COMPARATIVE STUDY OF THE DISTANCE/IMPROVEMENT BASED SHADE

Adam Viktorin
Roman Senkerik
Michal Pluhacek
Tomas Kadavy
Tomas Bata University in Zlin, Faculty of Applied Informatics
Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
{aviktorin, senkerik, pluhacek, kadavy}@utb.cz

KEYWORDS
Differential Evolution, SHADE, Db_SHADE, DIb_SHADE, Parameter Adaptation

ABSTRACT

In this paper, a comparative study of seven variants of the Success-History based Adaptive Differential Evolution (SHADE) algorithm is presented with the aim to show the influence of improvement and distance based parameter adaptation to its performance. Seven algorithms range from improvement only based SHADE through a balanced improvement and distance based DIb_SHADE to distance only based Db_SHADE. The algorithm set is compared on the CEC2015 benchmark set, results are statistically tested, commented and possible future research directions are provided.

INTRODUCTION

Differential Evolution (DE) is a tool for numerical optimization, which was developed by Storn and price in 1995 (Storn and Price 1995). Since then, it was studied and improved in numerous ways and a plethora of new versions appear every year. The research in the area of DE was summarized in (Neri and Tirronen 2010) and (Das et. al. 2016).

A recent trend in improving the performance of the DE is in self-adaptation of its control parameters – scaling factor F, crossover rate CR and population size NP. An algorithm that stands out from the self-adaptive DE crowd is Success-History based Adaptive Differential Evolution (SHADE) by Tanabe and Fukunaga (Tanabe and Fukunaga 2013). This algorithm is based on the JADE (Zhang and Sanderson 2009) and proposes a new way of storing of the successful scaling factor and crossover rate values in respective memories, which are periodically updated after each generation. SHADE is considered successful because it formed a basis for the last four CEC competition winners – CEC2014 L-SHADE (Tanabe and Fukunaga 2014), CEC2015 SPS-L-SHADE-EIG (Guo et. al. 2015), CEC2016 EpSin_LSHADE (Awad et. al. 2016) and CEC2017 jSO (Brest et. al. 2017).

One of the issues with SHADE algorithm is, that it suffers from premature convergence in higher dimensional spaces, and therefore a novel variant which addresses this issue has been proposed in (Viktorin et. al. 2017). The algorithm titled Db_SHADE uses the information about the distance between original and trial vectors to evaluate the weights for the parameter adaptation scheme and therefore promotes exploration of the decision space rather than its exploitation. It was shown, that this approach is beneficial for the performance of the algorithm on multi-modal complex functions.

In this paper, the weighted approach for distance and improvement based parameter adaptation is presented (DIb_SHADE) and seven SHADE algorithm variants with various preferences (improvement or distance) are compared on the CEC2015 benchmark set. The results provide an interesting insight into the performance of the algorithms and show future research direction possibilities.

The rest of the paper is structured as follows: The next section describes the basics of DE, SHADE, Db_SHADE and DIb_SHADE algorithms, the section that follows describes the experimental settings, second to the last section provides the results with discussion and the last section is devoted to concluding remarks.

FROM DE TO DIB_SHADE

In order to describe the Success-History based Adaptive Differential Evolution algorithm (SHADE) and its Distance based variants (Db_SHADE and DIb_SHADE), it is important to start from the canonical Differential Evolution (DE) by Storn and Price (Storn and Price 1995).

The canonical 1995 DE is based on the idea of evolution from a randomly generated set of solutions of the optimization task called population $\boldsymbol{P}$, which has a preset size of NP. Each individual (solution) in the population consists of a vector $\boldsymbol{x}$ of length D (each vector component corresponds to one attribute of the optimized task) and objective function value $f(\boldsymbol{x})$, which mirrors the quality of the solution. The number of

Proceedings 32nd European Conference on Modelling and Simulation ©ECMS Lars Nolle, Alexandra Burger, Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

optimized attributes D is often referred to as the dimensionality of the problem and such generated population $\boldsymbol{P}$, represent the first generation of solutions.

