DISH–XX Solving CEC2020 Single Objective Bound Constrained Numerical Optimization Benchmark

1st Adam Viktorin
Faculty of Applied Informatics
Tomas Bata University in Zlin
T. G. Masaryka 5555, 760 01 Zlin, Czech Republic
aviktorin@utb.cz

2nd Roman Senkerik
Faculty of Applied Informatics
Tomas Bata University in Zlin
T. G. Masaryka 5555, 760 01 Zlin, Czech Republic
senkerik@utb.cz

3rd Michal Pluhacek
Faculty of Applied Informatics
Tomas Bata University in Zlin
T. G. Masaryka 5555, 760 01 Zlin, Czech Republic
pluhacek@utb.cz

4th Tomas Kadavy
Faculty of Applied Informatics
Tomas Bata University in Zlin
T. G. Masaryka 5555, 760 01 Zlin, Czech Republic
kadavy@utb.cz

5th Aleš Zamuda
Faculty of Electrical Engineering and Computer Science
University of Maribor
Koroška cesta 46, 2000 Maribor, Slovenia
ales.zamuda@um.si

***Abstract*—This paper proposes a competitor to the CEC2020 single objective bound constrained numerical optimization competition – DISH–XX. The DISH–XX algorithm is based on its 2019 predecessor DISH. The main difference lies in the secondary crossover with the archive of historically best–found solutions. The results of the DISH–XX algorithm are presented in the competition specified format and the statistical comparison between DISH–XX and the original DISH is also presented as part of this paper.**

***Index Terms*—Differential Evolution, DISH, DISH–XX, crossover, CEC2020, competition, benchmark**

I. Introduction

The Differential Evolution (DE) algorithm [1], and especially its recent variants (mentioned in the next paragraph) have, over the past decade, proven its superior performance in numerous numerical single objective competitions. DE is, therefore, taken as a robust algorithm for numerical optimization. As can be seen in a recent survey [2], the go–to methods that successfully improve the overall performance are based on adaptivity or self–adaptivity [3] of the control parameters – scaling factor F and crossover rate CR. Also, management of the population size during the optimization phase is usually beneficial for the algorithm's performance [4]–[11]. DE–based algorithms using control parameter and population size adaptation are listed with their corresponding rankings below.

In 2013, algorithm with adaptive control parameters Success–History based Adaptive Differential Evolution (SHADE) [12] was introduced and evaluated on the CEC2013 benchmark [13]. It placed 4th as the best–performing DE–based algorithm. A year after that, updated version with linear population size decrease L–SHADE [9] won the CEC2014 competition [14]. The CEC2015 learning based scenario [15] was won by the L–SHADE Incorporated with Eigenvector–Based Crossover and Successful–Parent–Selecting Framework (SPS–L–SHADE–EIG) [16]. The CEC2016 single parameter and operator set scenario [14] had two winners – one of them was Ensemble Sinusoidal Parameter Adaptation incorporated with L–SHADE (LSHADE_EpSin) [17]. In 2017, L–SHADE–based algorithms placed on 2nd [18], 3rd [19] and 4th [20] place of the CEC2017 bound constrained competition [21], similarly, 2nd [22] and 3rd (ELSHADE_SPACMA) place on the CEC2018 competition [21] was held by L–SHADE–based algorithms. The CEC2019 single objective competition [23] brought a new format titled 100–digit and once again, algorithms based on DE ranked well – 1st place jDE100 [24], 2nd place DISHchain1e+12 [25] and 3rd place HyDE–DF [26].

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/2020/001. 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). 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).

Since authors of this paper have, also, co–authored DISH algorithm [27] (2nd and 9th place on the CEC2019 competition), the basis for the proposed algorithm in this paper is also DISH. The novelty of the proposed DISH–XX algorithm lies in the use of secondary crossover with a randomly selected individual from the historically best–found solutions in the optimization run. The initial population size of the DISH–XX have also been altered, after numerous experiments, to twice the original size.

The complete description of the DISH–XX algorithm can be found in the next section; section III describes the experimental setting, section IV shows results on the CEC2020 benchmark according to the benchmark rules and also provides the comparison with the original DISH algorithm. The paper is concluded in section V with suggestions for future work.

II. DISH–XX

The DISH–XX algorithm is an evolutionary step of the DISH algorithm presented in 2019 [27], and both share the ancestry line from the original 1995 DE [1]. The simplified concept of the DE algorithm can be described in a few steps – start with a random population of solutions. Iteratively produce candidate solutions via mutation and crossover. If the quality of the candidate solution is better than the quality of the original solution, place it into the next generation. Do this over and over, until the stopping criterion is met and return the best–found solution as the solution to the optimized problem. In the original version of the DE, there were three user-defined control parameters – population size *NP*, scaling factor *F* and crossover rate *CR*. The values of these parameters are usually adapted during the optimization in the modern versions of the DE, and DISH–XX is no exception. But, the mutation operator and adaptation mechanisms evolved via several successful algorithms. The evolution line from DE to DISH–XX is described in the following steps:

