Island Based Distributed Differential Evolution: An Experimental Study on Hybrid Testbeds

Javier Apolloni, Guillermo Leguizamón
LIDIC - Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950, 5700, Argentina
{javierma,legui}@unsl.edu.ar

José García-Nieto, Enrique Alba
Lenguajes y Ciencias de la Computación
University of Málaga
Campus de Teatinos s/n, 29071, Spain
{jnieto,eat}@lcc.uma.es

Abstract

This paper presents a new distributed Differential Evolution (dDE) algorithm and evaluates it according to the standard procedure set in the special session of continuous optimization of CEC'05. We statistically validate our results in continuous optimization versus several other efficient techniques. Our distributed Differential Evolution is simple and accurate, at the same time amenable, to be applied to a wide variety of problems, especially for noisy and multimodal functions.

Keywords: Distributed Differential Evolution, Non-Parametric Tests, CEC'05 Function Test Suite

1. Introduction

Evolutionary Algorithms [5, 10] are methods inspired by biological processes that perform stochastic search to solve problems modeled as discrete and continuous variables. Among the many EA families we are interested in Differential Evolution (DE). DE was first introduced by Price *et al.* [17] and has been applied on a wide range of problems. The ability to treat non-differentiable and multimodal optimization functions has made DE more and more popular in there last years. The power of DE comes from its simple structure and its inherent high performance.

The temporal complexity of some problems can be handled by regular evolutionary methods, but the exploration of the search space performed by them is also time-consuming. Therefore, parallel models have been very useful tools to improve the performance of such techniques during the search process. DE methods can be easily parallelized in a distributed model, since they are based in the evolution of a population of individuals. This population could be partitioned into small subsets known as islands, each subset evolving independently from each other. Besides, the islands are spatially structured and exchange information among them to hopefully increase the accuracy and efficiency of the resulting algorithm. When run in a parallel computer, the time reduction is an additional advantage [1].

The advance in the hardware technology, along software has allowed to link computers by means of a network to create a powerful tool for low-cost computing. Each computer runs an island with a DE algorithm evolving a set of individuals. The periodical migration of individuals in a given topology leads to a high exploration ability in DE, since the foreign solutions add diversity to the island populations [9, 12, 19]. The performance of a new distributed DE is shown here on a hybrid test suite of functions provided in the special session of continuous optimization of CEC'05. Statistical comparisons with all participant algorithms in this standard set of results show the high accuracy and efficiency of our parallel DE.

The remainder of this paper is organized as follows. In Section 2, we explain briefly the DE algorithm. In Section 3 we focus on the parallel model and the migration policy developed for this work. An experimental study is described in Section 4, where our proposal is compared against a series of algorithms which are the state-of-art in continuous optimization. The paper ends with a brief summary and general conclusion included in Section 5.

2. Differential Evolution: Background

DE is a stochastic population-based algorithm designed to solve optimization problems over continuous domains. The population is a set of individuals (tentative solutions) which evolve simultaneously through the search space of the problem. The individuals are real-value vectors that, combined with others from the population, generate new individuals. The task of generating new individuals is carried out by the differential operators. The mutation is an essential operator to DE since it adds the weighted difference of

two individuals of the population to a third member.

Formally, an individual is a vector $v_g^i = (v_g^i(1), v_g^i(2), \ldots, v_g^i(D))$ where $v_g^i(j) \in \mathbb{R}$ $(1 \leq i \leq N, 1 \leq j \leq D)$ and D is the number of variables, g is the current generation and N is the number of individuals in the population. A *mutant individual* w_{g+1}^i is generated by the following equation (1):

$$w_{g+1}^i = v_g^{r1} + \mu \cdot (v_g^{r2} - v_g^{r3}) \tag{1}$$

where $r1, r2, r3 \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$ are random integers mutually different, and also different from the index i, the mutation constant $\mu > 0$ stands for the amplification of the difference between the individuals v_g^{r2} and v_g^{r3}, and it avoids the stagnation of the search process.

