ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2023.1.045

Differential Evolution and Engineering Problems

Petr Bujok✉, Martin Lacko, Patrik Kolenovský

Department of Informatics and Computers, University of Ostrava, Czech Republic

petr.bujok@osu.cz✉, martin.lacko@osu.cz, patrik.kolenovsky@osu.cz

Abstract

In this paper, the performance of the Differential Evolution algorithm is evaluated when solving real-world problems. A Set of 13 engineering optimisation problems was selected from the fields of mechanics and industry to illustrate the usability of the Differential Evolution algorithm. Twelve variants of the standard Differential Evolution with various settings of the control parameters are compared with 19 state-of-the-art adaptive variants of this algorithm. The results are analysed statistically to achieve significant differences. Three variants of adaptive Differential Evolution provided better results compared to other algorithms. Some adaptive variants of Differential Evolution perform significantly worse than the original Differential Evolution with the fixed setting of the control parameters.

Keywords: *Global Optimisation, Differential Evolution, Parameter Adaptation, Real-world Problem, Engineering Problem, Experimental Comparison.*

Received: 29 May 2023
Accepted: 12 June 2023
Online: 16 June 2023
Published: 30 June 2023

1 Introduction

Engineering optimisation problems are often solved in various fields of research, industry or mechanics. The optimal solutions of these problems often have a crucial impact on economic or ecological demands. Clearly, using the most proper and powerful optimisation technique increases efficiency and decreases the costs of the solution.

Generally, the global optimisation problem is defined in the decision space Ω, which is bounded by limits, $\Omega = \prod_{j=1}^{D}[a_j, b_j]$, $a_j < b_j$. Then, the objective function f is defined in all $\boldsymbol{x} \in \Omega$ and the point $\boldsymbol{x}^*$ for $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \Omega$ is the solution of the global optimisation problem.

There are many ways to find the solution of the optimisation problems. Besides the traditional deterministic approaches that exist, stochastic techniques are inspired by processes from nature. A big group of stochastic methods is called Evolutionary algorithms (EAs) inspired by the evolution of species. One of the most popular EA is called Differential evolution (DE), and this algorithm will be used and studied in this paper.

The main idea of this paper is to illustrate the efficiency of the well-known DE variants when solving real-world engineering problems. A lot of engineering problems are defined by a small dimensionality, and therefore, using the newly designed adaptive optimisation methods seems unnecessary. For this study, several state-of-the-art DE algorithms are applied to 13 constrained optimisation engineering problems and compared statistically with the results of the standard DE using several settings of the control parameters. The results promise to show which settings of the DE control parameters or adaptation technique are better in real-world engineering problems, which are often solved in many fields of industry, research, and mechanics.

The rest of this paper is organised as follows. In Section 2, standard DE and several well-known state-of-the-art DE variants are selected and briefly described. Section 3 provides brief information on the selected engineering problems. In section 4, detail of the experimental comparison is provided. Sections 5 and 6 summarised a discussion of results and conclusions.

2 Differential Evolution

The DE algorithm was introduced in 1995 by Storn and Price [13]. DE is a population-based optimisation algorithm from a group of evolutionary algorithms which is very popular because of its simplicity and efficiency. The principle of this optimiser is depicted in Algorithm 1. The population P is initialised randomly in the search space Ω and evaluated by a cost function f. Then, the population is developed gradually using three evolutionary operators – mutation, crossover, and selection.

A mutation vector $\boldsymbol{u}$ is generated by a mutation operation. There are many variants of mutation variants, and each performs differently when solving various optimisation problems, where the most popular is rand/1 (1). This mutation variant uses three randomly selected individuals from the population, mutual from the current individual. Parameter F is called scale factor, typically from $F = (0,\ 2)$.

$$\boldsymbol{u} = \boldsymbol{x}_{r1} + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}) \tag{1}$$

Further, a new trial solution $\boldsymbol{y}$ is generated using crossover operation by the combination of elements from the original solution $\boldsymbol{x}_i$ and mutation vector $\boldsymbol{u}$.