1) DE from 1995 by Storn and Price [1].
2) JADE from 2009 [28] – algorithm created by Zhang and Sanderson proposed a novel mutation strategy – current–to–*p*best/1 with an optional archive of inferior solutions.
3) SHADE from 2013 by Tanabe and Fukunaga [12] – built on the JADE algorithm with added memories for historically successful *F* and *CR* values and new adaptation mechanism for these parameters. This algorithm placed 3rd in the CEC 2013 competition.
4) The linear decrease of population size was introduced into SHADE and created L–SHADE algorithm [9], the winner of the CEC 2014 competition.
5) Improved L-SHADE algorithm titled iL–SHADE [29] was proposed for a CEC 2016 competition by Brest et al. This algorithm introduced changes to the historical memory update system and the initialization of the historical memories. It also proposed a new mechanism for treating *F* and *CR* parameters based on the ratio between current and maximum generation (phase of the optimization). This algorithm placed 4th in the CEC 2016 competition.
6) Distance based parameter adaptation was proposed for SHADE based algorithms by Viktorin et al. in 2017 [30]. This novel adaptation mechanism based on the distance between solutions instead of on the difference between objective function value was presented on SHADE and L–SHADE algorithms and shown its superiority over the original.
7) jSO algorithm was proposed by Brest et al. in 2017 [18]. The algorithm uses a novel current–to–*p*best–w/1 mutation strategy and slightly changes fixed values for *F* and *CR* parameters. The jSO algorithm was 2nd in the CEC 2017 bound constrained competition.
8) DISH algorithm was introduced in 2018 by Viktorin et al. and published in 2019 [27], and it incorporates the distance based parameter adaptation into the jSO algorithm to improve its performance.
9) DISH–XX algorithm proposed in this study incorporates secondary crossover into the DISH algorithm and uses two times larger initial population.

The following subsections provide the details of DISH–XX algorithm mechanisms followed by a pseudo–code.

A. Initialization

First of all, the initial population $\boldsymbol{P}$, of solutions to the optimized problem, is generated randomly. The size of the population is determined by the user via NP_{init} parameter (initial population size). Each individual solution $\boldsymbol{x}$ is a vector of length *D*, which is a dimension of the problem and each vector component is generated within its lower *lo* and upper *up* bounds by a uniform pseudo–random number generator (1).

$$\boldsymbol{x}_{j,i} = U\left[lo_j,\ up_j\right] \text{ for } j = 1,\ \ldots,\ D; i = 1,\ \ldots,\ NP_{init} \tag{1}$$

Other parameters and variables that have to be set in the initialization phase are:

1) Final population size – NP_f.
2) Stopping criterion – a maximum number of objective function evaluations *MAXFES* in the most common case (also in this study).
3) p_{max} and p_{min} parameters for mutation operator. p_{max} = 0.25 and p_{min} = p_{max}/2 = 0.125
4) External archive $\boldsymbol{A}$ is initialized empty. $\boldsymbol{A}$ = Ø
5) Historical best archive $\boldsymbol{A}_{best}$ is initialized empty. $\boldsymbol{A}_{best}$ = Ø
6) Historical memory size *H*. *H* = 5
7) Historical memories for scaling factor $\boldsymbol{M}_F$ (2) and crossover rate $\boldsymbol{M}_{CR}$ (3).
8) Update historical memory index *k*. *k* = 1.

$$\boldsymbol{M}_{F,i} = 0.5 \text{ for } i = 1, \ldots, H-1,\ \boldsymbol{M}_{F,H} = 0.9 \tag{2}$$

$$\boldsymbol{M}_{CR,i} = 0.8 \text{ for } i = 1, \ldots, H-1,\ \boldsymbol{M}_{CR,H} = 0.9 \tag{3}$$

The following steps – mutation, crossover, and selection are repeated for each individual solution in the generation *G*, and these generations are repeated until the stopping criterion is met.

B. Mutation

The mutation operator used in DISH–XX is a jSO's current–to–*p*best–w/1, which combines a greedy approach in the first difference and the explorative factor in the second difference (4).

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F_{w,i}\left(\boldsymbol{x}_{pBest} - \boldsymbol{x}_i\right) + F_i\left(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}\right) \quad (4)$$

The $\boldsymbol{v}_i$ is the *i*–th mutated vector created from current solution vector $\boldsymbol{x}_i$, one of the $100p\%$ best solutions in the population $\boldsymbol{x}_{pBest}$ where p is determined by (5), a random solution from the population $\boldsymbol{x}_{r1}$ and random solution from the union of the population and external archive $\boldsymbol{x}_{r2}$. It is also important to note that all vectors are mutually different – $\boldsymbol{x}_i \neq \boldsymbol{x}_{pBest} \neq \boldsymbol{x}_{r1} \neq \boldsymbol{x}_{r2}$. The differences are scaled by two scaling factor parameters, scaling factor F_i (6) and weighted scaling factor $F_{w,i}$ (8).

