Performance Comparison of Parameter Variation Operators in Self-Adaptive Differential Evolution Algorithms

Rodrigo C. Pedrosa Silva*, Rodolfo A. Lopes*, Alan R. R. Freitas * and Frederico G. Guimarães†
*Graduate Program in Electrical Engineering
Federal University of Minas Gerais, UFMG, Belo Horizonte - MG, Brazil
Email: {rcpsilva, rodolfo.ufop, alandefreitas}@gmail.com
†Department of Electrical Engineering
Federal University of Minas Gerais, UFMG, Belo Horizonte - MG, Brazil
Email: fredericoguimaraes@ufmg.com

***Abstract*—Differential Evolution (DE) algorithm is an important Evolutionary Algorithm (EA) for global optimization over continuous spaces, which can also work with discrete variables. The success of DE in solving a specific problem is closely related to appropriately choosing its control parameters, in this context, self-adaptation allows the algorithm to reconfigure itself, automatically adapting to the problem being solved. In self-adaptation the control parameters are encoded into the genotype of the individuals and undergo the actions of variation operators. In the literature, there are several different operators proposed to vary the encoded parameters, however, there is a lack of information about their influence on the algorithms performance. To cover part of this lack of knowledge, in this paper a comparison of variation operators, commonly used to adapt parameters in self-adaptive versions of DE, is presented. The experiments on well know benchmark functions indicates that operators which maintain the control parameters diversity work better than the others.**

***Keywords*-Self-adaptation; Differential Evolution; Evolutionay Algorithms; Numerical Optimization;**

I. Introduction

Differential Evolution (DE) [1] is an important Evolutionary Algorithm (EA) for global optimization over continuous spaces, which can also work with discrete variables [2], [3]. The reasons for its success can be found in its simplicity and ease of implementation, while at the same time demonstrating reliability and high performance [4]. Its importance can be ascertained in the wide number of applications in which it was successfully used, such as, data clustering [5], flow shop scheduling [6], digital filter design [7], etc. Besides, it has been been showing remarkable performance in optimization contests, for instance, in the IEEE International Conference on Evolutionary Computation (CEC) contests like, CEC 2006 competition on constrained real parameter optimization (first rank), CEC 2007 competition on multiobjetive optimization (second rank), CEC 2008 competition on large scale global optimization (third rank) and CEC 2009 competition on evolutionary computation in dynamic and uncertain environments (first rank).

In DE new candidate solutions are created by combining a parent individual with several other individuals of the same population. Then, this candidate solution replaces the parent if it has a better fitness value. DE has three control parameters: the population size NP, the scale factor of the perturbations generated by mutation F and the crossover constant CR. As shown in [8], the choice of suitable parameter settings is critical on DE performance and depends on the problem being solved. Usually this may lead to additional computational costs due to the time-consuming parameter tuning process. In this context, self-adaptation has proven to be highly beneficial in automatically and dynamically adjusting evolutionary parameters in DE [8]–[11].

In Self-adaptation the control parameters are encoded into the genotype of the individuals and undergo the actions of variation operators. The better values of these encoded parameters tend to lead to better solutions which are more likely to survive and, hence, more likely to propagate these good parameter values.

In the literature, there are several different operators proposed to vary the encoded parameters in the self-adaptive versions of DE. However, there is a lack of information in the related literature about the influence of these operators on the algorithm performance. In this context, the objective of this paper is to compare common variation operators used in self-adaptive versions of DE in well known benchmark functions. In order to do that, a generic self-adaptive DE was implemented with different variation operators from the literature. These different versions of the self-adaptive DE were then tested on BBOB (Black Box Optimization Benchmark) [12]. The results have shown that among the majority of the operators there is not a great difference, however, there are some operators that stand out for good performance, and others for poor performance.

The remainder of this article is structured as follows: Section II briefly describes the DE algorithm, Section III describes the implemented self-adaptive DE, Section IV describes the used variation operators, Section V presents the results and finally Section VI presents the conclusions and final remarks.

