Archive Update Strategy Influences Differential Evolution Performance

Vladimir Stanovov(✉), Shakhnaz Akhmedova, and Eugene Semenkin

Reshetnev Siberian State University of Science and Technology, "Krasnoyarskiy Rabochiy" av. 31, 660037 Krasnoyarsk, Russia
vladimirstanovov@yandex.ru, shahnaz@inbox.ru, eugenesemenkin@yandex.ru

Abstract. In this paper the effects of archive set update strategies on differential evolution algorithm performance are studied. The archive set is generated from inferior solutions, removed from the main population, as the search process proceeds. Next, the archived solutions participate in the search during mutation step, allowing better exploration properties to be achieved. The LSHADE-RSP algorithm is taken as baseline, and 4 new update rules are proposed, including replacing the worst solution, the first found worse solution, the tournament-selected solution and individually stored solution for every solution in the population. The experiments are performed on CEC 2020 single objective optimization benchmark functions. The results are compared using statistical tests. The comparison shows that changing the update strategy significantly improves the performance of LSHADE-RSP on high-dimensional problems. The deeper analysis of the reasons of efficiency improvement reveals that new archive update strategies lead to more successful usage of the archive set. The proposed algorithms and obtained results open new possibilities of archive usage in differential evolution.

Keywords: Differential Evolution · Optimization · Archive set · Mutation · CEC benchmark

1 Introduction

The heuristic optimization techniques, which include evolutionary computation and swarm intelligence algorithms, are receiving significant attention from the scientific community in recent decades, mainly due to their high efficiency and ability to solve a wide range of optimization problems [1]. Among single-objective optimization techniques, the Differential Evolution (DE) algorithms represent an interesting class of approaches, due to their high efficiency and relatively simple implementation. The DE was first introduced in a pioneering work [2], which started further research in this direction. A variety of new mutation strategies, parameter adaptation techniques and other improvements have been proposed [3], making the DE often the prize-winning algorithm in many scenarios. Despite the fact that the No Free Lunch Theorem states [4] that no optimization algorithm could be considered as the best one, still there is a room for further improvement of DE algorithms, which could be important for practitioners.

Y. Tan et al. (Eds.): ICSI 2020, LNCS 12145, pp. 397–404, 2020.
https://doi.org/10.1007/978-3-030-53956-6_35

In this paper the effects of the archive set update strategies are studied in the LSHADE-RSP algorithm, originally proposed in [5]. The Success-History based Adaptive Differential Evolution (SHADE) [6] algorithm and those based on SHADE framework use an external archive of inferior solutions to improve the exploration properties of DE by using these stored solutions during the search. As shown in [7], the algorithms based on SHADE represent one of the most efficient DE variants. Still, these algorithms rely on random update of the archive set, where a randomly selected individual is replaced if the archive size is exceeded. This study proposes 4 new archive update strategies, which replace individuals in the archive based on fitness values, and shows that most of them are capable of improving the algorithm efficiency significantly. The experiments are performed in accordance with the CEC 2020 Competition on Single-Objective Numerical Optimization setup, which, unlike previous competitions, is focused on identifying the algorithms, capable of using large computational resource efficiently. The experimental results compare all proposed archive update strategies and show their effect on overall performance and archive successful usage rates.

The rest of the paper is organized as follows: Sect. 2 describes the basics of DE and LSHADE-RSP algorithm, Sect. 3 proposes the new archive update rules, Sect. 4 contains experimental setup and results, and Sect. 5 concludes the paper.

2 Differential Evolution and SHADE Framework

The DE is a population-based evolutionary algorithm, proposed for numerical optimization. The main idea of DE is to use the scaled difference vectors between the members of the population to produce new solutions. DE starts by randomly initializing a set of NP solutions $x_{i,j}$, $i = 1, \ldots, NP$, $j = 1, \ldots, D$, inside the given boundaries $[xmin_j, xmax_j]$, where D is the problem dimension. The original DE uses the *rand/*1 mutation strategy; however, nowadays the *current-to-pbest/*1 strategy is the most widely used. It was originally proposed in the JADE algorithm [8], and further applied in SHADE framework [6]. This strategy generates mutant vector v_i using scaling factor $F \subset [0,1]$ as follows:

$$v_{i,j} = F\left(x_{pbest,j} - x_{i,j}\right) + F\left(x_{r1,j} - x_{r2,j}\right), \tag{1}$$

where *pbest* is the randomly selected index from *pb**100% best individuals, $r1$ is randomly selected from the population, and $r2$ is selected from either the population or the archive set. All indexes i, *pb*, $r1$ and $r2$ are mutually different from each other. After the mutation, the crossover is performed with probability $Cr \subset [0,1]$ as follows:

