (x_{P_1}, ..., x_{P_D})$. Then, $\boldsymbol{x'}$ is evaluated using the evaluation function, and the result is the fitness score for trial vector $\boldsymbol{x}$. This permutation effectively eliminates arbitrary dependencies between variables with consecutive in the trial vector $\boldsymbol{x}$ – the dependencies are now between variables with indices that are consecutive in $\boldsymbol{x'}$, but exponential crossover, which operates on $\boldsymbol{x}$, can not exploit the consecutiveness in $\boldsymbol{x'}$.

2.1 Exponential Crossover on Adjacent/Distributed Functions

We evaluate exponential crossover on (1) functions with dependencies between lexicographically adjacent variables, i.e., standard versions of widely used synthetic benchmarks, and (2) modified versions of functions in (1) where the permutation method described above is used to randomize the variable dependencies and eliminate the adjacent dependency structure. Following [4], we call functions of the former class *adjacent functions*, and functions of the latter class *distributed functions*.

We used the 3 nonseparable, adjacent functions in Table 1. The Rosenbrock and Schwefels 1.2 functions are well-known, classical benchmark functions that have been widely used to evaluate EA's [22]. The Block-rotated Ellipsoid [1] is a partially separable function designed to only have dependencies between z_i and

[1] Of course, there are real-world problems that can be represented in such a way as to have dependencies between adjacent variables [1] – we are merely arguing that these are not representative of *black-box* optimization problems.



(a) Rosenbrock (DE) (b) Schwefels 1.2 (DE) (c) B. L. Ellipsoid (DE)

(d) Rosenbrock (jDE) (e) Schwefels 1.2 (jDE) (f) B. L. Ellipsoid (jDE)

Fig. 1: Evaluation of various crossover operators (standard DE and jDE). (A) and (D) stand "Adjacent" and "Distributed" respectively. The horizontal axis represents the dimensionality D, and the vertical axis represents the average fitness evaluations (for successful runs) divided by success rate.

z_{i+1}. Here, $\boldsymbol{z} = (y_{P_1}, ..., y_{P_D})$, $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{o}$, and for each function, the location of the global optimum has been shifted by offset $\boldsymbol{o}$ ($\boldsymbol{o} = o_1, ..., o_D$), where each component of $\boldsymbol{o}$ is a uniformly generated random offset. For adjacent functions, the permutation vector $\boldsymbol{P} = (1, ..., D)$, and for distributed functions, $\boldsymbol{P}$ is a randomly generated ordering such as $\boldsymbol{P} = (6, 1, 3, ...)$. The 2×2 rotation matrix $\boldsymbol{R}_i$ is uniformly generated according to the method of [15] and $\alpha =$ 1e+6.

We studied problems with $10 - 80$ dimensions. Each DE run continues until either (1) the difference between the best-so-far and the optimal solution $\leq$ 1e-8, in which case we treat the run as a "success", or (2) the # of objective function calls exceeds 2.0×10^6, in which case the run is treated as a "failure". On each problem, each algorithm is executed 50 times. Following [6], our evaluation metric is the average # of fitness evaluations in successful runs divided by the # of successes. Small values of this metric indicate a fast and stable search.

We use standard DE [16], as well as the state-of-the-art adaptive DE variants jDE [2], JADE [23], and SHADE [18].[2] The standard DE used a population size of 100 and $F = 0.5$, and the most commonly used, and rand/1 mutation strategy – this is a standard configuration in the DE literature [2, 23]. In

[2] jDE [2], JADE [23], SHADE [18] were originally designed to use binomial crossover; in order to evaluate exponential crossover on state-of-the-art DE's, we modified these to use exponential crossover.

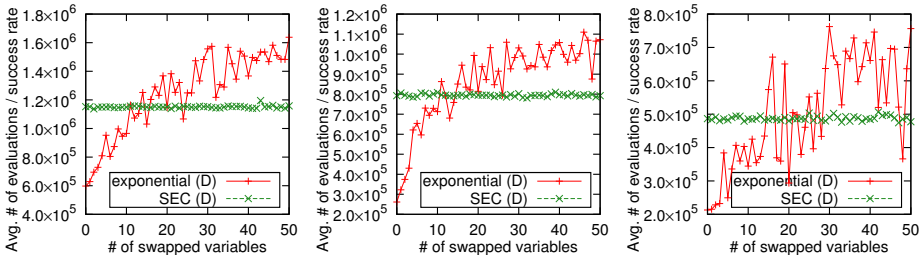
addition to exponential crossover, we also ran the experiments with binomial crossover for comparison. On the standard DE, for each crossover method, for each benchmark function, and for each dimensionality (D), we use the value of $CR \in \{0.90, 0.91, ..., 0.99\}$ that yields the best performance according to the performance metric defined above. For jDE, JADE, and SHADE, which automatically adjusts CR, we use the control parameters recommended in the original papers on these adaptive methods [2, 18, 23].