II. Differential Evolution Algorithm

In this section, the basic DE algorithm is briefly reviewed, see [1], [13], [14] for additional details. Like other evolutionary algorithms, the original DE algorithm works with a population of candidate solutions randomly generated within the domain region of the problem, usually described as:

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^n : x_k^{min} \leq x_k \leq x_k^{max}, k = 1, ..., D \right\} \quad (1)$$

where $x_k^{\min}$ and $x_k^{\max}$ are respectively the low and upper limits of each variable and D is the problem dimension, i.e., the number of variables in the problem.

We adopt in this paper the notation $x_{g,i,j}$ such that $g = 1, \ldots, G$ represents the generation counter; $i = 1, \ldots, NP$ represents the index of the individual in the population; and $j = 1, \ldots, D$ represents the variable index. A given individual is represented by:

$$\mathbf{x}_{g,i} = \left\langle x_{g,i,1}, x_{g,i,2}, x_{g,i,3}, ..., x_{g,i,D} \right\rangle \quad (2)$$

New individuals are generated by using the differential mutation. Mutation is based on the difference between individuals randomly chosen from the current population leading to the so-called mutant vector. Four widely used DE mutation operators are shown as follows:

1) **`rand/1`**

$$\mathbf{v}_{t,i} = \mathbf{x}_{t,r1} + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) \quad (3)$$

2) **`current-to-best/1`**

$$\mathbf{v}_{t,i} = \mathbf{x}_{t,i} + F(\mathbf{x}_{t,best} - \mathbf{x}_{t,i}) + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) \quad (4)$$

3) **`rand/2`**

$$\mathbf{v}_{t,i} = \mathbf{x}_{t,r1} + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) + F(\mathbf{x}_{t,r4} - \mathbf{x}_{t,r5}) \quad (5)$$

4) **`current-to-rand/1`**

$$\mathbf{v}_{t,i} = \mathbf{x}_{t,i} + F(\mathbf{x}_{t,r1} - \mathbf{x}_{t,i}) + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) \quad (6)$$

where $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \in \{1, \ldots, NP\}$ are mutually distinct random indexes, $best$ is the index of the best individual in the current population and F the scale factor applied to the differential vector. For each $\mathbf{x}_{g,i}$ in the population a corresponding mutant solution $\mathbf{v}_{g,i}$ is generated.

A trial vector $\mathbf{u}_{g,i}$ is produced through recombination of $\mathbf{x}_{g,i}$ and $\mathbf{v}_{g,i}$. In the basic DE algorithm, the discrete recombination with probability CR is used, as we can see in the following scheme:

$$\mathbf{u}_{g,i,j} = \begin{cases} \mathbf{v}_{g,i,j}, & \text{if } \mathcal{U}_{[0,1]} \leq CR \\ \mathbf{x}_{g,i,j}, & \text{otherwise} \end{cases} \quad (7)$$

where $\mathcal{U}_{[a,b]}$ represents the sampling of a random variable with uniform distribution in the interval $[a, b]$. In this way, F and CR represent control parameters of the algorithm.

Finally, the trial vector $\mathbf{u}_{g,i}$ competes with the current solution $\mathbf{x}_{g,i}$ based on their objective function evaluations. If the trial solution is better or equal than the current solution, it replaces the current solution, otherwise the current solution survives while the trial one is eliminated, as described below:

$$\mathbf{x}_{g+1,i} = \begin{cases} \mathbf{u}_{g,i} & \text{if } f(\mathbf{u}_{g,i}) \leq f(\mathbf{x}_{g,i}) \\ \mathbf{x}_{g,i} & \text{otherwise} \end{cases} \quad (8)$$

III. Generic Self-Adaptive Differential Evolution

In this section the implemented self-adaptive version of DE, in which the different variation operators are tested, is described. For simplicity this version will be called DEPS (Differential Evolution with Parameter Self-adaptation).