In order to increase even more the diversity in the population, each mutated individual undergoes a crossover operation with the *target individual* v_g^i, by means of which a *trial individual* u_{g+1}^i is generated. We use a uniform crossover also named *binomial crossover* [14] where each position of the trial individual has the same probability of being chosen from the target individual or from the mutant individual. A randomly chosen position is taken from the mutant individual to prevent that the trial individual replicates the target individual.

$$u_{g+1}^i(j) = \begin{cases} w_{g+1}^i(j) & \text{if } r(j) \leq Cr \text{ or } j = j_r, \\ v_g^i(j) & \text{otherwise.} \end{cases} \tag{2}$$

As shown in Equation 2, the crossover operator randomly chooses a uniformly distributed integer value j_r in $[1, \ldots, D]$ and a random real number r in $(0, 1)$, also uniformly distributed for each component j $(j = 1, \ldots, D)$ of the trial individual u_{g+1}^i. Then, the crossover probability Cr and r are compared just like j and j_r. If r is less than or equal than Cr (or j is equal to j_r) then we select the j^{th} element of the mutant individual to be allocated in the j^{th} element of the trial individual u_{g+1}^i. Otherwise, the j^{th} element of the target individual v_g^i becomes the j^{th} element of the trial individual. Finally, a selection operator decides the acceptance of the trial individual for the next generation if and only if it yields a reduction in the value of the evaluation function, as shown by the following equation (3):

$$v_{g+1}^i = \begin{cases} u_{g+1}^i & \text{if } f(u_{g+1}^i) \leq f(v_g^i), \\ v_g^i(j) & \text{otherwise.} \end{cases} \tag{3}$$

The DE is a simple algorithm and has only a few parameters to tune. However, the success of the performance of the algorithm is related to a complex interaction of the parameters, especially μ and Cr. The parameters can be set with a self-adapting technique during the evolutionary process, as shown by Brest *et al.* in [7].

3. Parallel Differential Evolution

In this section we present a version of a parallel DE method. Evolutionary techniques like DE are easily parallelized since the evaluation of each individual of the population is usually an independent task of the algorithms. The parallel computation is expected to improve the performance and to reduce of computational cost of DE [12].

3.1. Parallelization of Differential Evolution

Our work focus on a study of the the optimization of continuous multimodal functions by using a parallel version of DE similar to the suggested by Tasoulis *et al.* [19] . We use a distributed model in islands where the population is partitioned in small groups of individuals. The individuals inside each island evolve independently from the rest of islands, but each island made occasional communication operations with the others islands by exchanging solutions.

The exchange of solutions is determined by the *migration rate*, that defines the number of individuals that are sent to (received from) other islands. A neighborhood is defined in the *migration policy* in order to carry out a guided exchange of solutions between subpopulations. For the exchange of individuals we fix it every certain number of steps of the evolution process of the island.

The update of the islands could be done in a synchronous fashion but we use an asynchronical update where the individuals are received whenever they arrive, with no stops in the execution.

3.2. Island Based Model of dDE

In Algorithm 1, the pseudocode of our island model distributed DE is shown. In this algorithm, the whole population $\mathcal{P}$ is structured in m smaller subpopulations P_p of n_p individuals where $N = \sum_{i=1}^{m} n_i$. Each subpopulation is randomly initialized, and relatively isolated from the others, evolves independently in parallel performing periodical exchanges of solutions.

The migration policy is determined by a five-tuple $\mathcal{M} = \langle \gamma, \rho, \phi_s, \phi_r, \tau \rangle$ where $\gamma \in \mathbb{N}$ denotes the migration gap between two successive exchanges of individuals, $\rho \in \mathbb{N}$ denotes the migration rate in every exchange, the ϕ_s and ϕ_r functions decide how to select the individuals involved in the exchange, the *selection function* ϕ_s decides what individuals emigrate, and the *replacement function* ϕ_r determines the individuals to be substituted by the immigrants. The topological model is denoted by the function $\tau : \mathcal{P} \rightarrow 2^{\mathcal{P}}$, which selects what subpopulations can send to (or receive from) individuals.