The individuals in the population are combined in an evolutionary manner in order to create improved offspring for the next generation. This process is repeated until the stopping criterion is met (either the maximum number of generations, or the maximum number of objective function evaluations, or the population diversity lower limit, or overall computational time), creating a chain of subsequent generations, where each following generation consists of better solutions than those in previous generations – a phenomenon called elitism.

The combination of individuals in the population consists of three main steps: Mutation, crossover and selection.

In the mutation, attribute vectors of selected individuals are combined in simple vector operations to produce a mutated vector $\boldsymbol{v}$. This operation uses a control parameter – scaling factor F. In the crossover step, a trial vector $\boldsymbol{u}$ is created by selection of attributes either from mutated vector $\boldsymbol{v}$ or the original vector $\boldsymbol{x}$ based on the crossover probability given by a control parameter – crossover rate CR. And finally, in the selection, the quality $f(\boldsymbol{u})$ of a trial vector is evaluated by an objective function and compared to the quality $f(\boldsymbol{x})$ of the original vector and the better one is placed into the next generation.

From the basic description of the DE algorithm, it can be seen, that there are three control parameters, which have to be set by the user – population size NP, scaling factor F and crossover rate CR. It was shown in (Gämperle et. al. 2002) and (Liu and Lampinen 2002), that the setting of these parameters is crucial for the performance of DE. Fine-tuning of the control parameter values is a time-consuming task and therefore, many state-of-the-art DE variants use self-adaptation in order to avoid this cumbersome task. Which is also a case of SHADE algorithm proposed by (Tanabe and Fukunaga 2013) and since it is used in this paper, the algorithm is described in more detail in the next section along with the novel distance based parameter adaptation.

1.1 SHADE

As aforementioned, SHADE algorithm was proposed with a self-adaptive mechanism of some of its control parameters in order to avoid their fine-tuning. Control parameters in question are scaling factor F and crossover rate CR. It is fair to mention, that SHADE algorithm is based on Zhang and Sanderson's JADE (Zhang and Sanderson 2009) and shares a lot of its mechanisms. The main difference is in the historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ for successful scaling factor and crossover rate values with their update mechanism.

Following subsections describe individual steps of the SHADE algorithm: Initialization, mutation, crossover, selection and historical memory update.

Initialization

The initial population $\boldsymbol{P}$ is generated randomly and for that matter, a Pseudo-Random Number Generator (PRNG) with uniform distribution is used. Solution vectors $\boldsymbol{x}$ are generated according to the limits of solution space – *lower* and *upper* bounds (1).

$$\boldsymbol{x}_{j,i} = U\big[lower_j, upper_j\big], \tag{1}$$

where i (i = 1, …, NP) is the individual index and j (j = 1, …, D) is the attribute index. The dimensionality of the problem is represented by D, and NP stands for the population size.

Historical memories are preset to contain only 0.5 values for both, scaling factor and crossover rate parameters (2).

$$M_{CR,i} = M_{F,i} = 0.5 \text{ for } i = 1, \dots, H, \tag{2}$$

where H is a user-defined size of historical memories.
Also, the external archive of inferior solutions $\boldsymbol{A}$ has to be initialized. Because of no previous inferior solutions, it is initialized empty, $\boldsymbol{A} = \emptyset$. And index k for historical memory updates is initialized to 1.

The following steps are repeated over the generations until the stopping criterion is met.

Mutation

Mutation strategy "current-to-$p$best/1" was introduced in (Zhang and Sanderson 2009) and it combines four mutually different vectors in a creation of the mutated vector $\boldsymbol{v}$. Therefore, $\boldsymbol{x}_{pbest} \neq \boldsymbol{x}_{r1} \neq \boldsymbol{x}_{r2} \neq \boldsymbol{x}_i$ (3).

