A Population Adaptation Mechanism for Differential Evolution Algorithm

Johanna Aalto
Department of Computer Science
University of Vaasa
Vaasa, Finland
johanna.aalto@uva.fi

Jouni Lampinen
Department of Computer Science
University of Vaasa
Vaasa, Finland
jouni.lampinen@uva.fi

***Abstract*— In the original Differential Evolution algorithm three different control parameter values must be pre-specified by a user. These parameters are the population size, the crossover constant and the mutation scale factor. Control parameters affect strongly the performance and reliability of the algorithm. However, choosing good parameters can be very difficult for a user. In this paper a new adaptive Differential Evolution algorithm called Cumu-DE is proposed. The aim of the algorithm is to be a user friendly and reliable algorithm with moderate convergence speed. In the proposed algorithm, the so called effective population size is adapted automatically using mechanism based on probability mass function. The actual population size is kept fixed. Even though we talk about two different population sizes, we have only one population. The effective population size describes the effective part of the actual population. The more we get successful trials, the smaller the effective population is and vice versa. The algorithm was initially evaluated by using the set of 25 benchmark functions provided by CEC2005 special session on real-parameter optimization. It was compared with the results of standard DE/rand/1/bin. The proposed algorithm Cumu-DE proved to be significantly faster due to its average of FES in four cases and significantly slower in six cases. Additionally, Cumu-DE was significantly more reliable in six cases and significantly less reliable in none. These results are demonstrating the potential of the proposed adaptation approach.**

I. INTRODUCTION

Differential evolution (DE) algorithm is an evolutionary optimization method, which was first introduced by Storn and Price [1]. It is simple but yet an effective stochastic population-based algorithm, which was originally designed to operate on continuous variables [2].

Even though DE is one of the most successful algorithms for solving global optimization problems, there might be - as in many other evolutionary algorithms - difficulties for a user to choose appropriate control parameters. Often the user is not aware of or even interested in the complexity of tuning parameters. Control parameters affect substantially to performance and reliability of the algorithm. In DE, these control parameters are the population size *NP*, the crossover constant *CR* and the mutation scale factor *F*. Creating fully automatic plug-in solvers for global optimization requires an automatic adaptation to all three control parameters.

Eiben [3] categorized the control parameter setting in three different classes: deterministic, adaptive and self-adaptive mechanism. When there is no feedback from the search process, the parameter control is called deterministic. In adaptive parameter control, there is some feedback obtained from the search process which can be used to tune parameters. In self-adaptive parameter control, parameters are encoded in individuals and they go through the recombination and mutation with the individual. In this paper, we propose an algorithm with adaptive approach.

The aim of the algorithm is to adapt one of the control parameters – population size *NP* – automatically using a novel mechanism. The two other control parameters, *F* and *CR*, will be kept fixed. In the adaptation process, we tune the so called effective population size while the actual population size is kept fixed. Note, that even though we talk about two different population sizes, we do have only one population. The effective population size describes the very effective part of actual population.

Briefly, the individuals for trial population are selected based on their probability. For example, if an individual has a very low probability, it is the most unlikely that it is selected for the trial population. The more individuals with low probability we have, the smaller the effective part of population is and vice versa.

To calculate probabilities for each individual, we use a cumulative distribution function and a probability mass function which is also called a probability density function. Tuning these functions we can increase or decrease the effective population size in a needed direction. Increasing or decreasing depend on, how many successful trial vectors we had in previous generation.

To compare the proposed approach Cumu-DE initially with the corresponding standard DE/rand/1/bin version, we used 25 benchmark functions provided by CEC2005 special session on real-parameter optimization [4]. The statistical significance of the relative changes of FES was tested by the Wilcoxon two-sample (the sum of rank) test and reliability rate by the Fisher exact test for 2-by-2 contingency table.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.214

Cumu-DE was significantly faster in four cases and significantly slower in six cases. However, it was significantly more reliable in six cases and significantly less reliable in none. These results are motivating the studying further of the proposed adaptation approach.

II. Related work

A lot of research has been done within all of the categories Eiben introduced [3]. For example, Abbas et al. introduced both adaptive and self-adaptive versions of the Pareto-frontier Differential Evolution algorithm PDE [5, 6]. An adaptive algorithm FaDE [7] by Liu and Lampinen uses the fuzzy logic controller to adapt *F* and *CR*. Aalto and Lampinen introduces adaptive approaches in [8-10]. Also jDE [11] by Brest, JaDE by Zhang and Sanderson [12, 13], and Tvrdík in [14] use adaptive approaches. In self-adaptive approach called SaDE by Qin and Suganthan, a learning strategy is used to self-adapt *F* and *CR* [15, 16]. Mallipeddi and Suganthan introduced EPSDE in [17]. Tvrdík et al. compares adaptive variants in [18, 19] and Tvrdík and Poláková in [20]. In CoDE by Wang et al. a multiple parameter setting approaches is combined [21]. Population adaptation has also been studied for example by Teo in [22, 23], Mallipeddi and Suganthan in [24], Teng in [25], Yang et al. in [26, 27] and many others.

III. Differential evolution algorithm

In this paper we applied a classic version of Differential Evolution algorithm, also called Classic DE or DE/rand/1/bin, both as a basis of constructing the proposed algorithm, and also to provide a baseline for experimental comparisons. All test functions used in this paper are global optimization problems, which can be expressed as in [1, 28].

First, the population will be initialized with random values generated within the given boundary constraints. From the first generation forward, vectors in the current population, P_G are sampled and combined to create candidate vectors for the subsequent generation P_{G+1}. The population of trial vectors $P'_{G+1}=u_{j,i,G+1}$, is generated as follows:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} = x_{j,r_3,G} + F \cdot \left(x_{j,r_1,G} - x_{j,r_2,G} \right) \ldots \\ \ldots\text{if } rand_j[0,1) \leq CR \vee j = k \\ x_{j,i,G} \quad \text{otherwise,} \end{cases} \tag{1}$$

where $i = \{1,2,...,NP\}$, $j = 1,2,...,D$ and $k \in \{1,2,...,D\}$. Additionally $r_1, r_2, r_3 \in \{1,2,...,NP\} \mid r_1 \neq r_2 \neq r_3 \neq i$.