As mentioned before, in self-adaptation the control parameters are encoded into the solutions' genotype. Thus, all solutions in DEPS will have the following aspect:

$$\mathbf{x}_{g,i} = \left\langle x_{g,i,1}, ..., x_{g,i,D}, F_{g,i}, CR_{g,i} \right\rangle \quad (9)$$

where $x_{i,d}$ are the problem variables, F_i and CR_i are the control parameters related to the ith individual.

Before the mutation and crossover operations, the control parameters of each individual are updated with a defined variation operator. After this step the algorithm follows the usual flow of differential evolution. If the solution produced using the updated control parameters survives, the new control parameters survive as well, otherwise, the new control parameters are ignored and the previous ones are kept. For the sake of clarity, the pseudo-code of DEPS is shown in Algorithm 1.

Algorithm 1: DEPS

```
Initialize population;
for each individual x_{1,i} do
    Initialize(F_{1,i}, CR_{1,i});
end
while ¬ stop_condition do
    for each individual x_{g,i} do
        [newF_{g,i}, newCR_{g,i}] = Update([F_{g,i}, CRg, i]);
        Apply Mutation using newF_{g,i};
        Perform Recombination with newCR_{g,i};
        Apply Selection;
    end
end
```

IV. Variation Operators

This section describes the different variation operators used in this work. These operators were chosen by their constant appearance in relevant papers related to DE self-adaptation.

A. Differential Evolution

In this mechanism the differential evolution itself is used as a variation operator. The pairs F_i, CR_i are seen as individuals and go through mutation and crossover. In this way, this variation procedure works as follows:

$$\begin{aligned} mF_{g,i} &= F_{g,r1} + F'(F_{g,r2} - F_{g,r3}) \\ mCR_{g,i} &= CR_{g,r1} + F'(CR_{g,r2} - CR_{g,r3}) \end{aligned} \tag{10}$$

where mF and mCR are the "mutant" parameters, $r_1 \neq r_2 \neq r_3 \in \{1, \ldots, NP\}$ are mutually distinct random indexes and $F' = \mathcal{U}_{[0,1]}$. After mutation the control parameters go through crossover as well:

$$\begin{aligned} newF_{g,i} &= \begin{cases} mF_{g,i}, & \text{if } \mathcal{U}_{[0,1]} \leq CR' \\ F_{g,i}, & \text{otherwise} \end{cases} \\ newCR_{g,i} &= \begin{cases} mCR_{g,i}, & \text{if } \mathcal{U}_{[0,1]} \leq CR' \\ CR_{g,i}, & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

where $CR' = \mathcal{U}_{[0,1]}$.

This mechanism can be found with some variation in [10] and [15].

B. Random Perturbation

As the name suggests, in this operator, the new control parameters are just generated again in a random way, as shown in Equation 12, with a certain probability τ. $newF$ is in the interval $[0.1, 1]$ so that mutation is not null. This mechanism can be found in [8], [16] and [17].

$$\begin{aligned} newF_{g,i} &= \mathcal{U}_{[0.1,1]} \text{ , if } \mathcal{U}_{[0,1]} \leq \tau \\ newCR_{g,i} &= \mathcal{U}_{[0,1]} \text{ , if } \mathcal{U}_{[0,1]} \leq \tau \end{aligned} \tag{12}$$

C. Gaussian Perturbation

In this variation operator, the encoded control parameters are adapted by means of gaussian perturbations on the previous values. This procedure works as follows:

$$\begin{aligned} newF_{g,i} &= \mathcal{N}_{[F_{g,i},0.1]} \\ newCR_{g,i} &= \mathcal{N}_{[CR_{g,i},0.1]} \end{aligned} \tag{13}$$

where $\mathcal{N}_{[a,b]}$ represents the sampling of a random variable with gaussian distribution, with mean a and standard deviation b. The standard deviations were set to 0.1 because this value is commonly used when this variation operator is applied. This can be seen in [4], [9] and [18].

