Automatic design of mutation parameter adaptation for differential evolution

Vladimir Stanovov*, and *Eugene* Semenkin

Siberian Federal University, Krasnoyarsk, 660074, Russia

Abstract. In this paper the Efficient Global Optimization algorithm is applied to design the adaptation strategy for mutation parameter in Differential Evolution. The adaptation strategy is represented as a Taylor series, to allow exploring a search space of different curves. The tuning of the adaptation is performed on the L-NTADE algorithm using the benchmark of Congress on Evolutionary Computation competition on single-objective numerical optimization 2017. The experimental results show that the discovered dependence between the success rate and the parameter in current-to-pbest mutation strategy allows improving the algorithm performance in various cases.

1 Introduction

Modern Differential Evolution (DE) [1] variants represent efficient tools for global numerical optimization of black box functions. Among the winners of recent competitions on single-objective numerical optimization, most of the algorithms are based on DE, and use similar adaptation strategies [2]. The simplicity and small number of parameters were the main reasons of DE popularity, and its efficiency has been greatly improved once the success-history based adaptation was introduced in the SHADE algorithm [3].

Although there are new SHADE-based algorithms introduced every year, all of them rely on similar adaptation strategies, except some variants of the jDE algorithm [4]. One of the key features that SHADE algorithm has taken from JADE [5] is the *current-to-pbest* strategy, which has its own greediness parameter *pb*. There are known attempts to tune this parameter based on current computational resource ration in algorithms based on L-SHADE [6], such as jSO [7]. However, not all DE variants use this, and most rely on fixed parameter value.

In this study the mutation greediness parameter is connected to the success ratio in the population. In a recent study [8] it was shown that the success ratio can be an important source of information for the adaptation of scaling factor F, used in mutation. The method used in [8] included tuning the dependence between F and success rate with Taylor series, and its parameters were optimized with surrogate-assisted method Efficient Global Optimization (EGO) [9]. Such an approach allows automatic discovery of dependence between two parameters of the algorithm. Here a modified approach is used to connect the greediness parameter p and the success rate. The experiments are performed using the sets of test problems from the Congress on Evolutionary Computation competition on single-

* Corresponding author: vladimirstanovov@yandex.ru

objective black-box numerical optimization 2017 [10] and 2022 [11]. The recently proposed L-NTADE algorithm [12] is used as a baseline approach. The main finding of the study is that there is a clear and simple dependence between success rate and p, and it can be applied to other DE methods.

The rest of the paper is organized as follows. In section 2 the related work is discussed, section 3 contains the description of the proposed approach, in section 4 the experimental setup and results are presented, and section 5 concludes the paper.

2 Related work

As this paper addresses the problem of designing new parameter adaptation techniques for differential evolution, in the next subsections the DE algorithm will be briefly described, next the hyper-heuristic approach in general, and finally a short description of the EGO algorithm will be given.

2.1 Differential evolution

The differential evolution algorithm has many modifications, with SHADE-based being one of the most popular in the literature nowadays. However, based on a study on unbounded population [17], the L-NTADE algorithm was proposed, which has two populations, the first containing the newest individuals, and the second containing the best found ones. Here the description of L-NTADE algorithm will be given.

The first step of the differential evolution algorithm is initialization, where a set of N vectors with D dimensions each is generated using uniform distribution:

$$x_{i,j} = x_{lb,j} + rand \times (x_{ub,j} - x_{lb,j}),$$

where *rand* is a uniformly distributed random number in [0,1], $x_{lb,j}$ and $x_{ub,j}$ are the lower and upper boundaries of variable j, j=1,…,D.

After initialization and target function evaluation $f(x)$ for each vector, the initialized population is called the newest population x^{new}. It is copied to the second population of top individuals x^{top}. After this, the main loop of the algorithm is started and repeated until the computational resource, number of function evaluations *NFE* reaches NFE_{max}.

For every individual in the newest population the ranks are assigned: $rank_i = e^{-kp/N}$, where kp is the selective pressure parameter. Next, the mutation *r-new-to-ptop/n/t* (variant of *current-to-pbest*) is performed as follows:

$$v_{i,j} = x^{new}_{r1,j} + F \times (x^{top}_{pbest,j} - x^{new}_{i,j}) + F \times (x^{new}_{r2,j} - x^{top}_{r3,j}),$$

where F is the scaling factor, *pbest* is one of the *pb%* best individuals in the top population, *r1* and *r3* are generated randomly in [1,N], and *r2* is generated using discrete distribution with probabilities proportional to ranks $rank_i$. The scaling factor parameter is sampled before mutation using the Cauchy distribution:

$$F = randc(M_{F,k}, 0.1),$$

where $M_{F,k}$ is one of the H memory cells containing a pair of values of M_F and M_{Cr} each, k is randomly generated in [1,H].