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F_i\big(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i\big) + F_i(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}), \tag{3}$$

where $\boldsymbol{x}_{pbest}$ is randomly selected individual from the best $NP \times p$ individuals in the current population. The p value is randomly generated for each mutation by PRNG with uniform distribution from the range [p_{min}, 0.2] and $p_{min} = 2/NP$. Vector $\boldsymbol{x}_{r1}$ is randomly selected from the current population $\boldsymbol{P}$. Vector $\boldsymbol{x}_{r2}$ is randomly selected from the union of the current population $\boldsymbol{P}$ and external archive $\boldsymbol{A}$. The scaling factor value F_i is given by (4).

$$F_i = C\big[M_{F,r}, 0.1\big], \tag{4}$$

where $M_{F,r}$ is a randomly selected value (index r is generated by PRNG from the range 1 to H) from $\boldsymbol{M}_F$ memory and C stands for Cauchy distribution.

Therefore the F_i value is generated from the Cauchy distribution with location parameter value $M_{F,r}$ and scale parameter value of 0.1. If the generated value F_i higher than 1, it is truncated to 1 and if it is F_i less or equal to 0, it is generated again by (4).

Crossover

In the crossover step, trial vector $\boldsymbol{u}$ is created from the mutated $\boldsymbol{v}$ and original $\boldsymbol{x}$ vectors. For each vector component, a PRNG with uniform distribution is used to generate a random value. If this random value is less or equal to given crossover rate value CR_i, current vector component will be taken from a trial vector, otherwise, it will be taken from the original vector (5). There is also a safety measure, which ensures, that at least one vector component will be taken from the trial vector. This is given by a randomly generated component index j_{rand}.

$$u_{j,i} = \begin{cases} v_{j,i} & \text{if } U[0,1] \leq CR_i \text{ or } j = j_{rand} \\ x_{j,i} & \text{otherwise} \end{cases}. \quad (5)$$

The crossover rate value CR_i is generated from a Gaussian distribution with a mean parameter value $M_{CR,r}$ selected from the crossover rate historical memory $\boldsymbol{M}_{CR}$ by the same index r as in the scaling factor case and standard deviation value of 0.1 (6).

$$CR_i = N\left[M_{CR,r}, 0.1\right]. \quad (6)$$

When the generated CR_i value is less than 0, it is replaced by 0 and when it is greater than 1, it is replaced by 1.

Selection

The selection step ensures, that the optimization will progress towards better solutions because it allows only individuals of better or at least equal objective function value to proceed into the next generation $G+1$ (7).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if } f\left(\boldsymbol{u}_{i,G}\right) \leq f\left(\boldsymbol{x}_{i,G}\right) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases}, \quad (7)$$

where G is the index of the current generation.

Historical Memory Updates

Historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ are initialized according to (2), but their components change during the evolution. These memories serve to hold successful values of F and CR used in mutation and crossover steps. Successful in terms of producing trial individual better than the original individual. During every single generation, these successful values are stored in their corresponding arrays $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$. After each generation, one cell of $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ memories is updated. This cell is given by the index k, which starts at 1 and increases by 1 after each generation. When it overflows the memory size H, it is reset to 1. The new value of k-th cell for $\boldsymbol{M}_F$ is calculated by (8) and for $\boldsymbol{M}_{CR}$ by (9).

$$M_{F,k} = \begin{cases} \text{mean}_{WL}(\boldsymbol{S}_F) & \text{if } \boldsymbol{S}_F \neq \emptyset \\ M_{F,k} & \text{otherwise'} \end{cases} \quad (8)$$

$$M_{CR,k} = \begin{cases} \text{mean}_{WL}(\boldsymbol{S}_{CR}) & \text{if } \boldsymbol{S}_{CR} \neq \emptyset \\ M_{CR,k} & \text{otherwise'} \end{cases} \quad (9)$$

where $\text{mean}_{WL}()$ stands for weighted Lehmer (10) mean.

$$\text{mean}_{WL}(\boldsymbol{S}) = \frac{\sum_{k=1}^{|\boldsymbol{S}|} w_k \cdot S_k^2}{\sum_{k=1}^{|\boldsymbol{S}|} w_k \cdot S_k}, \quad (10)$$

where the weight vector $\boldsymbol{w}$ is given by (11) and is based on the improvement in objective function value between trial and original individuals in current generation G.

$$w_k = \frac{\text{abs}\left(f(\boldsymbol{u}_{k,G}) - f(\boldsymbol{x}_{k,G})\right)}{\sum_{m=1}^{|\boldsymbol{S}_{CR}|} \text{abs}\left(f(\boldsymbol{u}_{m,G}) - f(\boldsymbol{x}_{m,G})\right)}. \quad (11)$$

And since both arrays $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$ have the same size, it is arbitrary which size will be used for the upper boundary for m in (11).