$$p = FES_{ratio} * (p_{max} - p_{min}) + p_{min} \quad (5)$$

where FES_{ratio} stands for the ratio between the current number of objective function evaluations *FES* and the maximum number of objective function evaluations *MAXFES* (FES_{ratio} = *FES*/*MAXFES*). Therefore, parameter p increases linearly with objective function evaluations.

$$F_i = C\left[\boldsymbol{M}_{F,r}, 0.1\right] \quad (6)$$

The scaling factor value F_i is generated from Cauchy distribution with the location parameter $\boldsymbol{M}_{F,r}$ and scale parameter value of 0. The index *r* is randomly generated from the range [1, *H*]. If the generated value F_i is smaller or equal to 0, it is generated again and if it is higher than 1, it is set to 1. Also, the scaling factor F_i is influenced by the FES_{ratio} in order to truncate its value in the exploration phase of the algorithm run (7).

$$F_i = 0.7, \; FES_{ratio} < 0.6 \text{ and } F_i > 0.7 \quad (7)$$

$$F_{w,i} = \begin{cases} 0.7 * F_i, & FES_{ratio} < 0.2 \\ 0.8 * F_i, & FES_{ratio} < 0.4 \\ 1.2 * F_i, & \text{otherwise} \end{cases} \quad (8)$$

The weighted scaling factor $F_{w,i}$ is based on the optimization phase given by the FES_{ratio}.

The next step after the mutation is the crossover.

C. Double Crossover

The DISH–XX algorithm uses two crossovers. First classical and second newly proposed in order to use historically successful parameter values that might be forgotten during the optimization. Secondary crossover is also helpful for maintaining population diversity.

The first crossover operator in DISH–XX algorithm is binomial and is based on the crossover rate value CR_i generated from the normal distribution (9) with a mean parameter value $\boldsymbol{M}_{CR,r}$ selected from the crossover rate historical memory and standard deviation value of 0.1.

$$CR_i = N\left[\boldsymbol{M}_{CR,r}, 0.1\right] \quad (9)$$

The CR_i value is also bounded between 0 and 1 and whenever it is generated outside these bounds, it is truncated to the nearest bound. The crossover rate value is also a subject to the optimization phase given by FES_{ratio} (10).

$$CR_i = \begin{cases} \max(CR_i, 0.7), & FES_{ratio} < 0.25 \\ \max(CR_i, 0.6), & FES_{ratio} < 0.5 \\ CR_i, & \text{otherwise} \end{cases} \quad (10)$$

And finally, the binomial crossover is depicted in (11).

$$\boldsymbol{u}^*_{j,i} = \begin{cases} \boldsymbol{v}_{j,i} & \text{if } U\left[0,1\right] \leq CR_i \text{ or } j = j_{rand} \\ \boldsymbol{x}_{j,i} & \text{otherwise} \end{cases} \quad (11)$$

where $\boldsymbol{u}^*_i$ is called a temporary trial vector and j_{rand} is an index of one component that has to be taken from the mutated vector $\boldsymbol{v}_i$. The j_{rand} index ensures that at least one vector component of the original vector $\boldsymbol{x}_i$ will be replaced. Thus in the following selection step, the tested trial vector will provide new information.

The second crossover is also binomial and uses the same crossover rate CR_i value. The difference is that it combines temporary trial vector $\boldsymbol{u}^*_i$ with one of the historically best found solution vectors $\boldsymbol{x}_{rAbest}$ randomly selected from historical best archive A_{best} to obtain trial vector $\boldsymbol{u}_i$ (12).

$$\boldsymbol{u}_{j,i} = \begin{cases} \boldsymbol{u}^*_{j,i} & \text{if } U\left[0,1\right] \leq CR_i \text{ or } j = j_{rand} \\ \boldsymbol{x}_{j,rAbest} & \text{otherwise} \end{cases} \quad (12)$$

D. Selection

In the selection step, a quality of the trial solution vector $\boldsymbol{u}_i$ is compared to the quality of the original solution vector $\boldsymbol{x}_i$. The quality is given by the objective function value of these solutions. And since the selection operator is elitist, the trial solution has to have at least equal objective function value as the original solution in order to proceed into the next generation *G*+1 (13).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if } f\left(\boldsymbol{u}_{i,G}\right) \leq f\left(\boldsymbol{x}_{i,G}\right) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} \quad (13)$$

where $f()$ depicts the objective function value, and in this case, the objective is the minimization of it.

The mutation, crossover and selection operators are repeated for each individual solution in the population, and after the population is exhausted, the algorithm proceeds to the next generation. But before processing each individual solution of the next generation, two essential mechanisms are incorporated into the algorithm – a linear decrease of the population size and the update of historical memories. These two mechanisms are described in the following subsections.

E. Linear Decrease of the Population Size

The population size is decreased during the algorithm run in order to provide more time for exploration in the later phase of optimization. Thus, the smaller population of individual solutions will have more time to exploit promising areas of the objective function landscape.