$$u_{i,j} = \begin{cases} v_{i,j} \; if \; rand(0,1) < Cr \; or \; j = jrand \\ x_{i,j} \; otherwise \end{cases}, \tag{2}$$

where *jrand* is the randomly selected index from $[1, D]$, required to make sure that at least one coordinate is inherited from the mutant vector. After the crossover, the bound constraint handling method is applied, so that all vectors u_i would be inside the boundaries. For example, the parent's position could be used as follows:

$$u_{i,j} = \begin{cases} \frac{xmin_j + x_{i,j}}{2} \; if u_{i,j} < xmin_j \\ \frac{xmax_j + x_{i,j}}{2} \; if u_{i,j} > xmax_j \end{cases}. \tag{3}$$

After this the fitness values of trial vectors $f(u_i)$ are estimated, and the selection step is performed as follows:

$$x_{i,j}^{G+1} = \begin{cases} u_{i,j}^{G} \text{ if } f\left(u_{i,j}^{G}\right) \leq f\left(x_{i,j}^{G}\right) \\ x_{i,j}^{G} \text{ otherwise} \end{cases}, \tag{4}$$

where G is the current generation number. The application of the DE mutation leads to the situation when the fitness of the population either improves or stays the same during the search.

The L-SHADE algorithm, proposed in [9], follows the ideas of JADE, and improves them by introducing the Success-History parameter adaptation with several memory cells containing pairs of successful parameter values, and the Linear Population Size Reduction (LPSR), aimed at improving the convergence properties of the algorithm closer to the end of the search. More details on SHA and LPSR could be found in [9].

In the LSHADE-RSP algorithm, proposed in [5], the usage of Rank-based Selective Pressure (RSP) is proposed. In this method, the indexes $r1$ and $r2$ in (1) have non-equal probabilities to be chosen from the population, but depend on the fitness values. Each individual receives rank value $rank_i = i$, and the probabilities p_i are calculated as follows:

$$p_i = \frac{rank_i}{\sum_{j=1}^{NP} rank_j}. \tag{5}$$

Other methods of probabilities assignment and their effect were studied in [10].

The algorithm used in this study also relies on the parameter adaptation techniques, proposed in the jSO algorithm [11], and the Distance-based Success-History Adaptation [12], described in the mentioned papers.

The SHADE class of algorithm, as well as JADE, L-SHADE and LSHADE-RSP use the external archive of inferior solutions. The archive A is initially empty, and is filled with parents – individuals $x_{i,j}$, which were replaced by their corresponding trial vectors. If the archive set size $|A|$ is equal to the maximum size, usually set to current population size, $NA = NP$, then a randomly chosen individual is replaced. The archive update rules, proposed in this study, aim at changing this update procedure.

3 Archive Update Strategies

The original archive update rule has relatively obvious drawbacks: the archive is updated with no respect to the fitness values of the individual in the archive or the added individual. This may lead to the situation, when better solutions are replaced by worse ones, or the some useless solutions are kept in the archive for too long. To resolve this issue, four new update rules are proposed.

In the first method, the random index of an individual in the archive r is generated until a worse solution is found:

```
Set attempts counter C = 0
The individual to be added is x_i
```

```
do
  r = randInt(|A|)
  C = C + 1
while(f(A_r) < f(x_i) or C < |A|)
Replace A_r with x_i
```

This algorithm replaces only worse individuals in the archive, and the search for these solutions is random. It is possible that the worst individual will not be replaced, as the search time is limited.

In the second method the worst individual in the archive is found and replaced. Although this may seem like a straightforward strategy, it may result in filling the archive with a set of very similar solutions.

In the third method, the binary tournament selection is used to find the individual to be replaced:

```
Set rand1 = randInt(|A|)
Set rand2 = randInt(|A|)
If(f(A_rand1) < f(A_rand2))
  rand1 = rand2
Replace A_rand1 with x_i
```

This approach represents a softer version of the first method, where the search for a worst solution in the archive only considers two randomly selected archived solutions, and it does not consider $f(x_i)$.

The fourth method holds a separate position for every individual with index i in the population, so that the A_i is always replaced by corresponding x_i. This strategy is somewhat similar to those used in PSO algorithms, where the best found solutions are stored separately, but here A_i holds worse solutions then x_i.

All the described archive update strategies have different ideas behind them, and should be tested against the baseline approach with random replacement. These Archive Update Strategies will be further referred to as AUS_0 (baseline) – AUS_4. The experimental setup and results are presented in the next section.