The results for standard DE and jDE are shown in Figure 1. Detailed results for JADE and SHADE are in the supplemental material [19] due to space constraints, but the SHADE and JADE results are qualitatively similar to the jDE results. "Shuffled exponential crossover (SEC)" is explained in Section 3. For cases where all runs failed (success rate for 50 trials = 0), then the data is not shown. Since binomial crossover behaves almost identically for adjacent and distributed functions, only the distributed function results are shown.

Figure 1 shows that exponential crossover performs much better on adjacent functions compared to distributed functions. The performance gap increase as the dimensionality increases. For standard DE, exponential crossover outperforms binomial crossover on all adjacent functions, for all dimensionalities. In stark contrast, on the distributed functions, the performance of exponential crossover drops significantly – for all the distributed functions, for all D, exponential crossover performs worse than binomial crossover. In particular, on the distributed-Rosenbrock and distributed-Schwefels 1.2 functions, for $D \geq 70$ dimensions, exponential crossover fails on every single run. The results are similar for jDE. Aside from the results on the Rosenbrock function (Figure 1(d)), the performance of exponential crossover on distributed functions is clearly worse than on the adjacent functions.

These results clearly show that the performance of exponential crossover on nonseparable function benchmarks such as Rosenbrock and Schwefels 1.2 depends on an arbitrary feature of these synthetic benchmarks – variable dependencies between adjacent variables. Given essentially the same, nonseparable functions without this arbitrary structure, exponential crossover performs much worse (significantly worse than binomial crossover). Functions such as Rosenbrock and Schwefels 1.2 have been part of benchmark suites used by to evaluate DE since the original paper introducing DE [16], and continue to be used today [3, 7, 9, 13, 24, 25]. As a consequence, exponential crossover has been inaccurately overrated as a DE crossover operator for black-box optimization.

How fragile is exponential crossover to perturbations in the variable index order? Instead of completely randomizing the variable index order, we investigate the effect of gradually decreasing the dependency between adjacent variable indices by applying $n = 1, ..., D$ randomized swaps to the variable indices. As n increases, the number of dependencies between adjacent variables decreases. We ran standard DE on 50-dimensional problems (same setup as in the previous experiment). For each problem, we used the CR value that performed best for each n. Figure 2 shows that for all of the functions, exponential crossover performs best when $n = 0$, and rapidly degrades as n increases, i.e., exponential



(a) Rosenbrock (b) Schwefels 1.2 (c) B. L. Ellipsoid

Fig. 2: Effect of gradually decreasing dependencies between adjacent variables for standard DE using exponential crossover and SEC on 50-dimensional problems. Horizontal axis = # of randomized swaps ($n = 1, ..., D$); Vertical axis = average fitness evaluations (for successful runs) divided by success rate.

crossover is quite fragile with respect to perturbations in dependencies between adjacent variables.

3 Shuffled Exponential Crossover

The experiments in Section 2 showed that exponential crossover performs poorly on distributed functions. To alleviate this problem, we evaluate shuffled exponential crossover. Caruana et al observed that 1-point crossover in GA's introduced a very strong search bias in that variables located close to each other in a linear genome (i.e., variable indices with similar lexicographic values) tended to be inherited by the same child, while variables located far apart tend to be inherited by different children [4]. To eliminate this bias, they proposed shuffle crossover [4]: First, the variable indices of the parents are randomly shuffled (the same shuffle is applied to both parents). Next, standard 1-point crossover is applied to the shuffled genomes. Finally, the indices are restored to their pre-shuffled states.[3]

Price et al note that exponential crossover in DE is subject to the same bias as 1-point crossover in GA's, and suggested that the shuffling mechanism from shuffle crossover can be added to exponential crossover in order to alleviate this problem [14]. In this paper, we call this *shuffled exponential crossover* (SEC). The algorithm is shown in Algorithm 2. Recently, Lin et al evaluated a mechanism called non-consecutive exponential crossover which is in fact, the same as SEC [12]. However, they do not analyze the effect of this mechanism in the context of dependencies among variables as we do. In fact, [12] claims "in non-separable ridge functions (Rosenbrock and Schwefels 1.2 function), differential evolution

[3] While shuffle crossover is similar to uniform crossover in that the genes that are exchanged are dispersed throughout the genome, the # of genomes that are exchanged has a very different probability distribution, and there is a different search bias.

algorithms with consecutive crossover are more reliable than those with non-consecutive crossover" (this is clearly contradicted by our result above).