The last equation (11) is the subject of change in the novel Db_SHADE algorithm, which is described in the next section.

1.2 Db_SHADE

The original adaptation mechanism for scaling factor and crossover rate values uses weighted forms of means (10), where weights are based on the improvement in objective function value (11). This approach promotes exploitation over exploration and therefore might lead to premature convergence, which could be a problem especially in higher dimensions.

Distance approach is based on the Euclidean distance between the trial and the original individual, which slightly increases the complexity of the algorithm by exchanging simple difference for Euclidean distance computation for the price of stronger exploration. In this case, scaling factor and crossover rate values connected with the individual that moved the furthest will have the highest weight (12).

$$w_k = \frac{\sqrt{\sum_{j=1}^{D} \left(u_{k,j,G} - x_{k,j,G}\right)^2}}{\sum_{m=1}^{|\boldsymbol{S}_{CR}|} \sqrt{\sum_{j=1}^{D} \left(u_{m,j,G} - x_{m,j,G}\right)^2}}. \quad (12)$$

Therefore, the exploration ability is rewarded and this should lead to avoidance of the premature convergence in higher dimensional objective spaces. Such approach might be also useful for constrained problems, where constrained areas could be overcome by increased changes of individual's components.

Below is the pseudo-code of the Db_SHADE algorithm for a clear overview.

Algorithm pseudo-code 2: Db_SHADE

```
Set NP, H and stopping criterion;
G = 0, x_best = {}, k = 1, p_min = 2/NP, A = ∅;
Randomly initialize (1) population P = (x_1,G,…,x_NP,G);
Set M_F and M_CR according to (2);
P_new = {}, x_best = best from population P;
while stopping criterion not met
  S_F = ∅, S_CR = ∅;
  for i = 1 to NP do
    x_i,G = P[i];
    r = U[1, H], p_i = U[p_min, 0.2];
    Set F_i by (4) and CR_i by (6);
    v_i,G by mutation (3);
    u_i,G by crossover (5);
    if f(u_i,G) < f(x_i,G) then
      x_i,G+1 = u_i,G;
      x_i,G → A;
      F_i → S_F, CR_i → S_CR;
    else
      x_i,G+1 = x_i,G;
    end
    if |A| > NP then randomly delete individuals from A end;
    x_i,G+1 → P_new;
  end
  if S_F ≠ ∅ and S_CR ≠ ∅ then
    Update M_F,k (8) and M_CR,k (9) with distance based weights from (12), k++;
    if k > H then k = 1, end;
  end
  P = P_new, P_new = {}, x_best = best from population P;
end
return x_best as the best found solution
```

1.3 DIb_SHADE

The DIb_SHADE algorithm is a trade-off between simple improvement based SHADE and distance based Db_SHADE. While Db_SHADE neglects the improvement in objective function value and works only with distance between solutions when it calculates parameter weights (12), DIb_SHADE combines both approaches. The resulting weight $\boldsymbol{w}$ for the parameter combination is computed as a weighted sum (13) of distance based weight $\boldsymbol{w}_d$ (11) and improvement based weight $\boldsymbol{w}_i$ (12).

$$\boldsymbol{w} = WD * \boldsymbol{w}_d + WI * \boldsymbol{w}_i, \tag{13}$$

where *WD* and *WI* are user-defined weights for distance and improvement parts of the calculation respectively. Therefore, this approach combines both explorative and exploitative weights in order to balance those two characteristics and can be tuned to optimally solve the given task.