After mutation the binomial crossover is performed as follows:

$$u_{i,j} = \begin{cases} v_{i,j} \ if \ \ rand < Cr \ or \ \ j = jrand \\ x_{i,j} \ otherwise \end{cases}$$

where *jrand* is generated in range [1,D], and Cr is sampled using normal distribution as follows:

$$Cr = randn(M_{Cr,k},\ 0.1),$$

where $M_{Cr,k}$ is the value from a memory cell.

After checking the boundary conditions with midpoint-target method, the new vector u_i is evaluated using target function, and the selection step is performed:

$$x_{nc} = \begin{cases} u_i \ if \ \ f(u_i) \le f(x_{r1}^{new}) \\ x_{nc} \ otherwise \end{cases},$$

where *nc* is an index of an individual to be replaced, *nc* is iterated every generation from 1 to *N*. If the replacement occurred, the newly generated individual is stored in a temporary pool x^{temp} until the end of the generation. After the generation end, x^{top} and x^{temp} are joined, sorted, and the best *N* individuals are saved in x^{top}. Moreover, the parameter values *F* and i used to generated u_i are saved to S_F and S_{Cr}, as well as the improvement value Δf. These values are used to update memory cell with index h, iterated every generation, same as in success-history adaptation. The size of both populations is reduced using the linear population size reduction (LPSR) method, proposed in [6].

The L-NTADE algorithm has similar features to other SHADE-based methods, such as parameter adaptation, but differs in mutation strategy, population handling and update. These changes allowed L-NTADE to show superior performance in many cases [12], that is why it was chosen as a baseline approach in this study.

2.2 Hyper-heuristics for automatic design of algorithms

Since the emergence of the evolutionary computation field, there were ideas of using evolutionary search principles to generate entire computer programs. The hyper-heuristic (HH) approach [13] aims at the automatic generation or selection of heuristics. Examples are the generation of crossover and mutation operations, and automatic search for an optimal configuration of the algorithm. Recently several attempts were made to generate parameter adaptation methods, for example in [14] and [15] the scaling factor and mutation rate adaptation schemes were designed by genetic programming (GP). In particular, these studies have shown that GP is capable of finding high-performing adaptation techniques, which differ from a widely used success-history adaptation [3]. Instead of GP, the neuroevolutionary approach can be used, as shown in [16]. Although GP and neuroevolution are useful methods in this case, some alternative can be considered, for example in [8] the combination of Taylor series and EGO algorithm was used to find parameter adaptation schemes for scaling factor *F*, crossover rate *Cr* and population size *N* in NL-SHADE-RSP algorithm.

2.3 Efficient global optimization algorithm

Most real-world problems in the black-box scenario are computationally difficult, i.e. it takes a lot of calculations to evaluate a single target function value. Thus, it makes sense to build an approximation of the target function during the search process. One of the efficient ways to do this is to use Gaussian process models, which are based on Bayesian methods. The Gaussian processes are computationally heavy, but quite precise and efficient models, which allow building high-quality approximations, and building such models is often referred to as Kriging approach.

In the EGO algorithm the Kriging model is initially build upon a set of points, generated by a Latin hypercube method. Next, the surrogate Gaussian model is used to determine the next point to evaluate goal function. In EGO, for this purpose three methods are used, namely Surrogate Based Optimization (SBO), Lower Confidence Bound (LCB) and Expected Improvement (EI). The later combines the function values and the random variable with

parameters derived from the Gaussian Process. The maximum of this surrogate model should be found, so that EGO solves an optimization problem on every step of the algorithm, for this usually a variation of the quadratic programming method is used, such as SLSQP. After the new point is determined, the target function is evaluated, and the Kriging model is built again, considering new information. This process continues until the maximum number of evaluations is reached. A more detailed explanation of all the steps of EGO can be found in [9].