In our algorithm the individuals to be migrated are randomly (uniformly) chosen by the selection function ϕ_s (line

Algorithm 1 Pseudocode of the Distributed DE

1: **pardo** for each $p \in \{1, ..., m\}$
2: initialize(P_p)
3: **while** not stop condition(g_{max}) **do**
4: perform a step of canonical DE // see [14]
5: **for** each of the ρ individuals to send **do**
6: $v_g'^i \leftarrow \phi_s(P_p)$
7: send $v_g'^i$ to P_j chosen by τ
8: **end for**
9: /*asynchronous communication*/
10: **while** individuals are arriving **do**
11: receive $v_g'^i$ from P_j
12: replace an individual chosen from $\phi_r(P_p)$ by $v_g'^i$
13: **end while**
14: **end while**
15: **end pardo**
16: **Output:** best solution found

6 in Algorithm 1). Incoming individuals from other islands replace randomly chosen local individuals, only if the formers are better, by the replacement function ϕ_r (line 12 in Algorithm 1). The topology is a unidirectional *ring* in which the individuals are exchanged with the nearest neighbor subpopulation.

4. Experimental Study

This section analyzes the behavior of dDE by performing a set of experiments plus a statistical study on the performance of the proposal. We include in our study the canonical DE and also the best-to-date algorithms for the tackled problems.

4.1 Test Functions

The CEC'05 benchmark is proposed in the technical report of Suganthan *et al.* in [18]. The test suite includes 25 functions, some of which are shifted and/or rotated versions of classical functions, plus others that are a hybridization of some of those functions. The first five functions are unimodal, while the rest are multimodal functions. We are interested in the last set of 20 functions because of its high difficult level. The set is divided in three groups, the first two groups are basic functions and expanded functions, respectively. The third group contains hybrid merging functions. All functions have the optimum shifted to a value different from zero named *bias*. The shifted function to a bias is useful to avoid a symmetric search space that the algorithms could exploit in its benefit.

4.2. Parameter Setting

Note that a preliminary study of the algorithm dDE was performed to tune the set of parameters. The population $\mathcal{P}$ was set to 20 individuals and was partitioned in two subpopulations (islands), each one having 10 individuals. The migration gap and migration rate were set to $\gamma = 100$ and $\rho = 1$, respectively. As previously mentioned, one randomly selected individual is sent from one of the island to the other in a non-blocking policy of migration. The values of the parameters μ and Cr are adjusted to each of the functions of the suite, and they are shown in Table 1.

Table 1. Parameters μ and Cr of dDE to each function of the suite test

Func.	30		50	
	μ	Cr	μ	Cr
f_6	5.0E-01	4.0E-01	5.0E-01	4.5E-01
f_7	5.0E-01	5.0E-01	5.0E-01	5.0E-01
f_8	5.0E-01	5.0E-01	5.0E-01	5.0E-01
f_9	9.0E-01	1.0E-02	9.0E-01	1.0E-02
f_{10}	5.0E-01	2.0E-01	4.0E-01	5.0E-01
f_{11}	9.0E-01	5.0E-01	9.0E-01	1.0E-01
f_{12}	5.0E-01	1.0E-01	5.0E-01	1.0E-01
f_{13}	9.0E-01	1.0E-03	9.0E-01	1.0E-03
f_{14}	9.0E-01	1.0E-03	9.0E-01	1.0E-03
f_{15}	1.0E-01	1.0E-03	9.0E-01	1.0E-03
f_{16}	9.0E-01	1.0E-02	9.0E-01	2.0E-01
f_{17}	9.0E-01	5.0E-01	9.0E-01	5.0E-01
f_{18}	5.5E-01	4.0E-01	5.5E-01	4.0E-01
f_{19}	5.0E-01	5.0E-01	5.0E-01	5.0E-01
f_{20}	5.5E-01	5.0E-01	5.5E-01	5.0E-01
f_{21}	5.0E-01	1.0E-02	5.0E-01	1.0E-01
f_{22}	5.0E-01	5.0E-01	5.0E-01	5.0E-01
f_{23}	5.0E-01	1.0E-01	5.0E-01	1.0E-02
f_{24}	9.0E-01	9.0E-01	9.0E-01	9.0E-01
f_{25}	9.0E-01	9.0E-01	9.0E-01	9.0E-01