4 Experimental Setup and Results

The experiments were performed in accordance with the CEC 2020 competition rules, presented in [13]. The benchmark contains 10 functions, defined for D = 5, 10, 15 and 20, the computational resource NFE_{max} is set to $5 \cdot 10^5$ for *5D*, 10^6 for *10D*, $3 \cdot 10^6$ for 15*D* and 10^7 for 20*D*. Functions 6 and 7 are excluded from the competition for *5D*. The main announced goal of CEC 2020 is to find the approaches capable of using the large computational resource efficiently.

For every test function 30 independent runs were performed, and the best achieved goal function values were recorded after every $D^{0.25k-3}NFE_{max}$ function evaluations, $k = 0, \ldots, 15$. The population size *NP* was set to $30D^{1.5}$. The archive size *NA* was equal to *NP*, and decreased with the population size. The algorithm was implemented in C++

with GCC and run on PC with Ubuntu 19.04, Intel Core i7 8700 k processor and 48 GB RAM, results post-processing was performed using Python 3.6.

To compare the performance of different algorithms, the Mann-Whitney statistical test with significance level $p = 0.01$, tie correction, and normal distribution approximation was used. Table 1 shows the comparison of the baseline approach to newly developed AUS for all dimensions.

Table 1. Comparison of different archive update strategies, Mann-Whitney test, AUT_0 as baseline.

D	AUS_1	AUS_2	AUS_3	AUS_4
5	0+/8=/0−	0+/8=/0−	0+/8=/0−	0+/8=/0−
10	0+/10=/0−	1+/9=/0−	0+/10=/0−	0+/10=/0−
15	2+/8=/0−	1+/9=/0−	1+/9=/0−	1+/9=/0−
20	2+/8=/0−	2+/8=/0−	2+/8=/0−	1+/9=/0−

In Table 1 the "+" sign means that there was a statistically significant improvement against AUS_0, "=" means that there is no statistically significant difference, and "−" means that the performance was smaller. It could be seen that for 5D there is no difference in performance with proposed AUS, however, for larger dimensions there is a positive effect. The AUS_1 (search for worse for |A| steps), unlike other strategies, was able to deliver 2 improvements for 15D, while other methods gave only one improvement. On the other hand, the AUS_2 (replacing worst) delivered 1 improvement for *10D*. The improvements were mainly made on functions 5, 9 and 10. F5 is a composition function made of Shwefel's, Rastrigin's and High Conditioned Elliptic Function, and F9 combines Ackley's, High Conditioned Elliptic Function, Griewank's and Rastrigin's functions.

Additionally, the ranking procedure, applied in the Friedman statistical test was used to compare the algorithms. For this purpose, all the results of 5 different AUS methods were joined and sorted and ranked for every function, and the ranks were summed and normalized. This procedure was performed for all dimensions separately, and smaller ranks were assigned to better results. Table 2 contains the comparison of results.

From Table 2 it could be seen that for 5*D* the AUS_1 strategy shows the best results, however, for 10*D*, 15*D* and 20*D*, the AUS_2 (replacing worst) is better.

It is important to mention, that for 5*D*, 15*D* and 20*D* all four proposed archive update strategies are better than the baseline approach with random replacing.

To demonstrate the efficiency of the proposed archive update strategies, the Archive Successful Usage Ratio (ASUR) was calculated. The ASUR metric was defined as the number of times when the trial vector, generated with particular individual from the archive, was better than the parent; the success values were set to zero when an individual in the archive was replaced, and the ASUR value was divided by the current archive size for normalization. The ASUR metrics were averaged over 30 algorithm runs.

From Fig. 1 one may see that replacing worst individual in the archive (AUS_2) leads to many successful usages, as the search proceeds. Moreover, the stages of the

Table 2. Comparison of different archive update strategies, Friedman ranking.

D	AUS_0	AUS_1	AUS_2	AUS_3	AUS_4
5	9.8627	**8.82352**	9.60784	9.62745	9.13725
10	11.8627	11.5098	**10.7941**	12.1470	12.5098
15	12.7254	11.5000	**11.0686**	11.1960	12.3333
20	13.0490	11.8627	**10.8725**	11.2451	11.7941

search process are very well seen. Application of AUS_1 (search for worse) gives more improvements at the beginning of the search, and in the middle of the search process, a significant increase in ASUR metrics is observed. It is difficult to say what the reasons for such behavior are; however, one idea is that at this stage some of the previously visited areas, captured in the archive, appeared to be helpful for improving the search. Such peaks of ASUR metrics were observed for AUS_1 for other functions and dimensions, including F5, where significant performance improvement was observed. Another example of ASUR metrics change is presented in Fig. 2.