EXPERIMENTAL SETTING

Algorithms ranging from improvement only based SHADE through a combination of distance and improvement based DIb_SHADE to distance only based Db_SHADE were tested on a basis of CEC2015 benchmark set of 15 test functions in 30*D* because, in lower dimensional spaces, the algorithm does not suffer the premature convergence. The weight pairs (*WD*, *WI*) for distance and improvement parts of the DIb_SHADE were as follows:

- (0, 1) – SHADE
- (1, 3) – DIb13_SHADE
- (1, 2) – DIb12_SHADE
- (1, 1) – DIb_SHADE
- (2, 1) – DIb21_SHADE
- (3, 1) – DIb31_SHADE
- (1, 0) – Db_SHADE

These seven versions were run with the same parameter settings:

- Population size $NP = 100$.
- Historical memory size $H = 10$.
- External archive size $|\boldsymbol{A}| = NP$.
- Stopping criterion according to CEC2015, maximum number of function evaluations $MAXFES = 10{,}000 \times D = 300{,}000$.
- Number of runs $R = 51$.

RESULTS

The resulting optimization values of the seven algorithm versions were subject to the Friedman rank statistical test to find out, whether there are significant differences. Friedman rank test yielded a p-value of 0.02, which confirmed the hypothesis, that there are significant differences and the resulting ranks for each algorithm variant are shown in Figure 1. Mean error values used for the Friedman rank test are provided in Table 1 and the best-obtained error value is highlighted in bold text.

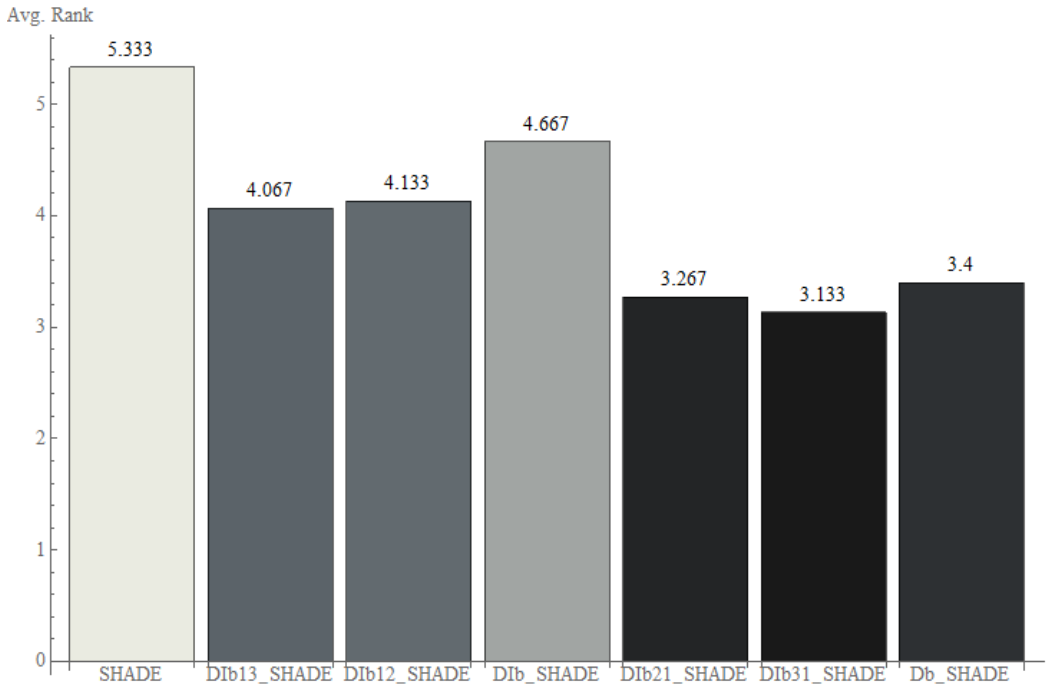

Figure 1: Friedman rank test results on CEC2015 in 30*D*.