D. Guided Gaussian Perturbation

This operator was proposed in [18] and can be seen in [19] and [9] as well. It also uses gaussian perturbations, however the means and standard deviations are different from the above described procedure. F is generated by $\mathcal{N}_{[0.5,0.3]}$. The CRs are initially generated by $\mathcal{N}_{[0.5,0.1]}$, then the CRs which produced improved solutions are stored. Therefore, the new CRs are produced by $\mathcal{N}_{[CRm,0.1]}$ where CRm is the median of the stored CRs. This procedure is synthesized by the following equations:

$$\begin{aligned} CR_{1,i} &= \mathcal{N}_{[0.5,0.1]} \\ newF_{g,i} &= \mathcal{N}_{[0.5,0.3]} \\ newCR_{g,i} &= \mathcal{N}_{[CRm_{g,i},0.1]} \end{aligned} \tag{14}$$

E. Pool of Parameters

In this procedure a parameter candidate pool is built and at each iteration a member of the pool is randomly attributed to the pair $[newF_{g,i}, newCR_{g,i}]$. This scheme was recently proposed in [20], and the parameters of the candidate pool were chosen based on the frequency of their use in various DE variants and on their well studied properties. The pool is composed by the following pairs:

1) $[F = 1.0, CR = 0.1]$
2) $[F = 1.0, CR = 0.9]$
3) $[F = 0.8, CR = 0.2]$

V. Results

In the experiments, 5 versions of Algorithm 1 were implemented. Each version differs only in the variation operator used on the encoded parameters. As the most used and common operator in the literature the `rand/1` mutation variant (Eq.3) was chosen. In the charts the algorithm variants will be referred as:

1) DEPS11 - DEPS using differential evolution as variation operator;
2) DEPS21 - DEPS using random perturbations;
3) DEPS31 - DEPS using gaussian perturbations;
4) DEPS41 - DEPS using guided gaussian perturbations; and
5) DEPS51 - DEPS using a pool of parameters;

For the purpose of this work a common parameter setting, found in the literature, was used and no effort was made to configure it. The probability τ for DEPS21 was set in 0.1. The population sizes were set in 30 individuals for the functions with 2 to 10 dimensions and 50 individuals for the functions with 20 and 40 dimensions, the maximum number of function evaluations was set in $10^5 \times D$, where D is the number of dimensions.

Table I presents the final convergence proportions of the algorithms in each dimension. A Friedman test was performed to detected significant differences ($p < 0.05$) among them, and when needed post-hoc comparisons (Wilcoxon Rank-Sum tests) were used to detect significant differences between the pairs of algorithms. Identical letters mean no statistical difference and different letters mean statistical difference.

The variants can be split into two groups. Group 1 has the variants that select new parameters in a random way, such as, DEPS21, DEPS31 and DEPS51. And Group 2 has the variants in which there is a pressure for some values, such as, DEPS11 which uses DE itself and has an embedded

	DEPS11	DEPS21	DEPS31	DEPS41	DEPS51
2D	$94.2\%^{(\mathbf{a})}$	$97.8\%^{(\mathbf{a})}$	$96.1\%^{(\mathbf{a})}$	$95.0\%^{(\mathbf{a})}$	$98.1\%^{(\mathbf{a})}$
3D	$71.1\%^{(\mathbf{a})}$	$92.5\%^{(\mathbf{a})}$	$93.3\%^{(\mathbf{a})}$	$83.1\%^{(\mathbf{a})}$	$92.5\%^{(\mathbf{a})}$
5D	$21.9\%^{(\mathbf{b})}$	$80.0\%^{(\mathbf{a})}$	$76.9\%^{(\mathbf{a})}$	$39.7\%^{(\mathbf{b})}$	$80.0\%^{(\mathbf{a})}$
10D	$6.4\%^{(\mathbf{c})}$	$51.1\%^{(\mathbf{a})}$	$50.3\%^{(\mathbf{a},\mathbf{b})}$	$21.9\%^{(\mathbf{b},\mathbf{c})}$	$46.9\%^{(\mathbf{a},\mathbf{b})}$
20D	$4.7\%^{(\mathbf{c})}$	$44.7\%^{(\mathbf{a})}$	$32.2\%^{(\mathbf{a},\mathbf{b})}$	$18.6\%^{(\mathbf{b},\mathbf{c})}$	$39.4\%^{(\mathbf{a},\mathbf{b})}$
40D	$2.5\%^{(\mathbf{b})}$	$36.4\%^{(\mathbf{a})}$	$21.1\%^{(\mathbf{a},\mathbf{b})}$	$16.9\%^{(\mathbf{a},\mathbf{b})}$	$25.6\%^{(\mathbf{a})}$