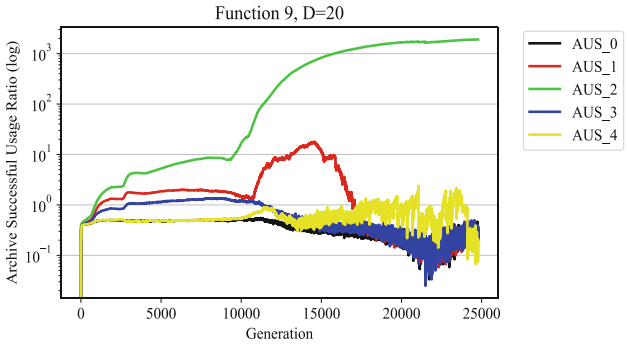

Fig. 1. Archive successful usage ratio for F9, 20D, all archive update strategies.

In Fig. 2 the change of archive successful usage ratio metrics change is shown for all five AUS methods. The difference from Fig. 1 is that here AUS_4 performs less similar to baseline method AUS_0 and relatively small modification AUS_3 (tournament-based replacement). Again, the AUS_1 method performs similarly to other methods at the beginning of the search, but further on there is a large increase in successful archive usage after 5000-th generation. Although the ASUR metrics provides important information about the archive usage dynamics, it does not consider the improvement value, so that the larger values for AUS_2 may simply mean that most of the archive set values is not replaced as often as in other archive update strategies.

The performed experiments and results post-processing show that archive update strategy significantly influences the DE performance, and should be further studied to improve the archive set usage efficiency.

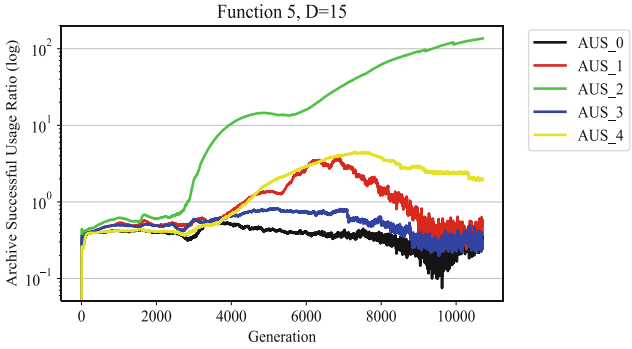

Fig. 2. Archive successful usage ratio for F5, 15D, all archive update strategies.

5 Conclusion

The archive set, used in many variants of Differential Evolution today, represent the memory of the algorithm about previous search. This memory could be successfully used during the search, so that the exploration properties of the algorithm could be improved. In this study several new archive handling techniques were proposed, in particular, the archive update strategies, and it was shown, that updating the archive by replacing the worst solution, or at least replacing the solution that is worse than current, gives significant performance improvement, and the archive set is used more efficiently. These modifications do not decrease the algorithm performance, but allow improvements on some functions. This proves that the archive handling techniques in DE should be further studied, as they represent an important part of the algorithm.

Acknowledgments. This work was supported by the Ministry of Science and Higher Education of the Russian Federation within limits of state contract № FEFE-2020-0013.

References

1. Del Ser, J., et al.: Bio-inspired computation: Where we stand and what's next. Swarm Evol. Comput. **48**, 220–250 (2019). https://doi.org/10.1016/j.swevo.2019.04.008
2. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997). https://doi.org/10.1023/A:1008202821328
3. Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution – an updated survey. Swarm Evol. Comput. **27**, 1–30 (2016)
4. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
5. Stanovov, V., Akhmedova, S., Semenkin, E.: LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2018). https://doi.org/10.1109/cec.2018.8477977
6. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 71–78 (2013)

7. Al-Dabbagh, R.D., Neri, F., Idris, N., Baba, M.S.: Algorithmic design issues in adaptive differential evolution schemes: review and taxonomy. Swarm Evol. Comput. **43**, 284–311 (2018)
8. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
9. Tanabe, R., Fukunaga, A.S.: Improving the search performance of SHADE using linear population size reduction. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1658–1665 (2014)
10. Stanovov, V., Akhmedova, S., Semenkin, E.: Selective pressure strategy in differential evolution: exploitation improvement in solving global optimization problems. Swarm Evol. Comput. **50** (2019). https://doi.org/10.1016/j.swevo.2018.10.014. ISSN 2210-6502
11. Brest, J., Maucec, M.S., Boskovic, B.: Single objective real-parameter optimization algorithm jSO. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1311–1318 (2017)
12. Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., Zamuda, A.: Distance based parameter adaptation for success-history based differential evolution. Swarm Evol. Comput. **50** (2019). https://doi.org/10.1016/j.swevo.2018.10.013
13. Yue, C.T., et al.: Problem definitions and evaluation criteria for the CEC 2020 special session and competition on single objective bound constrained numerical optimization. Technical report 201911, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, November 2019