Coello gives guidelines how to choose the best constraint-handling technique for different problems in [29]. In this study, if the trial vector violates possible boundary constraints, it will be handled as Liu and Lampinen [30] in (2). More detailed information of Differential Evolution algorithm can be found from [1] by Storn and Price.

$$u_{j,i,G} = x_{j,L} + (x_{j,U} - x_{j,L}) * rand[0,1] \tag{2}$$

IV. Cumulative Distribution Function, Probability Mass Function and Expected Value

In this chapter we go through the basics of cumulative distribution function, probability mass function and the expected value. These contents are crucial when using Cumu-DE.

The cumulative distribution function (CDF) gives the probability *P* that a random variable *X* is less than or equal to the independent variable *x*. CDF is defined for discrete random variables as in (3).

$$F(x) = P(X \leq x) \tag{3}$$

The probability *P* that a discrete random variable *X* takes on certain values *x*, is frequently denoted *f(x)*. The function *f(x)* is typically called the probability mass function (PMF), although some authors also refer to it as the frequency function or probability density function. PMF is defined as in (4).

$$f(x) = P(X = x) \tag{4}$$

The expected value *E(X)*, also known as population average μ of a discrete random variable *X*, is the average of all possible values based on their probability. The expected value of *X* is calculated by multiplying each possible value of the random variable by its probability and then adding the results up. *E(X)* is defined as in (5).

$$E(X) = \sum x\, f(x) \tag{5}$$

In the proposed algorithm Cumu-DE, we adapt the so called effective population size NP_{eff} using the mechanisms introduced above. In next chapter, we present the idea of this adaptation mechanism.

V. Cumu-DE

The basic idea of our algorithm is that we slow down the optimization process to achieve better reliability. It is obvious, that if we choose a population size large enough (population with all possible solutions), we can always find a feasible solution if there is one. However, this kind of procedure slows down the optimization process excessively. So, to choose a suitable population size can be very difficult for a user. Thus the goal of our algorithm is to be a user-friendly and reliable algorithm, which adapts the effective population size and finds a solution with moderate speed. To achieve this goal, we increase the effective population size when we get too many trials that fail or are even with corresponding vector in current population, and decrease the size when we have enough successful trials. Next we introduce this procedure more precisely.

The Cumu-DE is identical with the classic DE/rand/1/bin algorithm with the exception of adaptation of the effective population size NP_{Eff}. Note, that even though we talk about two different population sizes, we still have only one population. The effective population size NP_{Eff} describes the effective part of actual population *NP*. So, we have a fixed population size *NP* during the whole run. What we adapt during the run, is this very effective population size NP_{Eff}. Population is not sorted in any point of process.

First we calculate the cumulative distribution function CDF, the probability mass function PMF and the expected value *E(X)*, as introduced in previous section. In order to calculate CDF, we include an exponent value *ExV* as in (6). Next we calculate the probability mass function PMF as in (7) and (8). Population index is denoted by x_i.

$$F(x_i) = (x_i / NP)^{ExV} \mid i=1,2,\dots,NP \tag{6}$$

$$f(1)=F(1) \tag{7}$$

$$f(x_i)=F(x_i) - F(x_{i-1}) \mid i=2,3,\dots,NP \tag{8}$$

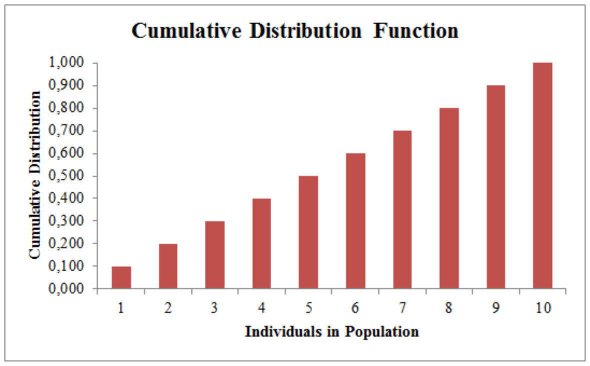
After calculating PMF, we can calculate expected value as in (9). Finally we can resolve the effective population size as in (10). Examples 1 and 2 show how *ExV* defines NP_{eff}.

$$E(X) = \sum x_i f(x_i) \tag{9}$$

$$NP_{eff} = 2\,(\,NP - E(X)\,) + 1 \tag{10}$$

A. Example 1

To achieve the same effective population size NP_{Eff} as the fixed population size *NP* (and like DE/rand/1/bin with population size *NP*), we define *ExV*=1. To keep the example simple, we choose *NP*=10. Table I reports the summary of calculations. Column 1 represents the individuals in population. Columns 2 and 3 show CDF and PMF, respectively. Column 4 shows the run of iterations, when calculating *E(X)*. Fig. 1 illustrates the CDF and PMF. From lower chart we can see clearly, how all the individuals have same probability that they are selected to next trial population. The status is now equal with DE/rand/1/bin.



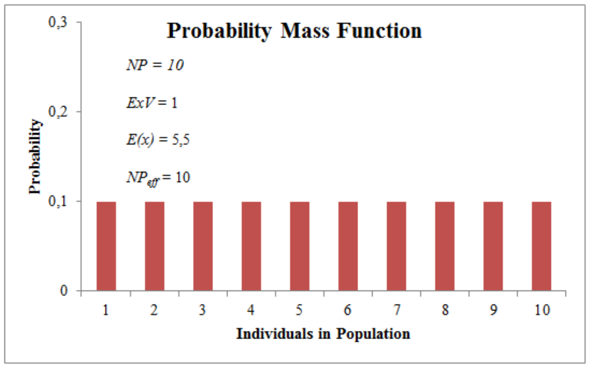


Fig. 1. CMF (upper) and PMF (lower) in Example 1, where *NP*=10, *ExV* = 1.