Algorithm 1 Pseudo-code of the DE algorithm

```
initialise population P = {x_1, x_2, ..., x_N}
while stopping condition not reached do
    for i = 1, 2, ..., N do
        create mutant vector u by mutation
        create a new trial vector y by crossover
        evaluate f(y)
        if f(y) ≤ f(x_i) then
            insert y into next generation Q
        else
            insert x_i into next generation Q
    P ← Q
```

There are two widely-used crossover variants in DE, a binomial and exponential. The widely used crossover variant is binomial (2), where the crossover ratio $CR \in (0,\ 1)$ controls the number of elements from a mutated individual selected for a trial solution.

$$y_{i,j} = \begin{cases} u_{i,j}, & if\ rand_j(0,1) \leq CR\ or\ j = rand_j(1,D) \\ x_{i,j}, & otherwise. \end{cases} \tag{2}$$

If a new trial solution is outside of the search area Ω, it is mirrored back. In constrained problems, it is also necessary to check if the trial solution is located in the feasible area (called feasible solution) or not. Kononova et al. propose a comprehensive study comparing various strategies for dealing with infeasible solutions [10].

Even though the efficiency of the standard DE algorithm is sufficient on most optimisation problems. For high dimensionality D or complex problems, DE often gets stuck in local solution and do not provide acceptable results. The problem is in fixed values of the DE control parameters, F, CR, and N. Some recommendations for selecting the proper variant of mutation and crossover provide comprehensive study [7].

The solution to this problem provides an adaptation of parameters in DE. Many adaptive DE variants employ various mechanisms to change parameters during the search process. This study selects several well-known adaptive DE variants to solve real-world engineering problems.

2.1 Variants of DE used in Experiment

This paper applies several well-known state-of-the-art DE variants to real-world engineering problems to show their true applicability. Also, standard DE with several settings of its control parameters is employed to illustrate if adaptation of the parameters provides a significantly higher performance of the DE algorithm. The DE variants are listed chronologically.

The standard DE with fixed parameter settings uses only the most popular mutation variant rand/1, whereas two crossover versions are compared, binomial abbreviated by symbol 'b' and exponential denoted by symbol 'e' in the comparison. Setting of numerical parameters (N, F, and CR) is illustrated in section 4.

In 2006, Brest et al. proposed a variant of adaptive DE introducing a self-adaptive mechanism of the control parameters called jDE [2]. jDE uses the only combination of rand/1 mutation and binomial crossover. The control parameters of F and CR are sampled randomly for each member of population ($\boldsymbol{x}_i$), $F = (0.1,\ 0.9)$ and $CR = (0,\ 1)$. These values are resampled in each generation with probability $\tau_1 = \tau_2 = 0.1$. In this study, several settings of the jDE algorithm parameters are used.

In 2014, Wang et al. introduced a variant of CoBiDE which uses the covariance-matrix-based crossover and bimodal distribution of control parameters (F and CR) using the values of the parameters in a stochastic manner in order to adapt the parameter setting to the currently solved problem [17]. The exploitation of the covariance matrix should increase efficiency in optimisation problems where the coordinates of points in the population are highly correlated.

In 2015, Tang et al. proposed a variant of IDE using an Individual-dependent mechanism [16]. A new mutation scheme is dependent on the exploration and exploitation phases and also on the superiority and inferiority of the current solution. The values of F and CR are sampled for individuals independently according to the quality of the individuals.

In 2016, Bujok et al. introduced an enhanced variant of SHADE4 where four different DE strategies compete to generate a new trial point [8, 9]. These strategies are based on the combination of two kinds of mutation (randrl/1 and current-to-$p$best/1) and two types of crossover (binomial and exponential). The strategies are used to generate new trial points with a probability equal to their success in previous generations. The remaining setting of the SHADE4 variant is the same as in the original SHADE [14].

Two new variants of SHADE4 were derived for this study in the following manner. At first, a variant of SHADE2 uses only the current-to-$p$best mutation with both crossover variants. Secondly, SHADE4eig enhances the original SHADE4 by the Eigen transformation from CoBiDE.

In 2016, Polakova et al. proposed an improved version of successful L-SHADE4 which uses the competition of four DE strategies based on randrl/1 and current-to-$p$best mutations, and binomial and exponential crossover variants [12]. The L-SHADE4 is an enhanced variant of the original L-SHADE [15], which adapts the parameters of F and CR according to their success in previous generations and uses an archive of outperformed old individuals. The population size in L-SHADE is linearly decreased to prefer the exploration phase in the early stages and the exploitation phase in the last stages of the search.