The results shown in this work were obtained from 25 independent runs of the algorithm. The 20 benchmark functions were solved running the algorithm a total of $10^4 * D$ evaluations, where the number of dimensions D considered were 30 and 50. The skeleton architecture in C++ of the MALLBA Library [2] was used to fast develop the implementation dDE algorithm. The distributed communication platform is implemented with the MPICH library (v.1.5.2) that is executed over machines running Linux. All machines used in the experiments have the following specification: Pentium IV at 2.4 GHz with 1 GB of RAM and Linux SuSE operative system with kernel version 2.4.19-4GB.

4.3 Comparative Study

This section performs a preliminary comparative study of a canonical version of DE and our dDE to determinate their respective advantages. Later, we perform a (statistically grounded) comparison of one of our models (selected as the best in this previous comparison) with the best existing algorithms (from CEC'05).

To provide well-grounded results, we have applied the Shapiro and Wilk test of normality with error probability $p = 0.05$ in order to check whether the results follow a normal distribution or not. Then using the Levene test we checked the heterocedasticity of these results. Both tests obtained in all cases confidences (p) lower than 0.05. Therefore, although we can assure the independency of the samples since all executions were independently made, the results violate the normality and heterocedasticity. For this reason, non-parametric tests are applied in the following comparisons.

4.3.1 Comparative Study of dDE with a Canonical Version of DE

As a first analysis, we compare here the canonical version of DE (seqDE) with our island model DE (dDE). In the former, a whole population of 20 individuals evolves without separate structures. In the latter, the population (20 individuals) is split into two and four subpopulations, hence constituting two different versions of dDE: dDE_2 with two islands of 10 individuals each one, and dDE_4 with four similar islands of 5 individuals.

We have applied a Wilcoxon Signed-Rank test [20] to the mean error values (out of 25 independent runs) obtained for each 20 functions by these three algorithms regarding the two dimensions considered (30 and 50 variables). In Table 2, the results of applying this test are organized as follows: once the rankings are calculated they are differentiated between positive and negative, included in columns, R+ to indicate the mean ranks where the first algorithm gets higher values than the second algorithm, and the column R- showing the opposite situation, where the second algorithm gets higher values than the first one. We must notice that the lower values correspond to the algorithms with better behavior since we are minimizing the function fitness. The fifth column shows the p-values obtained by the test in all the comparisons, which indicates the existence of statistically significant differences between the compared algorithms, regarding the confidence level used (0.05). In order to clarify this, an asterisk character ($*$) is allocated in the last column (SD) to point out the existence of significant differences in the corresponding comparison.

Table 2. Signed-Rank Test of seqDE, dDE_2 and dDE_4 in terms of the mean error value, and significance level of 95% (p=0.05)

Algorithm	Dimension	R+	R-	p-value	SD
seqDE vs. dDE_2	30	105	**48**	0.185	
	50	110	**61**	0.295	
seqDE vs. dDE_4	30	**66**	105	0.408	
	50	107	**103**	0.956	
dDE_2 vs. dDE_4	30	**15**	156	0.002	$*$
	50	**31**	140	0.018	$*$

We can observe in Table 2 that there does not exist a significant difference between the canonical and distributed models of DE, although the best mean rank is always obtained by dDE_2. In addition, when comparing both distributed DE algorithms (dDE_2 and dDE_4) the differences are statistically significant, obtaining dDE_2 the best mean rank and overcoming the dDE_4 in all the dimensions.