3 Proposed approach

As it was shown in [8], the success rate can be an important source of information about the efficiency of the search, so there should be dependence between this value and the algorithm's parameters. The success rate is determined as follows:

$$SR = \frac{NS}{N}$$

where NS is the number of successful solutions, i.e. the number of times the selection operation was successful.

In [8] the dependence between *SR* and the mean value to sample *F* for mutation was described by a Taylor series. Here this approach is modified by introducing the minimum and maximum values, which the *pb* value controlling the mutation greediness will use. In particular, the following equations are used:

$$pb_r = c_1 + \sum_{i=2}^{11} c_i (SR - c_0)^i ,$$

$$pb = \frac{pb_r - pb_{r,\max}}{pb_{r,\min} - pb_{r,\max}} (c_u - c_l) + c_l ,$$

where pb_r is the raw value, calculated from the Taylor series using coefficients c_i, i=0,1,…10. As the raw values may have arbitrary range, the normalization step should be performed. In particular, the maximum and minimum values pb_{max} and pb_{min} are found within the [0,1] range using a simple lattice search with 0.001 step. After this, the scaling is performed using the starting value c_l and upper value c_u. As c_l and c_u are tuned by the EGO algorithm to be within [0,1] range, their setting allows generating increasing or decreasing functions. This gives the approach more flexibility compared to the method used in [8], where the generated curves had to hit 0 and 1 and the ends of the [0,1] interval.

The efficiency of the designed *pb* tuning heuristic was estimated on the CEC 2017 benchmark suite, consisting of 30 test functions [10]. The 30-dimensional case was used, as 10-dimensional is too simple, and the difference in the efficiency cannot be clearly observed. In order to compare different heuristics, i.e. different sets of c_i values, the baseline results are used, and the Mann-Whitney statistical test is performed in order to compare the new results (51 runs on 30 functions) with the baseline. The baseline is the same L-NTADE algorithm but with standard *pb* tuning. In the Mann-Whitney test normal approximation is used, so the standard score Z value is calculated for every function. The sum of all standard scores over all test functions gives the total efficiency of the generated heuristic, and is used in the EGO algorithm as target function. Although the CEC 2017 test suite was used, the evaluation criterion was changed, i.e. the ranking of the solutions was performed in the same way as in CEC 2022 benchmark [11], where the number of function evaluations was also taken into consideration. Such approach allows a more precise tuning, as it is able to differentiate algorithms with different convergence speed.

It should be noted that the efficient global optimization algorithm is not designed to work with target functions which involve randomness, however, as the experiments have shown, it is still capable of delivering high performance results.

4 Experimental setup and results

In order to perform the training phase, a set of diverse benchmark functions from CEC 2017 competition was used. The parameters of both L-NTADE and EGO are given in the next subsection. After this, the experimental results are presented.

4.1 Algorithms parameters

As mentioned before, the CEC 2017 benchmark was used for the training phase. It consists of 30 test functions defined for dimensions D = 10, 30, 50 and 100, the number of function evaluations available is set to 10^5D, 51 independent run for every function (the number of runs was set according to the competition rules). For the L-NTADE algorithm the following parameters were set [12]: initial population size N=20D, final population size in LPSR N_{min}=4, number of memory cells H=5, initial values for memory cells $M_{F,r}$=0.3, $M_{Cr,r}$=1.0, r=1,2,…H, selective pressure parameter kp=3, scaling factor adaptation bias pm=4, mutation greediness pb=0.3. For the experiments on the CEC 2022 benchmark the dimensions were set to D = 10 and 20, and the number of function evaluations to $2*10^5$ and 10^6.

For the EGO algorithm the following settings were used. The number of initial points generated by Latin hypercube: 25, the number of iterations: 975, giving a total of 1000 evaluated solutions. The search range for c_i, i=0,1,…,10 was set to [-10,10], and the range for values c_u and c_l used the [0,1] range.

The L-NTADE algorithm was implemented in C++ and ran on an OpenMPI-powered cluster with 8 processors with 8 cores each. The surrogate modelling toolbox (SMT) library was used to apply EGO algorithm in Python language. In this manner, the EGO set the parameters and ran the C++ code of the algorithm, which took these values and ran the algorithm, next the results were saved and compared to the baseline using Mann-Whitney test to get the total Z score as target function value. So, the objective function of EGO was the total standard Z score, calculated by comparing the designed heuristic on a set of 30D test problems from CEC 2017 with the baseline algorithm with standard adaptation of pbest.