In 2017, Brest et al. proposed an adaptive DE called jSO using a weighted selection of mutation strategy and modified adaptation of the control parameters [3]. The variant of jSO was derived from the successful L-SHADE variant, and it took second place in the

CEC 2017 competition of the evolutionary methods.

In 2019, Bujok proposed a jSO variant with the competition of eight strategies called cSO [5]. The original combination of current-to-*p*best mutation and binomial crossover is completed by randrl/1 mutation and exponential crossover. These four combinations are also used with the Eigen transformation used in CoBiDE. All strategies compete to be selected and generate new solutions with probabilities equal to success in previous generations.

In 2019, Brest et al. introduced a variant of adaptive jDE100 [4] derived from the original jDE [2]. The jDE100 uses two independent populations of individuals where rand/1 mutation and binomial crossover are used. After each generation of a bigger population, the best solution is stored in a smaller population, and several generations of a smaller population are performed. Parameters of F and CR are adapted similarly to the original jDE. In this study, several settings of population sizes are used.

In 2021, Bujok et al. proposed a variant of DEDMNA using three mutation variants rand/1, best/2, and rand-to-best/1, an archive of good historical solutions and linear population size adaptation [6]. The values of parameter F are computed for each member of the population independently based on the individual's performance. Similarly, the values of parameter CR are sampled for each individual independently randomly from $(0,\ 1)$. The values of both parameters are resampled with a small probability of 0.1.

3 Engineering Optimisation Problems

In this paper, 13 optimisation problems from various fields of engineering were selected as a benchmark. The list of problems is in Table 1, and a detailed description of the problems is provided in the study [1]. All the problems are constrained by the linear and non-linear limits of equality and non-equality. The dimensionality of the problems is $D \in (2,\ 11)$. For each problem, the true optimal solution is provided to estimate the efficiency of the optimisation algorithms. The value of the optimal solution is given by the sum of the achieved least function value f and violation v, $f + v$.

The Speed Reducer Design problem defines one of the essential parts of the gearbox. The main goal of this optimisation problem is to minimise the weight of the speed reducer with subject to 11 constraints.

The Tension/Compression Spring Design problem is focused on the minimisation of the weight of a tension (compression) spring, subject to constraints on minimum deflection, shear stress, surge frequency, limits on the outside diameter, and design variables.

The Pressure Vessel Design problem defines that a cylindrical vessel is capped at two sides by hemispherical heads. The goal is to minimise the total cost of material, forming, and welding.

The Tree-Bar Truss Design Problem defines a three-bar planar truss structure. The goal is to minimise the volume of a statically loaded three-bar truss with subject to constraints on each of the truss members.

The Gear Train Design problem is defined as an unconstrained discrete problem in mechanical engineering. The goal of the problem is to minimise the ratio of the angular velocity of the output shaft to the angular velocity of the input shaft (also called gear ratio).

The Cantilever Stepped Beam problem belongs to continuous, discrete, and mixed variable structural problems. The goal is to minimise the volume of the beam.

The Optimal Design of I-Shaped Beam problem defines how to minimise a vertical deflection of the beam, respecting the cross-sectional area and constraints under given loads.

The Tubular Column Design problem was defined to minimise the cost of a compressive load of a uniform column in the tubular section.

The Piston Lever problem provides the main objective of arranging the piston components by minimising the oil volume, whereas the piston lever is lifted up.

The Corrugated Bulkhead Design problem minimises the corrugated bulkhead weight in a chemical tanker, where the width, depth, length, and plate thickness are the design variables.

The Car Side Impact Design problem illustrates how the car is exposed to a side impact. The goal of the problem is to minimise the weight of the door defined by nine control parameters as thicknesses of B-pillar inner, B-pillar reinforcement, floor side inner, cross members, door beam, door beltline reinforcement, roof rail, materials of B-pillar inner, floor side inner, barrier height, and hitting position.

The Design of Welded Beam problem shows how the beam is under a vertical force. The goal is to find the minimum manufacturing cost of the welded beam.

The Reinforced Concrete Beam Design problem represents the realisation of the reinforced concrete beam. The goal is to minimise the cost of the structure, the reinforcement area, the beam's width, and the beam's depth.

Notice that the results of the algorithms on artificial problems and real-world problems are often different. Kudela proposed a study focused on the different performances of optimisation algorithms when solving artificially constructed problems with solutions in the centre of search space and real-world problems where the solution is not in the centre [11].