B. Example 2

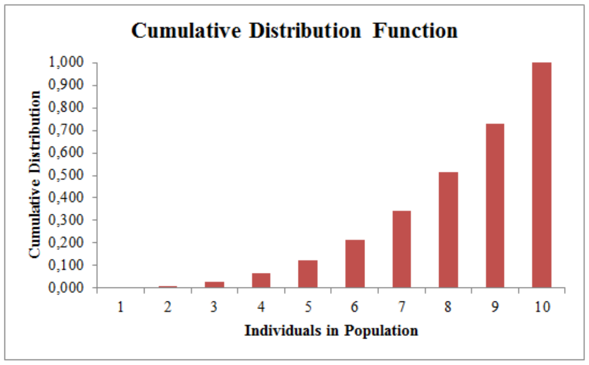
With the exponent value *ExV*=3, we can achieve an effective population size, which is approximately half of the actual population size *NP*. We choose again *NP*=10. Table II reports the summary of calculations. Fig. 2 illustrates the CDF and PMF. Lower chart shows clearly, how the individuals from the beginning of population have very small probability that they are selected to next trial population. The status would now be same as DE/rand/1/bin with *NP*=5.

TABLE I. CDF, PMF, *E(X)* AND THE EFFECTIVE POPULATION SIZE, IN EXAMPLE 1, WHEN *ExV* = 1

x	CDF	PMF	$x_i f(x_i)$
1	0,100	0,100	0,100
2	0,200	0,100	0,200
3	0,300	0,100	0,300
4	0,400	0,100	0,400
5	0,500	0,100	0,500
6	0,600	0,100	0,600
7	0,700	0,100	0,700
8	0,800	0,100	0,800
9	0,900	0,100	0,900
10	1,000	0,100	1,000
		E(X)	**5,500**
		NP $_{eff}$	**10**

TABLE II. CDF, PMF, *E(X)* AND THE EFFECTIVE POPULATION SIZE IN EXAMPLE 2, WHEN *ExV*=3

x	CDF	PMF	$x_i f(x_i)$
1	0,001	0,001	0,001
2	0,008	0,007	0,014
3	0,027	0,019	0,057
4	0,064	0,037	0,148
5	0,125	0,061	0,305
6	0,216	0,091	0,546
7	0,343	0,127	0,889
8	0,512	0,169	1,352
9	0,729	0,217	1,953
10	1,000	0,271	2,710
		E(X)	**7,975**
		NP $_{eff}$	**5,050**



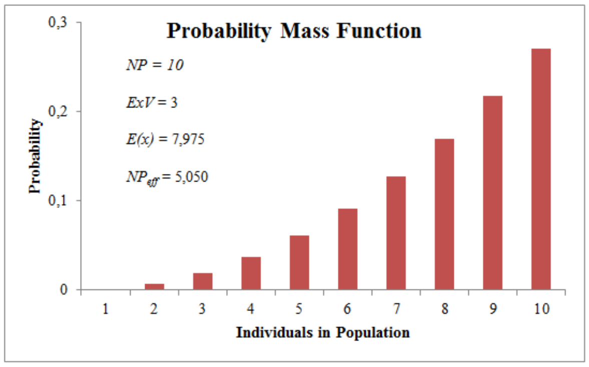


Fig. 2. CMF (upper) and PMF (lower) in Example 2, where *ExV* = 3.

Examples 1 and 2 show, that the effective population size NP_{Eff} can be adapted by changing the value of *ExV*, while the true population size *NP* is kept fixed.

In the beginning of the optimization process, the effective population size NP_{Eff} has the same value as the actual population size *NP*. When creating trial population, we choose the target vectors and three other randomly selected vectors as follows:

As target vectors we choose the most probable individuals from end of the population. For example, if the effective population size NP_{Eff} is 20 and the actual population size *NP* is 50, we choose the 20 most probable individuals for targets. In this case, the target vectors are found from indexes 31-50.

Next we choose three vectors *r1*, *r2* and *r3* randomly. The only difference here between Cumu-DE and DE/rand/1/bin is that in Cumu-DE we choose these vectors on the grounds of their probability. Fig. 3 also shows the pseudocode of this procedure more precisely.

After generating the trial population, Cumu-DE continues to select the next generation as DE/rand/1/bin. In this connection, we count every trial vector, which is strictly better (*hit*) than the comparable vector in the current population. We also count those selected trials, which are even (*equal*) with the comparable vector in current population. Variables *hit* and *equal* are needed in (11) and (12).

As told previously, we need to change the value of *ExV* to adapt the effective population size NP_{Eff}. We cannot adapt NP_{Eff} directly, because we need to update probabilities first.

```
size=round(NPeff)
targets(1:size)=((NP-size+1):NP);

for i = 1:size
{
    do
        r1=rand index between 1 and NP...
        based on its probability;
    while r1==targets(i);

    do
        r2=rand index between 1 and NP...
        based on its probability;
    while r2==targets(i)or r2==r1;

    do
        r3=rand index between 1 and NP...
        based on its probability;
    while r3==targets(i)or r3==r1 or r3==r2;
}
```

Fig. 3. Pseudocode of choosing target vectors and three randomly selected vectors *r1*, *r2* and *r3* using PMF in Cumu-DE.

However, adapting the NP_{Eff} using *ExV* is not that straightforward. The larger the effective population size is, the more even small variations of exponent value affect it, and vice versa. As a consequence, the degree of increase or decrease of *ExV* depends on the current effective population size NP_{Eff}. If we need to have larger effective population size, we update *ExV* as in (11). When smaller NP_{Eff} is needed, we use equation as in (12).

$$ExV = ExV - (1-(NP_{eff} \,/\, NP)) \qquad (11)$$

$$ExV = ExV + hit * (NP_{eff} \,/\, NP) \qquad (12)$$

The reason, why we have two different *update values* $(1-(NP_{eff}/NP))$ and (NP_{eff}/NP) is, that the smaller effective population size we have, the faster we want to increase it to avoid premature convergence. Respectively, the larger effective population size we have, the faster we want to decrease it to avoid optimization process getting too slow. Also, if we already have a small effective population size and we still get more strictly better trials, we are in no hurry to decrease the effective population size to be even smaller. The smaller effective population size we have, the less it is decreasing and vice versa. Table III shows some examples of how the update values (columns 3 and 4) affect *ExV* and NP_{Eff}.