The mechanism used in the DISH–XX algorithm is a simple linear decrease of population size, which uses the information of current objective function evaluations to shrink the population of solutions. A new population size NP_{new} is calculated as follows (14).

$$NP_{new} = \text{round}\left(NP_{init} - FES_{ratio} * (NP_{init} - NP_f)\right) \tag{14}$$

The size of an external archive $\boldsymbol{A}$ is connected to the size of the population, and therefore, after decreasing the population size, the archive size is reduced as well. Whereas when decreasing the population size, the worst individual solutions are discarded from the population, in the archive, solutions to discard are selected randomly.

F. Update of Historical Memory

Historical memories $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ store historically successful values of scaling factors *F* and crossover rates *CR* that were helpful in the production of better trial individual solutions. Therefore, these memories have to be updated during the optimization in order to store recently used values. After each generation, one cell of both memories is updated, and for that, the algorithm uses index *k* to remember, which cell will be updated. The index is initialized to 1, and therefore, after the first generation, the first memory cell will be updated. The index is increased by one after each update, and when it overflows the memory size *H*, it starts from 1 again. There is one exception to the update, the last cell of both memories is never updated and still contains values 0.9 for both control parameters.

What will be stored in the *k*–th cell after the generation *G* is computed by a weighted Lehmer mean (15) of corresponding generation control parameter arrays $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$. These arrays are filled during the generation by the values of control parameters when the trial solution succeeds in the selection step.

$$\text{mean}_{WL}(\boldsymbol{S}) = \frac{\sum_{n=1}^{|\boldsymbol{S}|} w_n \bullet \boldsymbol{S}_n^2}{\sum_{n=1}^{|\boldsymbol{S}|} w_n \bullet \boldsymbol{S}_n} \tag{15}$$

The $\text{mean}_{WL}()$ stands for weighted Lehmer mean and the computation is equal for both $\boldsymbol{S}_F$ and $\boldsymbol{S}_{CR}$, therefore, there is no subscript for $\boldsymbol{S}$ in the equation. The *k*–th memory cells of $\boldsymbol{M}_F$ and $\boldsymbol{M}_{CR}$ are then updated according to (16) and (17).

$$\boldsymbol{M}_{F,k} = \begin{cases} \text{mean}_{WL}(\boldsymbol{S}_F) & \text{if } \boldsymbol{S}_F \neq \emptyset \text{ and } k \neq H \\ \boldsymbol{M}_{F,k} & \text{otherwise} \end{cases} \tag{16}$$

$$\boldsymbol{M}_{CR,k} = \begin{cases} \text{mean}_{WL}(\boldsymbol{S}_{CR}) & \text{if } \boldsymbol{S}_{CR} \neq \emptyset \text{ and } k \neq H \\ \boldsymbol{M}_{CR,k} & \text{otherwise} \end{cases} \tag{17}$$

The weights for the weighted Lehmer means (15) are in the case of DISH–XX algorithm computed as depicted in (18). This weighting was introduced as the distance based parameter adaptation [4]. It is titled like that, because in the original SHADE, L–SHADE, iL–SHADE and jSO algorithms, the weights were based on the difference between objective function values of trial individual solution $\boldsymbol{u}_i$ and its corresponding original individual solution $\boldsymbol{x}_i$, whereas in DISH–XX, the weight is computed from the Euclidean distance between those two - $\boldsymbol{u}_i$ and $\boldsymbol{x}_i$.

$$w_n = \frac{\sqrt{\sum_{j=1}^{D} \left(\boldsymbol{u}_{n,j,G} - \boldsymbol{x}_{n,j,G}\right)^2}}{\sum_{m=1}^{|\boldsymbol{S}_{CR}|} \sqrt{\sum_{j=1}^{D} \left(\boldsymbol{u}_{m,j,G} - \boldsymbol{x}_{m,j,G}\right)^2}} \tag{18}$$

This approach promotes exploitation and tries to avoid the premature convergence of the algorithm into local optima.

III. Experimental Setting

Settings for the experiment are given by the benchmark itself – 10 single objective functions in four different dimensional settings (5, 10, 15 and 20) with stopping criterion given by the number of maximum objective function evaluations (50,000, 1,000,000, 3,000,000 and 10,000,000 respectively). In order to obtain reasonable statistical results, each function in each dimension should be run 30 times, and the initialization of the population should be uniform in the predefined search range [-100, 100]^D.

The suggested values from DISH algorithm were used for algorithm–specific parameters (with the exception of initial population size, which was doubled) and were as follows:

- Initial population size $NP_{init} = 50 * \log(D)\sqrt{D}$, which equals to 180, 364, 524 and 670 for dimensions 5, 10, 15 and 20 respectively.
- Final population size $NP_f = 4$.
- Historical memory size $H = 5$.
- Maximal external archive size $|\boldsymbol{A}| = NP$.
- Maximal historical best archive size $|\boldsymbol{A}_{best}| =$ unlimited.