4.2 Experimental results

In order to visually show the several best solutions, in Figure 1 the overall best (shown in red) and the top 25 found curves are presented, with colours from green (better) to black (worse).

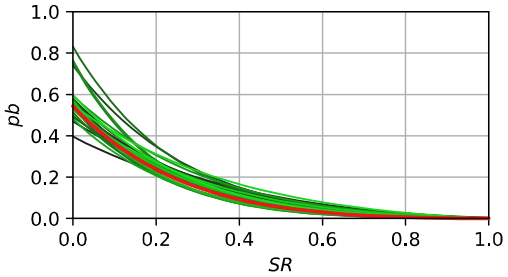

Fig. 1. The best found curves describing dependence between success rate and *pb* parameter.

As Figure 1 shows, the top 25 solutions have a similar trend in general: if the success rate is low, then the *pb* parameter should be set high. This means that if the algorithm struggles to improve any of its solutions, then the greediness parameter should be set to higher values, up to 0.5, allowing a broader search directed towards more than a half of the best individuals in current population. However, if the success rate is high, this means that the algorithm is going in the right direction, i.e. the *pb* parameter can be set to smaller values, even close to zero (only 1 best solution), when *SR*=1.

The performance comparison of the L-NTADE with *SR*-based tuning of *pb* parameters, denoted as L-NTADE$_{SRpb}$ on the CEC 2017 benchmark is shown in Table 1. The values in the table are the number of statistically significant improvements, ties and performance deteriorations (wins/ties/losses). The total standard score (Z statistics) is also provided for every dimension. Table 2 provides the same comparison on CEC 2022 benchmark.

Table 1. Comparison of L-NTADE and L-NTADE$_{SRpb}$, CEC 2017 benchmark.

Dimension	Wins/ties/losses	Total standard score
10*D*	7/22/1	47.46
30*D*	6/22/2	33.57
50*D*	7/22/1	56.58
100*D*	6/21/3	29.65

Table 2. Comparison of L-NTADE and L-NTADE$_{SRpb}$, CEC 2022 benchmark.

Dimension	Wins/ties/losses	Total standard score
10*D*	3/9/0	13.24
20*D*	1/11/0	4.59

As can be seen from Table 1, the L-NTADE$_{SRpb}$ outperforms the L-NTADE with fixed pb parameter value in all dimensions and almost all test functions. There were several performance losses observed on the most complicated composition functions with numbers 26-29, however, on most other functions the new algorithm performed better, which is also indicated by the total standard score values ($Z>2.58$ on a single function corresponds to significant change with significance level of $p=0.01$). As for the results on the CEC 2022 benchmark, the modified algorithm has shown similar or better performance, allowing getting up to three statistically significant improvements in 10*D* case, and one in 20*D* case.

To compare L-NTADE$_{SRpb}$ performance with alternative approaches, additional statistical tests were performed, shown in Tables 3 and 4. Note that here on the CEC 2017 benchmark the number of function evaluations was not taken into consideration, only the best function value found. Each cell in these tables contains the number of wins/ties/losses and total standard score.

Table 3. Comparison of L-NTADE$_{SRpb}$ with alternative algorithms, CEC 2017.

Algorithm	10D	30D	50D	100D
LSHADE-SPACMA	8/14/8 (-18.77)	13/9/8 (39.44)	13/6/11 (4.17)	12/2/16 (-29.38)
jSO	6/12/12 (-32.19)	13/14/3 (68.33)	18/9/3 (113.88)	19/6/5 (106.46)
EBOwithCMAR	4/15/11 (-54.43)	14/11/5 (52.85)	17/7/6 (77.13)	16/8/6 (76.62)
L-SHADE-RSP	5/13/12 (-37.38)	12/12/6 (48.78)	14/13/3 (90.46)	17/7/6 (84.52)
NL-SHADE-RSP	10/12/8 (-1.46)	22/5/3 (163.39)	29/0/1 (226.14)	28/1/1 (231.09)
NL-SHADE-LBC	7/14/9 (-26.70)	22/7/1 (160.60)	27/3/0 (216.56)	26/2/2 (203.71)

Table 4. Comparison of L-NTADE$_{SRpb}$ with alternative algorithms, CEC 2022.