According to our comprehensive tests, it is sufficient to check if *ExV* needs to be updated only once in every *NP*:th generation. If $hit > 0$, we increase *ExV* to get a smaller effective population. If $hit <= 1$, we decrease the *ExV* to get larger NP_{eff}. In addition, when we find selected trials which have equal value than corresponding vector in current population (*equal*), we need to update *ExV* every generation. In this case, *ExV* is decreased even more as in (13). This procedure helps to avoid premature convergence of population.

$$ExV = ExV - equal * (1-(NP_{eff} \,/\, NPx)) \qquad (13)$$

The whole procedure of adapting process can be found in Fig. 4.

```
if equal>0
   ExV=ExV-equal*(1-(NPeff/NP));
end
if mod(generation,NP)==0
   if hit>0
      ExV=ExV+(hit*((NPeff/NP)));
      //check for out-of-bounds
   end
   if hit<=1
      ExV=ExV-(1-(NPeff/NP));
      //check for out-of-bounds
   end
end
```

Fig. 4. Pseudocode of adapting process in Cumu-DE. Value of exponent *ExV* is updated.

TABLE III. EXAMPLE OF *ExV*, NP_{EFF} AND UPDATE VALUES.

***ExV* ≈**	**NP_{eff}≈**	***NPeff/N* ≈**	***1-(NPeff/N)* ≈**
1,00	50	1,00	0,00
1,22	45	0,90	0,10
1,50	40	0,80	0,20
1,86	35	0,70	0,30
2,35	30	0,60	0,40
3,00	25	0,50	0,50
4,00	20	0,40	0,60
5,68	15	0,30	0,70
9,00	10	0,20	0,80
10,20	9	0,18	0,82
11,50	8	0,16	0,84
13,40	7	0,14	0,86
15,70	6	0,12	0,88
19,20	5	0,10	0,90
24,50	4	0,08	0,92

VI. EXPERIMENTAL SETTINGS

To compare DE/rand/1/bin and Cumu-DE with each other, we used 25 benchmark functions and experimental arrangements as provided for CEC2005 special session on real-parameter optimization [4]. The most important evaluation criteria are:

- 25 ten-dimensional minimization problems.
- Runs per problem: 25.
- *Max_FES*: 20000**D*. Due to slowing down the optimization process, we define *Max_FES* larger than in [4].
- Terminate before reaching *Max_FES*, if the error in the function value is 10e-8 or less.
- In both algorithms *F* and *CR* has been initialized to 0.9, which is often considered as a robust setting without unnecessary stagnation risk.
- For DE/rand/1/bin: *NP* has been optimized similarly as in [31] by Rönkkönen et al.
- For Cumu-DE: $NP = 5 * D$, to get population size large enough and thus avoid stagnation risk.
- For Cumu-DE: $NP_{Eff} = NP$ at the start of every run.
- $1 < ExV <= NP/2$ to keep effective population size between 4 and *NP*.

To measure statistical significance between algorithms, the mean of the number of function evaluations FES was tested by Wilcoxon two-sample (sum of ranks) test and the reliability rate by the Fisher exact test for 2-by-2 contingency table as in [19]. The relative change of FES was evaluated as in (14).

$$\Delta fes = (FES_{Cumu\text{-}DE} - FES_{DE}) / FES_{DE})*100, \qquad (14)$$

where $FES_{Cumu\text{-}DE}$ and FES_{DE} are the average numbers of function evaluations of Cumu-DE and DE/rand/1/bin, respectively.

VII. RESULTS

Only those problems, which at least one algorithm could solve at least once, are presented. Table IV presents number of FES to achieve the fixed accuracy level with mean, standard deviation, success rate and success performance for both Cumu-DE and DE/rand/1/bin (Classic-DE). It also shows the relative change (in %) due to Cumu-DE (Δfes) and its significance by Wilcoxon test (*W*). Note, that negative value of Δfes means better convergence with respect to average FES. In columns *W* and *FT* (*FT*=significance by Fisher test) the symbol '+' means better performance for Cumu-DE and symbol '-' better performance for DE/rand/1/bin. If there is no significant difference at the level 0.05, the corresponding cell is left empty.

Results show that significant increase of FES was detected in six problems and significant decrease of FES in four problems. So, the Cumu-DE was slower in most cases, as expected. In 15 problems, there was no significant difference between algorithms (Table IV).

Comparison of the reliability by Fisher test shows, that Cumu-DE was significantly better in five cases, including for example Shifted Rastrigin's Function, Shifted Rotated Weierstrass Function and Schwefel's Problem. Its success rate was also better in case 6 (Shifted Rosenbrock's Function), although there were no significant difference comparing to DE/rand/1/bin. Cumu-DE was not significantly worse in any cases. In 20 problems there was no significant difference between algorithms (Table IV).

Table V shows error values achieved for problems 1 – 5. Table VI shows error values achieved for problems 6, 9, 11-12 and 15. Fig. 5 shows an example of progress of effective population size NP_{eff}, and corresponding convergence graph for function 5 median run during the optimization process.

VIII. CONCLUSIONS

In this paper we proposed an adaptive version of Differential Evolution algorithm called Cumu-DE. The algorithm is adapting the so called effective population size

NP_{eff}. The aim was to give the user an easier way to use DE, by taking care of population size automatically. The second goal was to make algorithm as reliable as possible. Both expectations were achieved. First, a user does not need to choose the population size, which is one of the three control parameters. The algorithm takes care of it. Secondly, our algorithm was significantly more reliable in five cases and less reliable in none. Additionally, Cumu-DE was significantly faster based on average of FES in four cases. It was significantly slower in six cases.