Table I
SIGNIFICANCE ANALYSIS

selective pressure and DEPS21 which pushes the center of the gaussian distribution for certain values.

For small dimensions (2 and 3), there was no difference among any of the variations. As can be seen the variants in each group had significantly equal performances. In overall, the DEPS21 variant achieved the best results. It was significantly better than the variants of Group 2 in almost all experiments with higher dimensions (5 or more). The worst results were presented by DEPS11, for dimensions higher than 3 it presented significantly worse results when compared with the variants of group 1. DEPS41 had competitive results when compared with Group 1 variants, however, taking apart the results for 40 dimensions it performed significantly worse than DEPS21.

Figure 1 depicts the empirical cumulative distribution of runtimes[1] (RTs) of each algorithm on all functions $f1$-$f24$ of BBOB [12], grouped by number of dimensions (D). For each function with a given dimension, different target precision values[2] are used within the range $\Delta f_t \in [10^{-8}, 100]$. Each graph in Figure 1 shows the proportion of solved problems, in a given dimension, by the runtime. Notice that the maximum allowed runtimes are marked with "$\times$" in the charts, after these marks the points do not represent any data. For more detailed information about the graphs and the experimental procedure see [21], [22]. The description of the benchmark functions can be found in [23] and [24].

As can be seen among the variants in Group 1, despite of having statistically equal convergence proportions, DEPS21 and DEPS31 are faster than DEPS51, converging in a higher number of problems with less function evaluations. Even DEPS11 and DEPS41 can perform better than DEPS51 with a smaller number of function evaluations.

In overall the variants which maintained parameters diversity presented better convergence proportions. When using differential evolution as a variation operator, there will be an indirect selective pressure on the encoded parameters what can make them to converge. Looking through the results this characteristic seems to be interesting if the number of available function evaluations is limited, however, in a long term it does not work well. A possible reason for this is that with the convergence of the control parameters, the number of possible movements in the search space will be limited and this can cause the stagnation of DE. This can also explain the poor performance of the "guided gaussian" which limits the ranges of parameter generation.

The charts also show the best results found in 2009 (see [12]). It is important to remark that the best2009 are the overall best results and could be obtained by different methods. Furthermore, some of the algorithms employed there used more complex approaches, which might involve local search procedures, multiple populations and a fine parameter tunning in order to improve their performance. This would increase the number of factors involved with the algorithms performance, losing the focus on the variation operators.

[1]Runtime (RT) is the number of function evaluations until the optimal solution (with a certain precision) was reached.

[2]The target precision value $\Delta f_t = f_{target} - f_{optimun}$.

VI. CONCLUSION

In this paper, a comparison of variation operators commonly used to adapt parameters in self-adaptive versions of DE is presented. In order to do this, a generic self-adaptive differential evolution was implemented and the different variation operators were tested on it. The main objective of the work was to cover a part of the lack of knowledge about the influence of these operators in the algorithms performance.

The experiments on well known benchmark functions have shown the difference of performance caused by these operators. In synthesis, the results indicate that operators which maintain the control parameters diversity work better than the other ones. On this account, the work has accomplished its objective, producing knowledge which will help future development of self-adaptive algorithms, in special, self-adaptive DE versions.

ACKNOWLEDGMENT

The authors would like to thank the following Brazilian agencies for the financial support. National Council for Scientific and Technological Development (CNPq), FAPEMIG and CAPES.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep., 1995.