Figure 1 shows the median performances (mean error) through the evolution steps of a typical execution of seqDE, dDE_2 and dDE_4, plotted when solving f_6 with dimension 30. Analyzing this figure, we can observe that both distributed versions of DE show a delayed convergence than the canonical one, probably generated by the migration policy in the formers. This migration operation promotes the diversity of individuals in the subpopulations, which can improve the final results in dDE_2. However, an excessive diversity, in contrast with a lower number of individuals in each subpopulation, can degrade the final results in dDE_4.

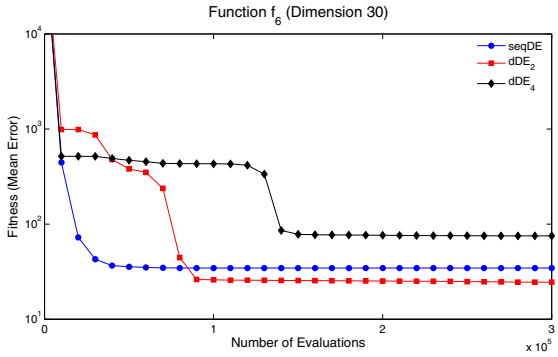

Figure 1. Number of function evaluations vs. the error ($f(x) - f(x^*)$, where x is the best solution found so far and x^* is the best solution known). Median performance of 25 runs of f_6 with dimension 30.

Therefore, the statistical results got in this previous analisys are useful to suggest the selection of dDE_2 as the base proposal to beat other existing algorithms.

4.3.2 Comparison with CEC'05 Algorithms

In this section, we compare the results obtained by our distributed DE, using the two islands model (dDE_2), with other algorithms following the CEC'05 standard protocol. The list of such algorithms includes several real-coded algorithms within the evolutionary computation paradigms; some of them have been improved by using hybridization techniques. The list comprises the following algorithms:

- BLXMA [11], Real-Coded Memetic Algorithm
- BLX-GL50[8], Hybrid Real-Coded Genetic Algorithm
- CoEVO[13], Cooperative Evolution

- DE[15], clasical DE
- G-CMA-ES[4], Evolution Strategy Adapting a Covariance Matrix
- K-PCX[16], Steady-State Optimization Algorithm
- L-CMA-ES [3], Covariance Matrix Algorithm Improved with a Local Search
- SPC-PNX[6], Steady-State Genetic Algorithm

Table 3 shows the results obtained by dDE_2 in terms of the mean error (out of 25 independent runs) for each multimodal function found in CEC'05 with dimension 30, in order to compare these results with results of other past and future studies following the same experimentation protocol.

Table 3. Results in terms of mean error values reached by dDE_2, dimension 30

Funct.	Mean Error	Funct.	Mean Error
f_6	2.16E+02	f_{16}	2.26E+02
f_7	3.76E-02	f_{17}	1.50E+02
f_8	2.10E+01	f_{18}	8.22E+02
f_9	1.19E-01	f_{19}	8.27E+02
f_{10}	9.04E+01	f_{20}	8.22E+02
f_{11}	3.90E+01	f_{21}	5.00E+02
f_{12}	7.27E+03	f_{22}	5.16E+02
f_{13}	9.12E-01	f_{23}	5.74E+02
f_{14}	1.28E+01	f_{24}	2.24E+02
f_{15}	1.43E+02	f_{25}	2.12E+02

In order to compare these results with the ones obtained by the CEC'05 algorithms (with dimension 30), we have followed the same statistical protocol as explained in the previous section (Section 4.3.1). In Table 4 the results of applying a Wilcoxon Signed-Rank test to dDE_2 versus each listed algorithm (CEC'05) are shown. In this table, R+ corresponds to the mean ranks obtained by dDE_2, and R- corresponds to the mean ranks of each compared algorithm. As we can observe, dDE_2 shows the best rank in almost all comparisons and is significantly better than CoEvo and DE. Concretely, the difference showed by our algorithm with regards to the regular DE (CEC'05) leads us to claim the real improvement obtained by dDE_2 since both algorithms perform basic operations of Differential Evolution.