The algorithm was programmed in Java language (Java 11.0.2) and run on a PC with 64-bit Windows 10, AMD A8-7600 Radeon R7 3.1 GHz CPU and 4 GB RAM.

IV. Results

The competition–mandatory basic statistical results of the DISH-XX algorithm in a form of best, worst, median, mean and standard deviation values can be found in Tables I, II, III and IV for dimensions 5, 10, 15 and 20 respectively. All tables present error values – difference from the global optima. It is also important to note that in Table I, there are no results for $f6$ and $f7$, since they were removed from the benchmark for 5*D*.

TABLE I
Results for 5D.

Func.	Best	Worst	Median	Mean	Std
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	1.25E-01	1.43E+02	6.95E+00	1.31E+01	2.82E+01
3	5.15E+00	8.15E+00	5.41E+00	5.65E+00	6.07E-01
4	0.00E+00	4.68E-02	1.23E-02	1.42E-02	9.57E-03
5	0.00E+00	6.24E-01	0.00E+00	2.08E-01	2.99E-01
6	–	–	–	–	–
7	–	–	–	–	–
8	0.00E+00	1.00E+02	0.00E+00	3.35E+00	1.83E+01
9	1.00E+02	3.00E+02	1.00E+02	1.10E+02	4.03E+01
10	3.00E+02	3.47E+02	3.47E+02	3.44E+02	1.20E+01

TABLE II
RESULTS FOR 10D.

Func.	Best	Worst	Median	Mean	Std
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	3.75E-01	1.45E+02	1.54E+01	1.97E+01	2.59E+01
3	1.07E+01	1.35E+01	1.13E+01	1.14E+01	5.85E-01
4	0.00E+00	7.40E-03	0.00E+00	2.47E-04	1.35E-03
5	0.00E+00	1.16E+01	4.16E-01	1.24E+00	2.80E+00
6	2.12E-04	1.12E+01	2.29E-01	6.96E-01	1.99E+00
7	3.51E-05	6.27E-01	3.12E-01	2.40E-01	2.50E-01
8	1.00E+02	1.00E+02	1.00E+02	1.00E+02	0.00E+00
9	1.00E+02	3.33E+02	3.30E+02	3.07E+02	7.01E+01
10	3.98E+02	4.46E+02	3.98E+02	4.07E+02	1.88E+01

TABLE III
RESULTS FOR 15D.

Func.	Best	Worst	Median	Mean	Std
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	2.50E-01	1.61E+02	2.91E+01	7.13E+01	6.52E+01
3	1.56E+01	1.78E+01	1.66E+01	1.66E+01	5.43E-01
4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
5	2.15E+00	1.40E+02	1.32E+01	2.18E+01	3.31E+01
6	4.47E-01	3.04E+01	7.33E+00	9.40E+00	8.66E+00
7	4.04E-02	1.14E+00	6.63E-01	6.33E-01	2.51E-01
8	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.57E-13
9	3.88E+02	3.92E+02	3.90E+02	3.90E+02	8.27E-01
10	4.00E+02	4.00E+02	4.00E+02	4.00E+02	0.00E+00

TABLE IV
RESULTS FOR 20D.

Func.	Best	Worst	Median	Mean	Std
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2	5.31E+00	3.89E+02	2.19E+01	8.67E+01	1.11E+02
3	2.45E+00	2.46E+01	2.17E+01	2.13E+01	3.74E+00
4	0.00E+00	7.40E-03	0.00E+00	2.47E-04	1.35E-03
5	1.10E+00	2.49E+02	1.92E+01	5.63E+01	6.63E+01
6	2.38E-01	1.20E+02	1.04E+00	1.50E+01	3.57E+01
7	7.62E-02	2.94E+01	2.26E+00	5.09E+00	6.42E+00
8	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.39E-13
9	3.99E+02	4.10E+02	4.05E+02	4.05E+02	2.50E+00
10	4.14E+02	4.14E+02	4.14E+02	4.14E+02	2.54E-02

Table V provides computational run–time in the specified format. It can be seen that the computational run–time of the algorithm increases with the dimension of the problem, but the run–time increase from 10D to 15D is only 0.4%, whereas the increase from 5D to 10D is approximately 11%. This suggests that the computational run–time will not increase linearly with the dimension. And thus, the algorithm might also be suitable for large–scale problems.

TABLE V
COMPUTATIONAL RUN–TIME

	$T0$	$T1$	$T2$	$T2 - T1/T0$
$D = 5$	29	178	8100.80	8094.66
$D = 10$	28	302	8967.80	8957.01
$D = 15$	29	406	9012.60	8998.6

Tables VI, VII, VIII and IX present the comparison between original DISH algorithm and DISH–XX variant. The comparison was done by Wilcoxon rank sum test with significance level α set to 0.05, in order to provide statistically proven difference in performance between the algorithm variants.

TABLE VI
DISH VS. DISH-XX ON CEC2020 5D.