Table 1: Mean results of 51 runs on the CEC2015 benchmark set.

f	SHADE	DIb13_SHADE	DIb12_SHADE	DIb_SHADE	DIb21_SHADE	DIb31_SHADE	Db_SHADE
1	2.62E+02	2.12E+02	1.72E+02	2.82E+02	1.94E+02	**1.37E+02**	2.42E+02
2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
3	2.01E+01	**2.01E+01**	2.01E+01	2.01E+01	2.01E+01	2.01E+01	2.01E+01
4	1.41E+01	1.36E+01	1.42E+01	1.38E+01	1.34E+01	1.34E+01	**1.31E+01**
5	1.50E+03	**1.46E+03**	1.49E+03	1.52E+03	1.51E+03	1.50E+03	1.52E+03
6	5.73E+02	6.22E+02	5.07E+02	4.80E+02	3.79E+02	4.70E+02	**3.48E+02**
7	7.26E+00	7.17E+00	6.91E+00	7.17E+00	6.83E+00	6.87E+00	**6.74E+00**
8	1.21E+02	1.01E+02	9.62E+01	1.05E+02	9.12E+01	7.84E+01	**7.38E+01**
9	1.03E+02	1.03E+02	1.03E+02	1.03E+02	1.03E+02	**1.03E+02**	1.03E+02
10	6.22E+02	5.96E+02	5.22E+02	5.72E+02	**5.18E+02**	5.87E+02	5.32E+02
11	4.50E+02	4.39E+02	4.36E+02	4.31E+02	4.27E+02	4.21E+02	**4.16E+02**
12	1.05E+02	**1.05E+02**	1.05E+02	1.05E+02	1.05E+02	1.05E+02	1.05E+02
13	9.50E+01	**9.40E+01**	9.59E+01	9.45E+01	9.42E+01	9.46E+01	9.50E+01
14	3.24E+04	3.24E+04	3.24E+04	3.23E+04	3.26E+04	**3.22E+04**	3.24E+04
15	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**	**1.00E+02**

It can be seen, that the best rank was obtained by DIb31_SHADE variant closely followed by DIb21_SHADE and Db_SHADE. In the previous study (Viktorin et. al. 2017), Db_SHADE's improved performance was associated with the ability to maintain population diversity for longer optimization period and therefore, it would be only reasonable to assume that the ranks would be linearly decreasing towards distance only based SHADE variant (Db_SHADE). However, the results show that incorporating improvement factor into the weight calculation with lower importance can bring overall performance improvement (DIb31_SHADE and DIb21_SHADE). Additional Wilcoxon rank-sum tests were performed on the selected pairs of SHADE variants in order to obtain clearer answers to the performance question. These results are summarized in a form of wins/loses/draws triplets in Table 2. The significance level of the Wilcoxon rank-sum test was set to 5%.

From the Table 2, it is clear that Db_SHADE algorithm significantly outperforms the DIb31_SHADE on one test function and algorithms draw on the rest of the benchmark. This fact is in contradiction with their respective ranks from the Friedman test. It is also visible that introduction of the distance based parameter adaptation to the SHADE is beneficial in all cases. One of the reasons for surprising results in Friedman ranks is that the ranking is based on the mean error values of 51 runs on each test function. Therefore, even when the difference in obtained error values is not significant, the ranking system will prefer the smaller value and assign it with smaller rank.

Table 2: Selected pairwise comparisons.

Algorithm pair	wins/loses/draws
SHADE – DIb31_SHADE	0/4/11
SHADE - DIb_SHADE	0/2/13
SHADE - Db_SHADE	0/5/11
Dib31_SHADE - DIb_SHADE	2/0/13
Dib31_SHADE - Db_SHADE	0/1/14
DIb_SHADE - Db_SHADE	0/3/12

Another interesting aspect is that higher values of the distance weight factor *WD* appear to have a higher impact on the performance. This might be due to increased differences between distance weights by the given factor *WD* and therefore, originally significant weights will have an even higher influence to the weighted Lehmer mean computation in comparison with weights from the other side of the spectrum (10). This phenomenon is an attractive future research direction for the authors.

CONCLUSION

This paper presented a comparative study of the effect of improvement and distance based parameter adaptation to the overall performance of the SHADE algorithm. On the seven algorithm variants, ranging from improvement only based SHADE through balanced improvement and distance based DIb_SHADE to distance only based Db_SHADE, it was shown that introduction of the distance based parameter adaptation to the weight computation in historical memory update is a beneficial step. However, there is also a number of open questions in the weighting of both improvement and distance factors. These provide an interesting future research direction for the authors and will be addressed in their future work.