Table 4. Signed-Rank Test of dDE_2 vs. CEC'05 algorithms with dimension 30 and significance level of 95% (p=0.05)

Algorithm	R+	R-	p-value	SD
G-CMA-ES	121	**69**	0.2950	
K-PCX	**89**	121	0.5500	
BLXMA	**76**	114	0.4450	
BLX-GL50	**65**	125	0.2270	
SPC-PNX	**62**	128	0.1840	
L-CMA-ES	**60**	150	0.0929	
DE	**32**	178	0.0064	$*$
CoEvo	**5**	205	0.0001	$*$

In this comparison, our dDE_2 is similarly ranked on top of all algorithms (that are very specialized). These last results are still improved in the following experiments, in which the multimodal functions are tackled at dimension 50. Table 5 shows the results obtained by dDE_2 in terms of mean error values (out of 25 independent runs) for each function with dimension 50.

Table 5. Results in terms of mean error values reached by dDE, dimension 50

Funct.	Mean Error	Funct.	Mean Error
f_6	1.18E+02	f_{16}	1.51E+02
f_7	3.80E-03	f_{17}	1.46E+02
f_8	2.12E+01	f_{18}	8.39E+02
f_9	7.96E-02	f_{19}	8.55E+02
f_{10}	2.05E+02	f_{20}	8.41E+02
f_{11}	5.11E+01	f_{21}	7.27E+02
f_{12}	2.96E+04	f_{22}	5.00E+02
f_{13}	1.80E+00	f_{23}	7.09E+02
f_{14}	2.26E+01	f_{24}	3.46E+02
f_{15}	1.04E+02	f_{25}	2.68E+02

In Table 6, the results (dimension 50) of applying the Wilcoxon Signed-Rank test to dDE_2 versus each listed algorithm (of CEC'05) are shown. In this case, only two algorithms (G-CMA-ES and L-CMA-ES) have ever been applied to these problems of high complexity. We can observe here that dDE_2 obtains the best mean rank in comparison with both CMA-ES versions, even being statistically better than L-CMA-ES. Concerning G-CMA-ES, the imposition of a high significance level (95%) in the statistical test does not allow to show a significant difference from our dDE_2, but it could be easily reached by using a significance level of 90%.

Table 6. Signed-Rank Test of dDE_2 vs. CEC'05 algorithms with dimension 50 and significance level of the 95% (p=0.05)

Algorithm	R+	R-	p-value	SD
G-CMA-ES	**59**	151	0.0859	
L-CMA-ES	**52**	158	0.0478	$*$

Therefore, we can state that our dDE_2 shows a competitive performance mainly in high dimensionality problems where only specialized versions of CMA-ES obtained results until now. The high specialization of CMA-ES for these problems and its complex implementation contrast with the wide applicability of our proposal and with its easy implementation and understanding.

5 Conclusions

In this work we have experimentally studied, in terms of the quality of solutions, the performance of a two island distributed Differential Evolution (dDE_2) whose population is structured in two subpopulations running in parallel, with a certain migration policy and island topology. Using the natural behavior of exploitation (and early convergence) observed in the basic DE, and incorporating a diversification mechanism by means of migrant particles, we can provide it with a higher search capacity in order to improve its global performance. The resulted algorithm (dDE_2) was tested on the benchmark of multimodal functions provided in the special session of continuous optimization of CEC'05 and used its protocol of experimentation to ensure fairness and future comparisons. Our proposal shows a highly competitive performance in comparison with the canonical version of DE and a four islands version of DE (dDE_4), and even improves on all the existing algorithms using the CEC'05 protocol of evaluation. With dimension 30, dDE_2 is the second best algorithm and statistically equivalent to the best. However, with dimension 50 our proposal beats all the algorithms, and shows a competitive performance in comparison with two well-known specialized variants of CMA-ES.