Algorithm 2: Shuffled Exponential Crossover (SEC)

$\boldsymbol{u}_{i,t} = \boldsymbol{x}_{i,t}$, $k = 1$, $\boldsymbol{S}(= s_1, ..., s_D)$ is randomly shuffled permutation $\{1, ..., D\}$;
repeat
  $j = s_k$, $u_{j,i,t} = v_{j,i,t}$, $k = k + 1$;
until rand$[0, 1) < CR$ *and* $k < D$;

Figure 1 shows the results of SEC for standard DE as well as jDE for the same problems/settings as in Section 2.1.[4] Unlike exponential crossover (but, similar to binomial crossover), the performance of SEC is unaffected by whether the test function is adjacent or distributed; thus, only the distributed function results are shown. Figure 2 also shows the results for SEC with best CR value when $n = 0$ for the same problems/settings as in Section 2.1. The performance of SEC is clearly shown to be independent of n.

Figure 1 shows that overall, SEC slightly outperforms exponential crossover on the distributed functions. However, on the distributed Rosenbrock function (60 dimensions), SEC significantly outperforms exponential crossover. Also, on the distributed-Schwefels 1.2 function, SEC is competitive with binomial crossover, even on $70 \geq$ dimensional problems where exponential crossover failed on every single run. These results show that while exponential crossover depends on the ordering of variables (i.e., whether the function is adjacent or distributed), SEC, as expected, does not depend on the index positions of the variables and yields a much more stable search performance as a result.

4 Is Exponential Crossover Overrated?

Past evaluations of DE crossover operators on standard benchmark test suites need to be reconsidered in light of our analysis of the interaction between adjacent functions and exponential crossover. Functions with dependencies between adjacent variables (e.g., Rosenbrock, Schaffer $F7$, Whitley's composite functions [21]) are included in widely used benchmark suites. Table 2 shows the number of adjacent functions out of the total number of functions in the typical benchmarks. The larger the number of adjacent functions in the benchmark suite, the more favorable the suite is for methods that exploit the adjacent structure, such as DE with exponential crossover. In particular, note that 10 out of 19 functions in the recent Soft Computing Journal (SOCO) benchmarks [8] are adjacent functions, making it particularly vulnerable to exploitation by exponential crossover. *In fact, all 7 of the DE algorithms submitted to the SOCO special issue evaluation used exponential crossover.* It would seem that due to the presence of

[4] Results for JADE and SHADE are in the supplemental material [19].

Table 2: # of Adjacent Functions in Standard Benchmark Suites

Benchmarks	# of adjacent functions	Function
13 classical [22]	2 / 13	F_3, F_5
CEC 2005 [17]	4 / 25	F_2, F_4, F_6, F_{13}
CEC 2010 [20]	2 / 20	F_{19}, F_{20}
GECCO BBOB [5]	1 / 24	F_8
SOCO benchmarks [8]	10 / 19	$F_3, F_8, F_9, F_{11}, F_{12}, F_{13}, F_{14}, F_{16}, F_{17}, F_{18}$

Table 3: SEC vs. binomial and exponential crossover on the CEC2014 benchmark functions [10] for $D = 50$ (Wilcoxon rank-sum test significance threshold $p < 0.05$).

		exponential				binomial			
		DE	jDE	JADE	SHADE	DE	jDE	JADE	SHADE
vs. SEC	# of better	1	0	0	0	12	18	19	13
Wilcoxon rank-sum	# of worse	8	2	3	5	9	5	7	6
(significance: $p < 0.05$)	# of no sig.	21	28	27	25	9	7	4	11

adjacent functions, exponential crossover has been overrated in previous evaluations (assuming that benchmark suites are supposed to model true "black-box" scenarios where there is no a priori reason to believe that adjacent variables have dependencies).

How do DE crossover operators compare on a benchmark set that does not contain any adjacent functions? The recently proposed CEC2014 benchmark set [10] consisting of 30 problems, does not include any adjacent functions. We evaluated SEC vs exponential crossover vs binomial crossover on DE, jDE, JADE, and SHADE on the CEC2014 benchmarks (in 10, 30, and 50 dimensions), following the evaluation methodology specified in the CEC2014 benchmark competition [10]. The overall results for $D = 50$ dimensions are shown in Table 3.[5] As shown in Table 3, for a diverse benchmark set, SEC outperforms exponential crossover for all DE algorithms. Binomial crossover performs best for all DE algorithms.