The algorithm was compared to the standard DE algorithm DE/rand/1/bin/. We used 25 benchmark functions provided by CEC2005 special session on real-parameter optimization. We used Wilcoxon two-sample sum of rank test and Fisher exact test for 2-by-2 contingency table to compare statistical significance of relative change of FES and reliability rate respectively.

Thus this mechanism with current results is justifying and motivating further studying of the proposed adaptation approach. The next step is to investigate the possibility to combine Cumu-DE with EWMA-DEFCr, which is introduced in [10]. In EWMA-DEFCr the two other control parameters; *F* and *CR* are adapted automatically based on exponentially moving average. If the combining of these two algorithms is successful, we have a control parameter free algorithm. Finally we compare these variants with the contemporarily top-rated Differential Evolution algorithms.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Optimization, Vol 11, pp. 341-359, 1997.

[2] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces", International Computer Science Institute Publications, 1995.

[3] E. Eiben, "Parameter control in evolutionary algorithms", IEEE Transactions On Evolutionary Computation, pp. 124-141, 1999.

[4] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. Chen, A. Auger and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization" Nanyang Technological University, Singapore, Technical Report, 2005.

[5] H. A. Abbass, R. Sarker and C. Newton, "PDE: A pareto-frontier differential evolution approach for multi-objective optimization problems", Proceedings of the Congress on Evolutionary Computation, pp. 971-978, 2001.

[6] H. A. Abbass and R. Sarker, "The pareto differential evolution algorithm", International Journal of Artificial Intelligence Tools, Vol. 11, pp. 531-552, 2002.

[7] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm", Soft Computing, Vol. 9, pp. 448-462, 2002.

[8] J. Aalto and J. Lampinen, "A mutation adaptation mechanism for Differential Evolution algorithm", IEEE Congress on Evolutionary Computation, pp. 55-62, 2013.

[9] J. Aalto and J. Lampinen, "A crossover adaptation mechanism for Differential Evolution algorithm," Mendel - International Conference on Soft Computing, pp. 19-24, 2013.

[10] J. Aalto and J. Lampinen, "A mutation and crossover adaptation mechanism for Differential Evolution algorithm", IEEE Congress on Evolutionary Computation, pp. 451-458, 2014.

[11] 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", IEEE Transactions on Evolutionary Computation, pp. 646-657, 2006.

[12] J. Zhang and A. C. Sanderson, "JADE: Self-adaptive differential evolution with fast and reliable convergence performance", IEEE Congress on Evolutionary Computation, pp. 2251 – 2258, 2007.

[13] J. Zhang, "JADE: Adaptive differential evolution with optional external archive", IEEE Transactions on Evolutionary Computation, pp. 945-958, 2009.

[14] J. Tvrdík, I. Křivý, and L. Mišík "Adaptive population-based search: Application to estimation of nonlinear regression parameters", Computational Statistics & Data Analysis, Vol. 52, pp. 713-724, 2007.

[15] Qin and P. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization", IEEE Congress on Evolutionary Computation, 2005.

[16] A. K. Qin, "Differential evolution algorithm with strategy adaptation for global numerical optimization", IEEE Transactions on Evolutionary Computation, pp. 398-417, 2009.

[17] R. Mallipeddi, P. N. Suganthan, Q. K. Pan and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies", Applied Soft Computing, Vol. 11, pp. 1679-1696, 2011.

[18] J. Tvrdík, "Adaptation in differential evolution: A numerical comparison", Applied Soft Computing, Vol. 9, pp. 1149-1155, 2009.

[19] J. Tvrdík, R. Poláková, J. Veselský and P. Bujok, "Adaptive variants of differential evolution: Towards control-parameter-free optimizers", Handbook of Optimization, 2013.

[20] R. Poláková and J. Tvrdík, "Competitive differential evolution algorithm in comparison with other adaptive variants", Soft Computing Models in Industrial and Environmental Applications, 2013.

[21] Y. Wang, Z. Cai and Q. Zhang "Differential evolution with composite trial vector generation strategies and control parameters", IEEE Transactions On Evolutionary Computation, Vol. 15, pp. 55-66, 2011.

[22] J. Teo, "Differential evolution with self-adaptive populations", Knowledge-Based Intelligent Information and Engineering Systems, pp. 1284-1290, 2005.

[23] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution", Soft Computing - A Fusion of Foundations, Methodologies & Applications, Vol. 10, pp. 673-686, 2006.

[24] R. Mallipeddi, "Empirical study on the effect of population size on differential evolution algorithm", IEEE Congress on Evolutionary Computation, pp. 3663 - 3670, 2008.

[25] S. T. Nga, "Self-adaptive population sizing for a tune-free differential evolution", Soft Computing - A Fusion of Foundations, Methodologies & Applications, Vol. 13, pp. 709-724 , 2009.

[26] M. Yang, "An improved adaptive differential evolution algorithm with population adaptation", Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, 2013.

[27] M. Yang, C. Li, Z. Cai and J. Guan, "Differential evolution with auto-enhanced population diversity", IEEE Transactions on Cybernetics, Vol. 45, pp. 302-315, 2015.

[28] J. Lampinen, "Global optimization by differential evolution", Optimization, 1999.

[29] A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art", Computer Methods in Applied Mechanics and Engineering, Vol. 191, pp. 1245-1287, 2002.

[30] J. Liu and J. Lampinen, "Adaptive parameter control of differential evolution", MENDEL - Proceedings of the 8th International Conference on Soft Computing, pp. 19–26 , 2002.

[31] J. Rönkkönen, S. Kukkonen and K. V. Price, "Real-parameter optimization with differential evolution", IEEE Congress on Evolutionary Computation, pp. 506-513, 2005.