	DISH		**DISH-XX**		
Func.	Median	Mean	Median	Mean	Diff.
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
2	1.88E-01	2.29E-01	6.95E+00	1.31E+01	-
3	5.15E+00	4.90E+00	5.41E+00	5.65E+00	-
4	9.86E-03	9.76E-03	1.23E-02	1.42E-02	-
5	0.00E+00	2.08E-01	0.00E+00	2.08E-01	=
6	–	–	–	–	*
7	–	–	–	–	*
8	0.00E+00	6.69E+00	0.00E+00	3.35E+00	=
9	1.00E+02	1.03E+02	1.00E+02	1.10E+02	=
10	3.47E+02	3.44E+02	3.47E+02	3.44E+02	=

TABLE VII
DISH VS. DISH-XX ON CEC2020 10D.

	DISH		**DISH-XX**		
Func.	Median	Mean	Median	Mean	Diff.
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
2	6.95E+00	1.94E+01	1.54E+01	1.97E+01	-
3	1.21E+01	1.19E+01	1.13E+01	1.14E+01	+
4	0.00E+00	3.29E-04	0.00E+00	2.47E-04	=
5	2.08E-01	4.31E+00	4.16E-01	1.24E+00	=
6	6.19E-02	1.80E-01	2.29E-01	6.96E-01	-
7	1.98E-04	1.48E-02	3.12E-01	2.40E-01	-
8	1.00E+02	9.72E+01	1.00E+02	1.00E+02	=
9	3.28E+02	2.67E+02	3.30E+02	3.07E+02	=
10	3.98E+02	4.04E+02	3.98E+02	4.07E+02	=

As can be seen, the DISH–XX variant's results are comparable to the DISH, and in some cases (specifically on problems in 20D) the DISH–XX has shown superior performance. The result of the Wilcoxon rank sum test is given in the last column of each table. If there is an = sign, both algorithms perform similarly, if there is a + sign, DISH–XX outperformed DISH and if there is a – sign, DISH outperformed DISH–XX. In 5D, DISH outperformed DISH–X on 3 functions ($f2$, $f3$ and $f4$), in 10D, the score is 3:1 for DISH outperforming DISH–XX on $f2$, $f6$ and $f7$, but losing on $f3$. In 15D the DISH algorithm outperformed DISH–XX in one test case - $f6$ and in 20D the score is 2:0 in favor of DISH–XX (better on $f3$ and $f5$). The overall combined score is therefore in favor of DISH with 7 wins, 3 loses and 28 ties ($f6$ and $f7$ in 5D were not evaluated).

What is interesting is the fact that the performance comparison is very variable with the dimension of the problem. This might be caused by different properties of optimized functions in different dimensions - especially for hybrid and composition cases. It seems, that DISH–XX algorithm might be more suitable for solving higher dimensional problems.

V. CONCLUSION

This paper proposes a participant for the CEC2020 single objective bound constrained numerical optimization competition, titled DISH–XX. It is an evolution of the previously successful DISH algorithm [27] with secondary crossover and two times larger initial population size. The secondary crossover serves to maintain the diversity of the population along with the intent to re–use historically beneficial parameter

TABLE VIII
DISH VS. DISH-XX ON CEC2020 15D.

	DISH		DISH-XX		
Func.	Median	Mean	Median	Mean	Diff.
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
2	1.23E+02	9.98E+01	2.91E+01	7.13E+01	=
3	1.68E+01	1.70E+01	1.66E+01	1.66E+01	=
4	0.00E+00	6.57E-04	0.00E+00	0.00E+00	=
5	1.06E+01	2.84E+01	1.32E+01	2.18E+01	=
6	1.44E+00	2.34E+00	7.33E+00	9.40E+00	-
7	7.05E-01	9.42E+00	6.63E-01	6.33E-01	=
8	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=
9	3.90E+02	3.90E+02	3.90E+02	3.90E+02	=
10	4.00E+02	4.00E+02	4.00E+02	4.00E+02	=

TABLE IX
DISH VS. DISH-XX ON CEC2020 20D.

	DISH		DISH-XX		
Func.	Median	Mean	Median	Mean	Diff.
1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=
2	7.29E+01	8.93E+01	2.19E+01	8.67E+01	=
3	2.23E+01	2.26E+01	2.17E+01	2.13E+01	+
4	0.00E+00	4.93E-04	0.00E+00	2.47E-04	=
5	5.87E+01	9.32E+01	1.92E+01	5.63E+01	+
6	1.57E+00	1.56E+00	1.04E+00	1.50E+01	=
7	1.61E+00	1.18E+01	2.26E+00	5.09E+00	=
8	1.00E+02	1.00E+02	1.00E+02	1.00E+02	=
9	4.05E+02	4.04E+02	4.05E+02	4.05E+02	=
10	4.14E+02	4.14E+02	4.14E+02	4.14E+02	=

information. In the result section of the paper, it is shown that the proposed DISH–XX variant performs competitively with its predecessor on the basis of CEC2020 benchmark. Thus, it might be a suitable successor for higher dimensional problems, since its performance in this area is better. Another benefit of the proposed variant is that it does not introduce any additional parameters, which values should be determined by the user. The future research direction in this area will be aimed at a thorough analysis of the benefits introduced by the secondary crossover along with its tuning.