5 Conclusion

This paper showed that exponential crossover, one of the standard crossover methods in DE, implicitly exploits an unnatural structure found in some syn-

[5] The results for $D = 10, 30$ are in the supplemental data [19].

thetic benchmark problems, including some widely used, nonseparable functions (Rosenbrock and Schwefels 1.2), where there are strong dependencies between variables with consecutive indices. We showed that exponential crossover performs significantly worse if we slightly perturbing these classical benchmarks to remove these arbitrary, lexicographic dependencies, i.e., after artificial dependencies between adjacent variables are removed, the "true" performance of exponential crossover appears to be significantly worse than previously believed. We believe that for synthetic benchmarks [5, 8, 17, 20, 22], the performance of exponential crossover has been overrated [3, 7, 9, 13, 24, 25]. Thus, the suitability of exponential crossover should be reevaluated in light of our results. We showed that SEC, which does not implicitly assume sequential dependencies between variables and does not have the same search bias as exponential crossover, significantly outperforms exponential crossover overall, and is competitive with binomial crossover. While SEC was suggested by [14], and also proposed by [12], our work is the first to identify the specific weaknesses described above for exponential crossover and show that shuffling results in improved overall performance. Although there is still no clear criteria to determine whether binomial crossover or SEC should be used for a particular problem (a direction for future work), we believe that we have presented sufficient evidence to suggest that plain, exponential crossover should no longer be considered as an appropriate, default crossover operator for DE. SEC (or binomial crossover) should be used instead of exponential crossover, unless there is some prior knowledge that there are dependencies between consecutive variables.

As we showed for exponential crossover in DE, algorithms that (intentionally or unintentionally) exploit this dependency can appear to perform much better than they would actually perform on real-world problems without this artificial structure. As shown in Section 2, randomizing the lexicographic positions of the variables for all benchmark functions, as suggested by [20] is a simple method for avoiding this benchmarking pitfall, and we believe that black-box benchmark suites should apply this randomization to avoid unintentionally biasing the evaluation results.

References

1. Z. Bouzarkouna, A. Auger, and D. Y. Ding. Local-meta-model CMA-ES for partially separable functions. In *GECCO*, pages 869–876, 2011.
2. J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Tran. Evol. Comput.*, 10(6):646–657, 2006.
3. J. Brest and M. S. Maučec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Comput.*, 15(11):2157–2174, 2011.
4. R. Caruana, L. J. Eshelman, and J. D. Schaffer. Representation and Hidden Bias II: Eliminating Defining Length Bias in Genetic Search via Shuffle Crossover. In *IJCAI*, pages 750–755, 1989.
5. N. Hansen. GECCO BBOB. http://coco.gforge.inria.fr/doku.php, 2014.

6. N. Hansen and S. Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In *PPSN*, pages 282–291, 2004.
7. F. Herrera, M. Lozano, and D. Molina. Components and parameters of de, real-coded chc, and g-cmaes. Technical report, Univ. of Granada, 2010.
8. F. Herrera, M. Lozano, and D. Molina. Test suite for the spec. iss. of Soft Computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. Technical report, Univ. of Granada, 2010.
9. A. LaTorre, S. Muelas, and J. M. Peña. A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. *Soft Comput.*, 15(11):2187–2199, 2011.
10. J. J. Liang, B. Y. Qu, and P. N. Suganthan. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical report, Zhengzhou Univ. and Nanyang Technological Univ., 2013.
11. J. J. Liang, P. N. Suganthan, and K. Deb. Novel Composition Test Functions for Numerical Global Optimization. In *Swarm Intell. Symp.*, pages 68–75, 2005.
12. C. Lin, A. Qing, and Q. Feng. A comparative study of crossover in differential evolution. *J. Heuristics*, 17(6):675–703, 2011.
13. N. Noman and H. Iba. Accelerating Differential Evolution Using an Adaptive Local Search. *IEEE Tran. Evol. Comput.*, 12(1):107–125, 2008.
14. K. V. Price, R. N. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005.
15. R. Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.
16. R. Storn and K. Price. Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, Berkeley, CA, 1995.
17. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological Univ., 2005.
18. R. Tanabe and A. Fukunaga. Success-History Based Parameter Adaptation for Differential Evolution. In *IEEE CEC*, pages 71–78, 2013.
19. R. Tanabe and A. Fukunaga. Supplemental material. https://sites.google.com/site/tanaberyoji/home/ppsn2014-supplement.pdf, 2014.
20. K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization. Technical report, Univ. of Science and Technology of China, 2010.
21. D. Whitley, K. Mathias, S. Rana, and J. Dzubera. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85:245–276, 1996.
22. X. Yao, Y. Liu, and G. Lin. Evolutionary Programming Made Faster. *IEEE Tran. Evol. Comput.*, 3(2):82–102, 1999.
23. J. Zhang and A. C. Sanderson. JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Tran. Evol. Comput.*, 13(5):945–958, 2009.
24. S. Zhao and P. N. Suganthan. Empirical investigations into the exponential crossover of differential evolutions. *Swarm and Evol. Comput.*, 9:27–36, 2013.
25. S. Zhao, P. N. Suganthan, and S. Das. Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Comput.*, 15(11):2175–2185, 2011.