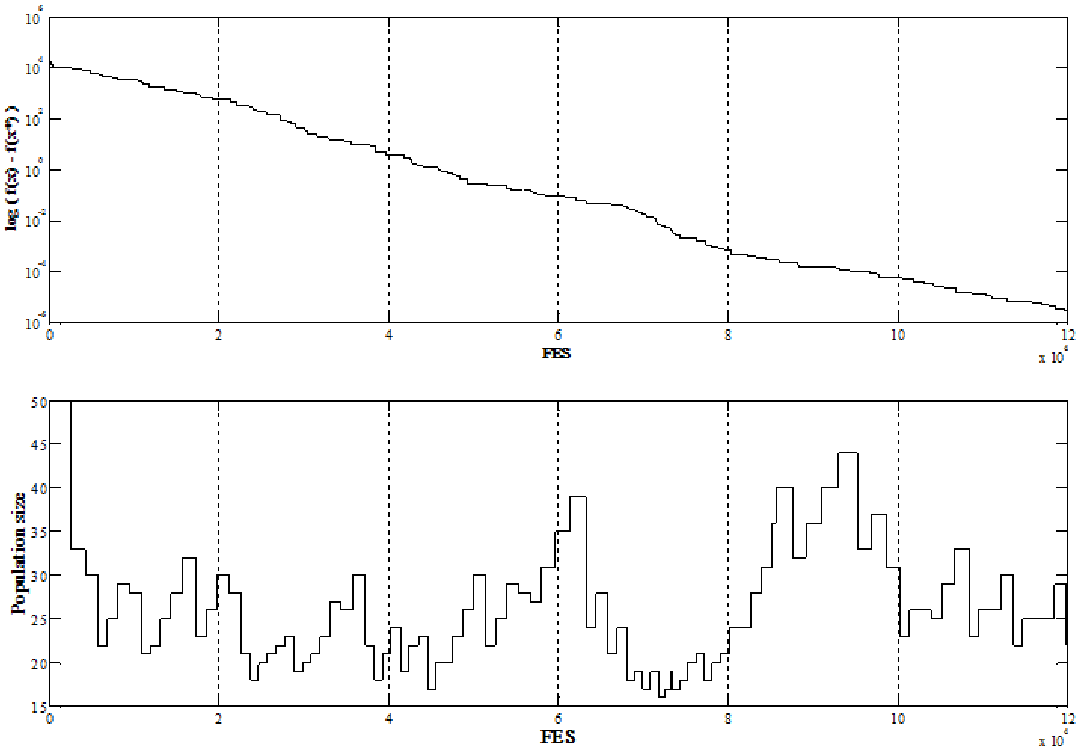


Fig. 5. Progress of convergence (upper chart) and population adaptation (lower chart) for median run of function 5 – Schwefel's Problem

TABLE IV. NUMBER OF FES, SUCCESS PERFORMANCE FOR CUMU-DE AND CLASSIC DE (DE/RAND/1/BIN), RELATIVE CHANGE OF FES DUE TO CUMU-DE AND ITS SIGNIFICANCE BY WILCOXON TEST, P-VALUES OF RELIABILITY CHANGE DUE TO CUMU-DE AND ITS SIGNIFICANCE BY FISHER TEST FOR PROBLEMS 1-6, 9, 11-12 AND 15.

Strategy	Fno	1st	7th	13th	19th	25th	Mean	Std	Su	SuccRate	SuccPerf	Δfes	W	*p* –value	FT
Cumu-DE	*1*	1,19E+04	1,70E+04	2,11E+04	2,45E+04	2,76E+04	2,09E+04	4,80E+03	25	100 %	2,09E+04	26	-	1	
Classic-DE	*1*	1,47E+04	1,59E+04	1,65E+04	1,70E+04	1,80E+04	1,65E+04	8,42E+02	25	100 %	1,65E+04				
Cumu-DE	*2*	2,84E+04	3,83E+04	4,19E+04	4,38E+04	5,86E+04	4,11E+04	6,50E+03	25	100 %	4,11E+04	79	-	1	
Classic-DE	*2*	1,97E+04	2,18E+04	2,33E+04	2,44E+04	2,57E+04	2,29E+04	1,61E+03	25	100 %	2,29E+04				
Cumu-DE	*3*	1,36E+05	1,65E+05	1,84E+05	2,00E+05	2,00E+05	1,70E+05	1,56E+04	15	60 %	2,84E+05	-12	+	9,96E-05	+
Classic-DE	*3*	1,89E+05	2,00E+05	2,00E+05	2,00E+05	2,00E+05	1,93E+05	6,12E+03	2	8 %	2,42E+06				
Cumu-DE	*4*	2,97E+04	4,76E+04	5,25E+04	5,69E+04	7,37E+04	5,18E+04	9,94E+03	25	100 %	5,18E+04	94	-	1	
Classic-DE	*4*	2,23E+04	2,50E+04	2,72E+04	2,80E+04	3,07E+04	2,67E+04	2,21E+03	25	100 %	2,67E+04				
Cumu-DE	*5*	9,94E+04	1,11E+05	1,24E+05	1,34E+05	1,61E+05	1,24E+05	1,60E+04	25	100 %	1,24E+05	105	-	1	
Classic-DE	*5*	5,36E+04	5,89E+04	6,00E+04	6,13E+04	6,78E+04	6,07E+04	3,11E+03	25	100 %	6,07E+04				
Cumu-DE	*6*	3,13E+04	3,47E+04	4,16E+04	5,09E+04	2,00E+05	4,25E+04	8,36E+03	23	92 %	4,62E+04	73	-	3,26E-01	
Classic-DE	*6*	1,67E+04	2,29E+04	2,52E+04	2,75E+04	2,00E+05	2,46E+04	3,19E+03	22	88 %	2,79E+04				
Cumu-DE	*9*	6,34E+04	1,38E+05	2,00E+05	2,00E+05	2,00E+05	1,23E+05	2,94E+04	10	40 %	3,07E+05	645	-	2,19E-03	+
Classic-DE	*9*	1,65E+04	2,00E+05	2,00E+05	2,00E+05	2,00E+05	1,65E+04	0,00E+00	1	4 %	4,12E+05				
Cumu-DE	*11*	1,28E+05	1,80E+05	2,00E+05	2,00E+05	2,00E+05	1,65E+05	2,21E+04	10	40 %	4,12E+05	-6	+	2,19E-03	+
Classic-DE	*11*	1,75E+05	2,00E+05	2,00E+05	2,00E+05	2,00E+05	1,75E+05	0,00E+00	1	4 %	4,37E+06				
Cumu-DE	*12*	2,50E+04	4,12E+04	4,67E+04	6,40E+04	2,00E+05	4,54E+04	1,08E+04	20	80 %	5,67E+04	-77	+	1,13E-09	+
Classic-DE	*12*	2,00E+05	2,00E+05	2,00E+05	2,00E+05	2,00E+05	0,00+00	0,00+00	0	0 %	0,00+00				
Cumu-DE	*15*	6,96E+04	1,08E+05	2,00E+05	2,00E+05	2,00E+05	9,82E+04	3,23E+04	9	36 %	2,73E+05	-51	+	8,12E-04	+
Classic-DE	*15*	2,00E+05	2,00E+05	2,00E+05	2,00E+05	2,00E+05	0,00+00	0,00+00	0	0 %	0,00+00				