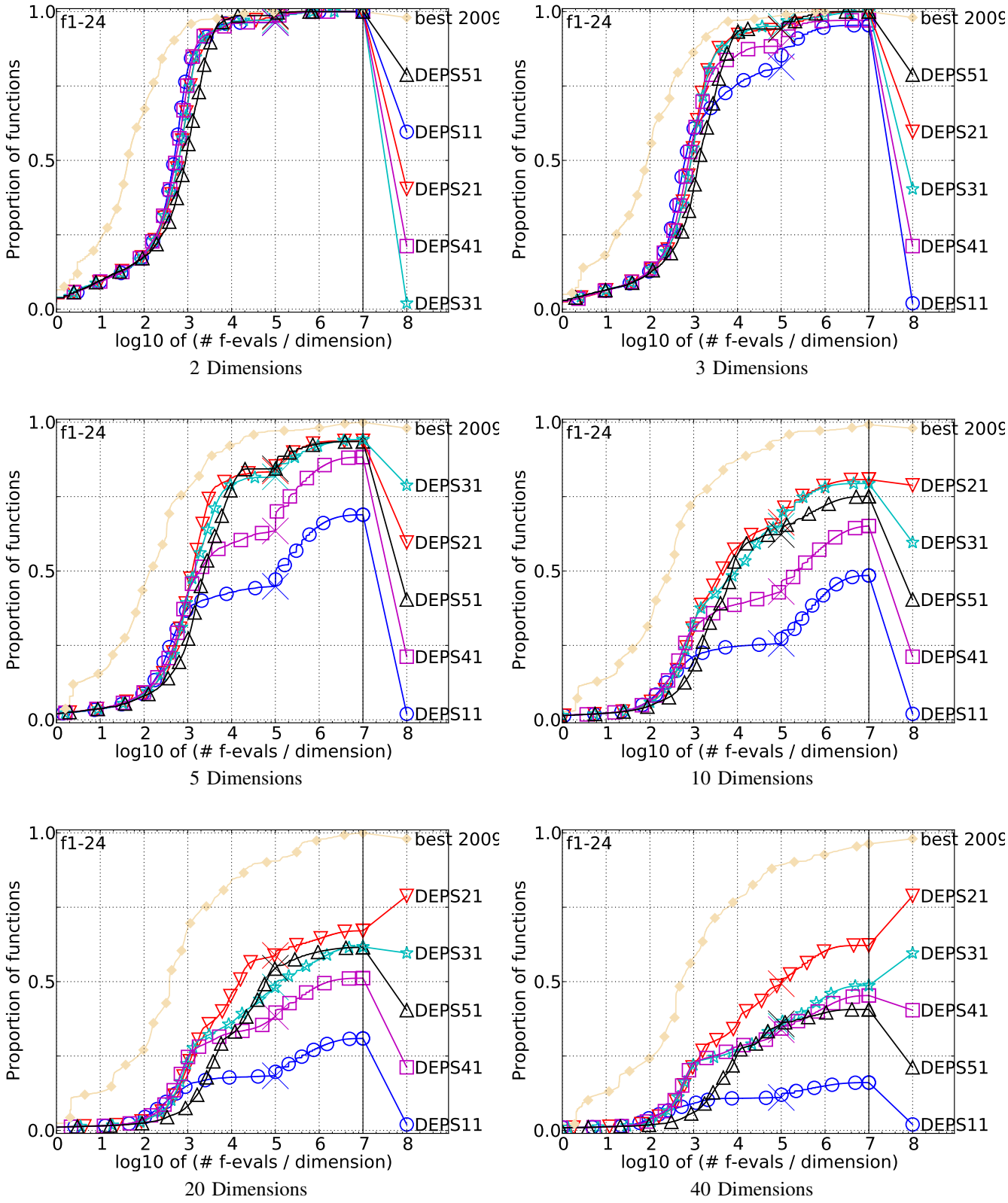

Figure 1. Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

[2] R. S. Prado, R. C. Pedrosa Silva, F. G. Guimaraes, and O. M. Neto, "Using differential evolution for combinatorial optimization: A general approach," in *In 2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*. IEEE Press, 2010, pp. 11–18.