REFERENCES

[1] K. Price and R. Storn, "Differential Evolution-a simple and efficient adaptive scheme for global optimization over continuous space," *Technical Report, International Computer Science Institute*, 1995.

[2] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution – an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[3] P. J. Angeline, "Adaptive and self-adaptive evolutionary computations," in *Computational intelligence: a dynamic systems perspective*. Citeseer, 1995.

[4] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Sánchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1088–1113, 2008.

[5] J. L. J. Laredo, C. Fernandes, J. J. Merelo, and C. Gagné, "Improving genetic algorithms performance via deterministic population shrinkage," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 819–826.

[6] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in *2005 IEEE congress on evolutionary computation*, vol. 2. IEEE, 2005, pp. 1769–1776.

[7] M. A. M. De Oca, T. Stutzle, K. Van den Enden, and M. Dorigo, "Incremental social learning in particle swarms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 2, pp. 368–384, 2010.

[8] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[9] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 2014, pp. 1658–1665.

[10] R. Poláková, J. Tvrdík, and P. Bujok, "Adaptation of population size according to current population diversity in differential evolution," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–8.

[11] A. Viktorin, M. Pluhacek, and R. Senkerik, "Network based linear population size reduction in shade," in *2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*. IEEE, 2016, pp. 86–93.

[12] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2013, pp. 71–78.

[13] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, vol. 635, 2013.

[14] ——, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, vol. 635, 2013.

[15] J. Liang, B. Qu, P. Suganthan, and Q. Chen, "Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization," *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, vol. 29, pp. 625–640, 2014.

[16] S.-M. Guo, J. S.-H. Tsai, C.-C. Yang, and P.-H. Hsu, "A self-optimization approach for l-shade incorporated with eigenvector-based crossover and successful-parent-selecting framework on cec 2015 benchmark set," in *2015 IEEE congress on evolutionary computation (CEC)*. IEEE, 2015, pp. 1003–1010.

[17] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, "An ensemble sinusoidal parameter adaptation incorporated with l-shade for solving cec2014 benchmark problems," in *2016 IEEE congress on evolutionary computation (CEC)*. IEEE, 2016, pp. 2958–2965.

[18] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: algorithm jso," in *2017 IEEE congress on evolutionary computation (CEC)*. IEEE, 2017, pp. 1311–1318.

[19] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 372–379.

[20] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems," in *2017 IEEE Congress on evolutionary computation (CEC)*. IEEE, 2017, pp. 145–152.

[21] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," in *Technical Report*. Nanyang Technological University Singapore, 2016.

[22] V. Stanovov, S. Akhmedova, and E. Semenkin, "Lshade algorithm with rank-based selective pressure strategy for solving cec 2017 benchmark problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[23] K. Price, N. Awad, M. Ali, and P. Suganthan, "Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization," in *Technical Report*. Nanyang Technological University, 2018.

[24] J. Brest, M. S. Maučec, and B. Bošković, "The 100-digit challenge: Algorithm jde100," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 19–26.

[25] A. Zamuda, "Function evaluations upto 1e+ 12 and large population sizes assessed in distance-based success history differential evolution for 100-digit challenge and numerical optimization scenarios (dishchain 1e+ 12) a competition entry for" 100-digit challenge, and four other

numerical optimization competitions" at the genetic and evolutionary computation conference (cecco) 2019," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 11–12.

[26] F. Lezama, J. Soares, R. Faia, and Z. Vale, "Hybrid-adaptive differential evolution with decay function (hyde-df) applied to the 100-digit challenge competition on single objective numerical optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 7–8.

[27] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, and A. Zamuda, "Distance based parameter adaptation for success-history based differential evolution," *Swarm and Evolutionary Computation*, vol. 50, p. 100462, 2019.

[28] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation*, vol. 13, no. 5, pp. 945–958, 2009.

[29] J. Brest, M. S. Maučec, and B. Bošković, "iL-SHADE: Improved L-SHADE Algorithm for Single Objective Real-Parameter Optimization," in *2016 IEEE World Congress on Computational Intelligence (IEEE WCCI 2016), Vancouver, Canada*, 2016, pp. 1188–1195.

[30] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, and A. Zamuda, "Distance based parameter adaptation for differential evolution," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–7.

Algorithm 1 DISH–XX, updates to DISH are highlighted by red color.