TABLE V. ERROR VALUES ACHIEVED WHEN FES=1E3, FES=1E4, FES=1E5 AND FES=2E5 FOR PROBLEMS 1-5.

	Probl.	**Function 1**		**Function 2**		**Function 3**		**Function 4**		**Function 5**	
FES		***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***
1,00E+03	***1.st***	1,13E+03	1,04E+03	4,59E+03	3,51E+03	1,32E+07	1,57E+07	3,66E+03	2,84E+03	8,74E+03	6,22E+03
	7.th	4,37E+03	1,76E+03	7,90E+03	5,45E+03	3,63E+07	4,67E+07	9,30E+03	4,79E+03	1,14E+04	8,44E+03
	13.th	5,39E+03	2,33E+03	1,03E+04	6,41E+03	5,24E+07	5,42E+07	1,10E+04	6,63E+03	1,17E+04	9,26E+03
	19.th	6,29E+03	2,75E+03	1,15E+04	8,37E+03	7,56E+07	7,75E+07	1,30E+04	8,19E+03	1,29E+04	9,89E+03
	25.th	7,63E+03	4,08E+03	1,54E+04	1,13E+04	1,27E+08	9,80E+07	1,93E+04	1,37E+04	1,42E+04	1,16E+04
	Mean	5,15E+03	2,28E+03	9,62E+03	6,67E+03	5,76E+07	5,79E+07	1,12E+04	6,94E+03	1,19E+04	9,08E+03
	Std	1,43E+03	8,32E+02	2,67E+03	2,05E+03	3,06E+07	2,21E+07	3,48E+03	2,64E+03	1,25E+03	1,32E+03
1,00E+04	***1.st***	1,37E-04	1,29E-03	8,12E-01	1,47E-02	6,09E+04	1,54E+06	2,81E+00	2,90E-01	3,59E+02	1,28E+02
	7.th	2,55E-02	5,71E-03	6,37E+01	3,96E-01	4,10E+05	3,21E+06	7,23E+01	1,31E+00	1,25E+03	2,30E+02
	13.th	2,45E-01	7,71E-03	1,21E+02	5,61E-01	9,55E+05	4,36E+06	2,35E+02	2,23E+00	1,88E+03	2,68E+02
	19.th	1,47E+00	1,32E-02	4,75E+02	1,67E+00	2,00E+06	5,42E+06	3,47E+02	4,78E+00	3,19E+03	3,39E+02
	25.th	3,49E+01	3,07E-02	8,60E+02	7,27E+00	8,42E+06	8,98E+06	1,10E+03	2,03E+01	4,25E+03	5,29E+02
	Mean	2,92E+00	1,02E-02	2,54E+02	1,24E+00	1,62E+06	4,39E+06	2,94E+02	3,80E+00	2,13E+03	2,97E+02
	Std	7,33E+00	7,14E-03	2,80E+02	1,64E+00	1,92E+06	1,73E+06	2,83E+02	4,34E+00	1,10E+03	9,93E+01
1,00E+05	***1.st***	0,00+00	0,00+00	0,00+00	0,00+00	1,77E-03	7,03E-01	0,00+00	0,00+00	8,88E-07	0,00+00
	7.th	0,00+00	0,00+00	0,00+00	0,00+00	5,77E-02	5,08E+00	0,00+00	0,00+00	9,50E-06	0,00+00
	13.th	0,00+00	0,00+00	0,00+00	0,00+00	4,33E-01	8,38E+00	0,00+00	0,00+00	1,45E-04	0,00+00
	19.th	0,00+00	0,00+00	0,00+00	0,00+00	1,30E+01	1,94E+01	0,00+00	0,00+00	1,07E-03	0,00+00
	25.th	0,00+00	0,00+00	0,00+00	0,00+00	5,61E+02	5,11E+01	0,00+00	0,00+00	6,26E-03	0,00+00
	Mean	0,00+00	0,00+00	0,00+00	0,00+00	4,00E+01	1,48E+01	0,00+00	0,00+00	9,70E-04	0,00+00
	Std	0,00+00	0,00+00	0,00+00	0,00+00	1,16E+02	1,43E+01	0,00+00	0,00+00	1,67E-03	0,00+00
2,00E+05	***1.st***	0,00+00	0,00+00	0,00+00	0,00+00	0,00+00	1,69E-07	0,00+00	0,00+00	0,00+00	0,00+00
	7.th	0,00+00	0,00+00	0,00+00	0,00+00	0,00+00	3,58E-06	0,00+00	0,00+00	0,00+00	0,00+00
	13.th	0,00+00	0,00+00	0,00+00	0,00+00	0,00+00	1,55E-05	0,00+00	0,00+00	0,00+00	0,00+00
	19.th	0,00+00	0,00+00	0,00+00	0,00+00	7,92E-05	3,42E-05	0,00+00	0,00+00	0,00+00	0,00+00
	25.th	0,00+00	0,00+00	0,00+00	0,00+00	1,71E-02	6,65E-05	0,00+00	0,00+00	0,00+00	0,00+00
	Mean	0,00+00	0,00+00	0,00+00	0,00+00	1,95E-03	2,13E-05	0,00+00	0,00+00	0,00+00	0,00+00
	Std	0,00+00	0,00+00	0,00+00	0,00+00	5,10E-03	2,07E-05	0,00+00	0,00+00	0,00+00	0,00+00

TABLE VI. ERROR VALUES ACHIEVED WHEN FES=1E3, FES=1E4, FES=1E5 AND FES=2E5 FOR PROBLEMS 6, 9, 11-12 AND 15.