[3] G. Onwubolu and D. Davendra, *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*, ser. Studies in Computational Intelligence. Springer, 2009. [Online]. Available: http://books.google.com.br/books?id=_NFr2iAMoogC

[4] M. Weber, V. Tirronen, and F. Neri, *Soft Computing*, vol. 14, no. 11, pp. 1187–1207, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1007/s00500-009-0510-5

[5] H. A. Sai, B. A. Vinaya, A. Govardhan, and S. S. C., "Data clustering using almost parameter free differential evolution technique," *International Journal of Computer Applications*, vol. 8, no. 13, pp. 1–7, October 2010, published By Foundation of Computer Science.

[6] G. C. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171, no. 2, pp. 674–692, 2006.

[7] N. Karaboga, "Digital iir filter design using differential evolution algorithm," *EURASIP J. Appl. Signal Process.*, vol. 2005, pp. 1269–1276, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1155/ASP.2005.1269

[8] J. Brest and a. M. S. M. Viljem Zumer, "Control parameters in self-adaptive differential evolution," 2006.

[9] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transaction on Evolutionary Computation*, vol. 13, pp. 398–417, April 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1650356.1650368

[10] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 673–686, 2006.

[11] V. Tirronen, F. Neri, and T. Rossi, "Enhancing differential evolution frameworks by scale factor local search - part i," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 94–101.

[12] "BBOB2012 - black box optimization benchmark," April 2012. [Online]. Available: http://coco.gforge.inria.fr/doku.php?id=bbob-2012

[13] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[14] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*, ser. Natural Computing Series, G. Rozenberg, T. Bäck, A. E. Eiben, J. N. Kok, and H. P. Spaink, Eds. Berlin, Germany: Springer-Verlag, 2005.

[15] R. C. Pedrosa Silva, R. A. Lopes, and F. G. Guimarães, "Self-adaptive mutation in the differential evolution," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 1939–1946. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001837

[16] J. Brest, "Constrained Real-Parameter Optimization with ϵ-Self-Adaptive Differential Evolution," in *Constraint-Handling in Evolutionary Computation*, E. Mezura-Montes, Ed. Berlin: Springer. Studies in Computational Intelligence, Volume 198, 2009, ch. 4, pp. 73–93, iSBN 978-3-642-00618-0.

[17] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Differential Evolution with Self-Adaptation and Local Search for Constrained Multiobjective Optimization," in *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*. Trondheim, Norway: IEEE Service Center, May 2009, pp. 195–202.

[18] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *IEEE Congress on Evolutionary Computation (CEC 2005)*. IEEE Press, 2005, pp. 1785–1791.

[19] V. L. Huang, S. Z. Zhao, R. Mallipeddi, and P. N. Suganthan, "Multi-objective optimization using self-adaptive differential evolution algorithm," in *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, ser. CEC'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 190–194. [Online]. Available: http://dl.acm.org/citation.cfm?id=1689599.1689624

[20] Y. Wang, Z. Cai, S. Member, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *Evolutionary Computation IEEE Transactions on*, vol. 15, no. 1, pp. 55–66, 2011.

[21] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2012: Experimental setup," INRIA, Tech. Rep., 2012. [Online]. Available: http://coco.gforge.inria.fr/bbob2012-downloads

[22] K. Price, "Differential evolution vs. the functions of the second ICEO," in *Proceedings of the IEEE International Congress on Evolutionary Computation*, 1997, pp. 153–157.

[23] S. Finck, N. Hansen, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions," Research Center PPE, Tech. Rep. 2009/20, 2009, updated February 2010. [Online]. Available: http://coco.gforge.inria.fr/bbob2010-downloads

[24] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2009, updated February 2010. [Online]. Available: http://coco.gforge.inria.fr/bbob2012-downloads