Set NP_{init}, NP_f, D, and $MAXFES$ (stopping criterion);
$NP = NP_{init}$, $H = 5$, $G = 1$, $\boldsymbol{x}_{best} = \{\}$, $k = 1$, $p_{max} = 0.25$, $p_{min} = 0.125$, $\boldsymbol{A} = \emptyset$, $\boldsymbol{A}_{best} = \emptyset$, $FES = 0$;
Randomly initialize population $\boldsymbol{P} = (\boldsymbol{x}_{1,G}, \ldots, \boldsymbol{x}_{NP,G})$ (1);
$FES+ = NP$;
Set all values in $\boldsymbol{M}_F$ to 0.5 and $\boldsymbol{M}_{CR}$ to 0.8;
$\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$;
$\boldsymbol{x}_{best} \rightarrow \boldsymbol{A}_{best}$;
while stopping criterion not met **do**
  $\boldsymbol{S}_F = \emptyset$, $\boldsymbol{S}_{CR} = \emptyset$;
  for $i = 1$ to NP **do**
    $r = U[1, H]$;
    if $r = H$ **then**
      $\boldsymbol{M}_{F,r} = 0.9$;
      $\boldsymbol{M}_{CR,r} = 0.9$;
    end if
    $CR_{i,G} = N(\boldsymbol{M}_{CR,r}, 0.1)$;
    if $CR_{i,G} < 0$ **then**
      $CR_{i,G} = 0$;
    else if $CR_{i,G} > 1$ **then**
      $CR_{i,G} = 1$;
    end if
    $F_{i,G} = C(\boldsymbol{M}_{F,r}, 0.1)$;
    while $F_{i,G} \leq 0$ **do**
      $F_{i,G} = C(\boldsymbol{M}_{F,r}, 0.1)$;
    end while
    if $F_{i,G} > 1$ **then**
      $F_{i,G} = 1$;
    end if
    $FES_{ratio} = FES/MAXFES$;
    if $FES_{ratio} < 0.6$ **and** $F_{i,G} > 0.7$ **then**
      $F_{i,G} = 0.7$;
    end if
    if $FES_{ratio} < 0.25$ **then**
      $CR_{i,G} = \max(CR_{i,G}, 0.7)$;
    else if $FES_{ratio} < 0.5$ **then**
      $CR_{i,G} = \max(CR_{i,G}, 0.6)$;
    end if
    $\boldsymbol{x}_{i,G} = \boldsymbol{P}[i]$;
    $p_i = p_{min} + FES_{ratio} * (p_{max} - p_{min})$ (5);
    if $FES_{ratio} < 0.2$ **then**
      $F_{w,i,G} = 0.7F_{i,G}$;
    else if $FES_{ratio} < 0.4$ **then**
      $F_{w,i,G} = 0.8F_{i,G}$;
    else
      $F_{w,i,G} = 1.2F_{i,G}$;
    end if
    $\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F_{w,i,G}(\boldsymbol{x}_{pBest} - \boldsymbol{x}_{i,G}) + F_{i,G}(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2})$ (4);
    $\boldsymbol{u}^*_{i,G}$ by binomial crossover (11);
    Randomly select $\boldsymbol{x}_{rAbest}$ from the $\boldsymbol{A}_{best}$;
    $\boldsymbol{u}_{i,G}$ by binomial crossover between $\boldsymbol{u}^*_{i,G}$ and $\boldsymbol{x}_{rAbest}$ (12);
    if $f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G})$ **then**
      $\boldsymbol{x}_{i,G+1} = \boldsymbol{u}_{i,G}$;
      $\boldsymbol{x}_{i,G} \rightarrow \boldsymbol{A}$;
      $F_i \rightarrow \boldsymbol{S}_F$, $CR_i \rightarrow \boldsymbol{S}_{CR}$;
      if $f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{best})$ **then**
        $\boldsymbol{u}_{i,G} \rightarrow \boldsymbol{A}_{best}$;
      end if
    else
      $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$;
    end if
    if $|\boldsymbol{A}| > NP$ **then**
      Randomly delete $|\boldsymbol{A}| - NP$ individuals from $|\boldsymbol{A}|$;
    end if
    $\boldsymbol{x}_{i,G+1} \rightarrow \boldsymbol{P}_{new}$;
  end for
  $NP_{new} = \text{round}(NP_{init} - FES_{ratio} * (NP_{init} - NP_f))$ (14);
  if $NP_{new} < NP$ **then**
    Sort individuals in $\boldsymbol{P}$ according to their objective function values and remove $NP - NP_{new}$ worst ones;
    $NP = NP_{new}$;
  end if
  if $|\boldsymbol{A}| > NP$ **then**
    Randomly delete $|\boldsymbol{A}| - NP$ individuals from $|\boldsymbol{A}|$;
  end if
  if $\boldsymbol{S}_F \neq \emptyset$ **and** $\boldsymbol{S}_{CR} \neq \emptyset$ **then**
    Update $\boldsymbol{M}_{F,k}$ (16) and $\boldsymbol{M}_{CR,k}$ (17) with Lehmer mean computed by (15) with distance based weights from (18);
    $k++$;
    if $k > H$ **then**
      $k = 1$;
    end if
  end if
  $\boldsymbol{P} = \boldsymbol{P}_{new}$, $\boldsymbol{P}_{new} = \{\}$, $\boldsymbol{x}_{best}$ = best from population $\boldsymbol{P}$, $G++$;
end while
return $\boldsymbol{x}_{best}$ as the best–found solution;