	Probl.	**Function 6**		**Function 9**		**Function 11**		**Function 12**		**Function 15**	
FES		***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***	***Cumu-DE***	***Classic DE***
1,00E+03	***1.st***	2,23E+07	1,09E+07	5,55E+01	4,63E+01	1,12E+01	9,66E+00	1,90E+04	9,55E+03	6,12E+02	6,16E+02
	7.th	2,59E+08	4,05E+07	7,19E+01	6,35E+01	1,19E+01	1,15E+01	4,28E+04	6,20E+04	7,04E+02	7,52E+02
	13.th	4,46E+08	7,13E+07	7,75E+01	6,83E+01	1,21E+01	1,18E+01	5,61E+04	7,95E+04	7,39E+02	7,82E+02
	19.th	5,36E+08	9,63E+07	8,39E+01	7,57E+01	1,25E+01	1,22E+01	7,17E+04	9,02E+04	7,73E+02	8,30E+02
	25.th	1,15E+09	4,01E+08	9,59E+01	8,48E+01	1,30E+01	1,31E+01	9,45E+04	1,33E+05	8,08E+02	8,64E+02
	Mean	4,44E+08	9,11E+07	7,71E+01	6,79E+01	1,22E+01	1,17E+01	5,78E+04	7,47E+04	7,33E+02	7,75E+02
	Std	2,68E+08	9,25E+07	9,89E+00	9,47E+00	5,22E-01	8,64E-01	1,97E+04	3,12E+04	5,32E+01	6,53E+01
1,00E+04	***1.st***	8,03E+00	9,50E+00	2,57E+01	4,96E+00	8,68E+00	8,90E+00	9,45E+01	9,55E+03	1,22E+02	4,29E+02
	7.th	1,35E+02	1,13E+01	4,02E+01	8,89E+00	1,01E+01	9,66E+00	2,70E+03	1,64E+04	3,83E+02	5,30E+02
	13.th	4,63E+02	1,81E+01	4,16E+01	1,86E+01	1,05E+01	1,03E+01	6,43E+03	1,82E+04	4,40E+02	5,54E+02
	19.th	2,56E+03	3,56E+01	4,75E+01	2,39E+01	1,09E+01	1,06E+01	9,30E+03	2,07E+04	4,97E+02	5,99E+02
	25.th	1,88E+04	1,78E+02	5,35E+01	3,92E+01	1,16E+01	1,12E+01	1,95E+04	2,40E+04	5,53E+02	6,58E+02
	Mean	3,12E+03	4,09E+01	4,22E+01	1,84E+01	1,05E+01	1,01E+01	6,84E+03	1,81E+04	4,11E+02	5,48E+02
	Std	5,35E+03	4,68E+01	6,79E+00	9,51E+00	7,23E-01	6,77E-01	5,11E+03	3,67E+03	1,26E+02	6,27E+01
1,00E+05	***1.st***	0,00+00	0,00+00	0,00+00	0,00+00	2,25E-01	3,00E-01	0,00+00	5,57E+01	0,00+00	1,89E+02
	7.th	0,00+00	0,00+00	1,24E+00	1,99E+00	4,28E+00	4,52E+00	0,00+00	3,46E+02	4,04E+01	2,10E+02
	13.th	0,00+00	0,00+00	2,98E+00	2,98E+00	8,05E+00	8,04E+00	0,00+00	5,59E+02	6,32E+01	2,28E+02
	19.th	0,00+00	0,00+00	8,47E+00	5,11E+00	9,09E+00	8,90E+00	0,00+00	8,64E+02	1,19E+02	2,57E+02
	25.th	3,99E+00	3,99E+00	1,81E+01	1,57E+01	9,51E+00	9,48E+00	1,35E+03	2,19E+03	4,32E+02	2,77E+02
	Mean	3,99E+00	3,99E+00	5,13E+00	4,12E+00	6,41E+00	6,72E+00	2,77E+02	6,92E+02	1,30E+02	2,32E+02
	Std	0,00E+00	0,00E+00	4,67E+00	3,50E+00	3,29E+00	2,83E+00	5,98E+02	5,30E+02	1,28E+02	2,60E+01
2,00E+05	***1.st***	0,00+00	0,00+00	0,00+00	0,00+00	1,28E-07	3,04E-03	0,00+00	3,74E-01	0,00+00	1,21E+02
	7.th	0,00+00	0,00+00	0,00+00	1,99E+00	5,47E-04	5,69E-02	0,00+00	1,95E+00	0,00+00	1,50E+02
	13.th	0,00+00	0,00+00	9,95E-01	2,98E+00	1,50E+00	5,30E-01	0,00+00	8,27E+00	4,15E+01	1,64E+02
	19.th	0,00+00	0,00+00	9,95E-01	5,11E+00	4,81E+00	4,73E+00	0,00+00	1,33E+01	7,54E+01	1,80E+02
	25.th	3,99E+00	3,99E+00	3,98E+00	1,57E+01	9,42E+00	9,17E+00	1,35E+03	4,46E+01	4,32E+02	2,07E+02
	Mean	3,99E+00	3,99E+00	1,47E+00	4,12E+00	2,68E+00	2,65E+00	2,77E+02	9,72E+00	1,39E+02	1,64E+02
	Std	0,00E+00	0,00E+00	9,80E-01	3,50E+00	3,27E+00	3,32E+00	5,98E+02	9,96E+00	1,41E+02	2,16E+01