Algorithm	10D	20D
APGSK-IMODE	3/0/9 (-37.68)	1/2/9 (-56.90)
MLS-LSHADE	8/0/4 (22.79)	5/3/4 (11.34)
MadDE	2/0/10 (-46.04)	2/2/8 (-39.89)
EA4eigN100	6/5/1 (30.61)	3/3/6 (-8.62)
NL-SHADE-RSP-MID	5/3/4 (5.33)	3/2/7 (-25.85)
L-SHADE-RSP	4/2/6 (-6.96)	1/8/3 (-9.97)
NL-SHADE-RSP	3/1/8 (-33.70)	1/3/8 (-38.39)
NL-SHADE-LBC	7/3/2 (32.99)	4/4/4 (6.55)

Table 3 shows that the L-NTADE$_{SRpb}$ algorithm performs better than most of the alternative approaches, especially in high-dimensional cases. However, it has problems with low-dimensional case, i.e. 10*D* functions. As for the CEC 2022 benchmark, the results in Table 4 demonstrate that L-NTADE$_{SRpb}$ does not always win against other algorithms. For example, one of the top methods for this benchmark, EA4eigN100, is better in 10D functions, but worse in 20*D*. In 20*D* case on the MLS-SHADE and NL-SHADE-LBC have positive total standard score against L-NTADE$_{SRpb}$. In general, it can be concluded that the proposed algorithm is capable of showing competitive performance against some of the state-of-the-art algorithms thanks to the applied modification.

5 Conclusion

In this paper the efficient global optimization algorithm was applied to perform the tuning of the Taylor series parameters of the function describing the dependence between the success rate and the mutation greediness parameter in differential evolution algorithm. The analysis of the automatically designed curves has shown that there is a clear and understandable dependence between these two parameters, and that it can be applied even to modern DE variants to improve total performance. The comparison to other algorithms on two benchmarks has shown that the modified algorithm L-NTADE$_{SRpb}$ with automatically designed heuristics is able to outperform some of the state-of-the-art algorithms on two sets of benchmark problems. Further work in this direction may include applying the same approach to find possible dependencies between DE parameters, which would allow further improvement of overall performance.

This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Grant No.075-15-2022-1121).

References

1. S. Das, S.S. Mullick, P.N. Suganthan, Swarm Evol. Comput. **27**, 1-30 (2016)
2. A.P. Piotrowski, J.J. Napiorkowski, Swarm Evol. Comput. **43**, 88-108 (2018)
3. R. Tanabe, A.S., Fukunaga, 2013 IEEE Congress on Evolutionary Computation, 71-78 (2013)
4. J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Zumer, IEEE Transactions on Evolutionary Computation **10**, 646-657 (2006)
5. J. Zhang, A.C. Sanderson, IEEE Transactions on Evolutionary Computation **13**, 945-958 (2009)

6. R. Tanabe, A.S., Fukunaga, IEEE Congress on Evolutionary Computation (CEC), 1658-1665 (2014)
7. J. Brest, M.S. Maučec, B. Bošković, IEEE Congress on Evolutionary Computation (CEC), 1311-1318 (2017)
8. V. Stanovov, E. Semenkin, Surrogate-Assisted Automatic Parameter Adaptation Design for Differential Evolution. Mathematics (2023)
9. D.R. Jones, M. Schonlau, W.J. Welch, Journal of Global Optimization **13**, 455-492 (1998)
10. N. Awad, M. Ali, J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. Technical report, Nanyang Technological University, Singapoure (2016)
11. A. Kumar, K. Price, A. W. Mohamed, A. A. Hadi, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Technical Report Nanyang Technological University Singapoure (2021)
12. V. Stanovov, S. Akhmedova, E. Semenkin, Dual-Population Adaptive Differential Evolution Algorithm L-NTADE. Mathematics (2022)
13. E.K. Burke, M. Gendreau, M.R. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Journal of the Operational Research Society **64**, 1695-1724 (2013)
14. V. Stanovov, S. Akhmedova, E. Semenkin, Knowl. Based Syst. **239**, 108070 (2022)
15. V. Stanovov, S. Semenkin, Genetic Programming for Automatic Design of Parameter Adaptation in Dual-Population Differential Evolution. Proceedings of the Companion Conference on Genetic and Evolutionary Computation (2023)
16. V. Stanovov, S. Akhmedova, E. Semenkin, Algorithms **15**, 122 (2022)
17. T. Kitamura, A.S. Fukunaga, IEEE Congress on Evolutionary Computation (CEC), 1-8 (2022)