ACKNOWLEDGMENT

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme Project no. LO1303 (MSMT-7778/2014), further by the European Regional Development Fund under the Project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the Projects no. IGA/CebiaTech/2018/003. This work is also based upon support by COST (European Cooperation in Science & Technology) under Action CA15140, Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO), and Action IC1406, High-Performance Modelling and Simulation for Big Data Applications (cHiPSet). The work was further supported by resources of A.I.Lab at the Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

REFERENCES

Awad, N. H., Ali, M. Z., Suganthan, P. N., & Reynolds, R. G. (2016, July). An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems. In Evolutionary Computation (CEC), 2016 IEEE Congress on (pp. 2958-2965). IEEE.

Brest, J., Maučec, M. S., & Bošković, B. (2017, June). Single objective real-parameter optimization: Algorithm jSO. In Evolutionary Computation (CEC), 2017 IEEE Congress on (pp. 1311-1318). IEEE.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution–An updated survey. Swarm and Evolutionary Computation, 27, 1-30.

Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study for differential evolution. Advances in intelligent systems, fuzzy systems, evolutionary computation, 10, 293-298.

Guo, S. M., Tsai, J. S. H., Yang, C. C., & Hsu, P. H. (2015, May). A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set. In Evolutionary Computation (CEC), 2015 IEEE Congress on (pp. 1003-1010). IEEE.

Liu, J., & Lampinen, J. (2002). On setting the control parameter of the differential evolution method. In Proceedings of the 8th international conference on soft computing (MENDEL 2002) (pp. 11-18).

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review, 33(1-2), 61-106.

Storn, R., & Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces (Vol. 3). Berkeley: ICSI.

Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. In Evolutionary Computation (CEC), 2013 IEEE Congress on (pp. 71-78). IEEE.

Tanabe, R., & Fukunaga, A. S. (2014). Improving the search performance of SHADE using linear population size reduction. In Evolutionary Computation (CEC), 2014 IEEE Congress on (pp. 1658-1665). IEEE.

Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T. & Zamuda, A. (2017) Distance Based Parameter Adaptation for Differential Evolution. In Computational Intelligence (SSCI), 2017 IEEE Symposium Series on (pp. 2612-2618) IEEE. In press.

Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, 13(5), 945-958.

AUTHOR BIOGRAPHIES

ADAM VIKTORIN was born in the Czech Republic in 1989, and went to the Faculty of Applied Informatics at Tomas Bata University in Zlín, where he studied Computer and Communication Systems and obtained his MSc degree in 2015. He is studying his Ph.D. at the same University and the fields of his studies are: Artificial Intelligence, Data Mining and Evolutionary Computation. His most recent professional "hobby" is a development and analysis of self-adaptive strategies for Differential Evolution in the area of numerical optimization; and also application of such algorithms to real-world problems. His email address is: aviktorin@utb.cz

ROMAN SENKERIK received his Ph.D. degree in Technical Cybernetics from the Tomas Bata University in Zlin, Czech Republic in 2008. He is currently an associated professor at the Tomas Bata University in Zlin, Faculty of Applied Informatics. His research interests include interdisciplinary applications of evolutionary computation, modification and development of evolutionary and swarm based algorithms, computational intelligence, optimization, cyber-security, theory of chaos, emergence and complexity. His email address is: senkerik@utb.cz

MICHAL PLUHACEK received his Ph.D. degree in Information Technologies from the Tomas Bata University in Zlin, the Czech Republic in 2016 with the dissertation topic: Modern method of development and modifications of evolutionary computational techniques. Currently works as a junior researcher at the Regional Research Centre CEBIA-Tech of Tomas Bata University in Zlin. His research focus includes swarm intelligence theory and applications and artificial intelligence in general. His email address is: pluhacek@utb.cz

TOMAS KADAVY was born in the Czech Republic in 1990, and went to the Faculty of Applied Informatics at Tomas Bata University in Zlin, where he studied Information Technologies and obtained his MSc degree in 2016. He is studying his Ph.D. at the same University and the fields of his studies are: swarm based algorithms, computational intelligence and optimization. His email address is: kadavy@utb.cz