As future work, we plan to experiment with different variations of the parallel configuration as well as the evaluation of other novel large-scale functions test suite.

Acknowledgments.

E. Alba and J. García-Nieto acknowledge founds from the Spanish Ministry of Education TIN2005-08818-C04-01 (the OPLINK project, http://oplink.lcc.uma.es), and CICE, Junta de Andalucía under contract P07-TIC-03044 (DIRICOM project, http://diricom.lcc.uma.es).
J. Apolloni under research fellowship MAEC (BOE 213) would like to thank the Malaga University and the Spanish Agency for International Co-operation and Development Agency (AECID).

References

[1] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley, July 2005.

[2] E. Alba and M. Group. Mallba: A Library of Skeletons for Combinatorial Optimisation. In B. Monien and R. Feldmann, editors, *Proceedings of the Euro-Par*, volume LNCS 2400, pages 927–932, 2002.

[3] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. *IEEE Congress on Evolutionary Computation*, 2:1777–1784, 2005.

[4] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. *IEEE Congress on Evolutionary Computation*, 2:1769–1776, 2005.

[5] T. Back, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Computational Intelligence Library, The Institute of Physics; Ringbound, 1997.

[6] P. Ballester, J. Stephenson, J. Carter, and K. Gallagher. Real-parameter optimization performance study on the cec-2005 benchmark with spc-pnx. *IEEE Congress on Evolutionary Computation*, 1:498–505, 2005.

[7] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, Dec. 2006.

[8] C. Garcia-Martinez and M. Lozano. Hybrid real-coded genetic algorithms with female and male differentiation. *IEEE Congress on Evolutionary Computation*, 1:896–903, 2005.

[9] W. Kwedlo and K. Bandurski. A parallel differential evolution algorithm for neural network training. *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*, pages 319–324, 2006.

[10] M. Laguna and R. Martí. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *J. of Global Optimization*, 33(2):235–255, 2005.

[11] D. Molina, F. Herrera, and M. Lozano. Adaptive local search parameters for real-coded memetic algorithms. *IEEE Congress on Evolutionary Computation*, 1:888–895, 2005.

[12] V. Plagianakos and M. Vrahatis. Parallel evolutionary training algorithms for 'hardware-friendly' neural networks. *Natural Computing*, 1:307–322, 2002.

[13] P. Posik. Real-parameter optimization using the mutation step co-evolution. *IEEE Congress on Evolutionary Computation*, 1:872–879, 2005.

[14] K. V. Price, R. Storn, and J. Lampinen. *Differential Evolution: A practical Approach to Global Optimization*. Springer-Verlag, London, UK, 2005.

[15] J. Ronkkonen, S. Kukkonen, and K. Price. Real-parameter optimization with differential evolution. *IEEE Congress on Evolutionary Computation*, 1:506–513, 2005.

[16] A. Sinha, S. Tiwari, and K. Deb. A population-based, steady-state procedure for real-parameter optimization. *IEEE Congress on Evolutionary Computation*, 1:514–521, 2005.

[17] R. Storn and K. V. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, TR-95012-ICSI, 1995.

[18] 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 KanGAL Report 2005005, Nanyang Technological University, Singapore and Kanpur, India, 2005.

[19] D. Tasoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis. Parallel differential evolution. *Congress on Evolutionary Computation, CEC2004*, 2:2023–2029 Vol.2, 19-23 June 2004.

[20] R. Wilcox. *New statistical procedures for the social sciences*. Hillsdale, 1987.