4 Experiments

A set of 13 engineering optimisation problems is selected for this study. The dimensionality and the optimal solutions of the problems are in Table 1. The problems are constrained, and more detailed information about the limits and conditions is in [1]. In the experiments, 30 independent runs for each test problem are performed to assess a reliable sample of results for analysis. In each run, the minimal function value of the test problem achieved by the algorithm is used in

Prob.	Problem name	D	Optimal solution
1	Speed Reducer Design	7	2994.4244658
2	Tension/Compression Spring Design	3	0.012665232788
3	Pressure Vessel Design	4	6059.714335048436
4	Tree-Bar Truss Design Problem	2	263.89584338
5	Design of Gear Train	4	2.70085714e-12
6	Cantilever Stepped Beam	5	1.3399576
7	Optimal Design of I-Shaped Beam	4	0.0130741
8	Tubular Column Design	2	26.486361473
9	Piston Lever	4	8.41269832311
10	Corrugated Bulkhead Design	4	6.8429580100808
11	Car Side Impact Design	11	22.84296954
12	Design of Welded Beam	4	1.724852308597366
13	Reinforced Concrete Beam Design	3	359.2080

Table 1: Engineering problems info

the comparison. The algorithm's run is restricted by the number of function evaluations $MaxFES = 9000$. The algorithms are stopped early if the difference between a founded solution and an optimal solution is acceptable, i.e. less than $= 1 \times 10^{-8}$.

The control parameters of all parameters are set according to the original recommendations. Fixed parameters studied in this paper are abbreviated in the names of algorithms as an index. The population size is set automatically in jSO, cSO, and LSHADE4 variants according to dimensionality. The remaining variants use fixed population size (or its initial value) - CoBiDE has a population size set to 50, DEDMNA has a dynamic size of initial population size, linearly decreased to 5, standard DE has a population size set to $(5, 10, 30, 50, 100, 200, 1000)$, IDE has population size 50, the population size of jDE is set to $(5, 50)$, jDE100 has two populations, and their sizes are set to four combinations $(200 - 20, 500 - 25, 1000 - 25, 100 - 5)$, SHADE4 has three different fixed population sizes $(10, 50, 100)$, SHADE4eig has population size 10. The effect of population size is studied mainly in the classic DE variant, where no adaptive mechanism is applied.

Further, several F and CR values in the classic DE are set to show its efficiency - $F = (0.1, 0.5, 0.8, 0.9)$ and $CR = (0.1, 0.5, 0.8, 0.9)$. Similarly, values of lower and upper limits for generating F values are set to three different combinations $(0.1-0.5), (0.5-0.9), (0.1-0.9)$. So, the variant of $DE_{b50F08CR08}$ represents the standard DE with rand/1 mutation and binomial crossover, population size $N = 50$, $F = 0.8$, and $CR = 0.8$.

All the algorithms are implemented in Matlab 2020b environment. All computations were performed on a standard PC with Windows 10, Intel(R) Core(TM)i7-9700 CPU 3.0 GHz, 16 GB RAM.

5 Results

Twelve variants of the standard DE with various settings of the control parameters are compared with 19 state-of-the-art DE algorithms when solving 13 real-world engineering optimisation problems. The experiment produces relatively huge data results to analyse; therefore, an advanced statistical comparison is performed besides standard descriptive values and plots.

The median values from 30 independent runs for each algorithm and engineering problem are presented in Tables 2-7. The best (minimal) achieved median values are for each problem highlighted in bold and underlined. More than one algorithm achieves the best solution for most of the problems. But the highlighting helps to see how efficient the methods are. Regarding the classic DE variants, the best performing is DE with binomial crossover, $N = 10$, $F = 0.8$, and $CR = 0.5$, which is the original setting recommended by the authors of DE. Very efficient is also the DE with binomial crossover and $N = 50$, $F = 0.8$, and $CR = 0.9$. The inefficient DE variants used huge population size $N = 200$, 1000 or small $CR = 0.1$. Regarding all DE variants, the best results achieved $SHADE4_{50}$ provides the best results in 12 out of 13 engineering problems. The variants of LSHADE4 and cSO are very efficient and are the best in 11 and 10 problems. The variant of jDE with $N = 5$ and F generated from $(0.1, 0.5)$ is not able to solve no engineering problem.

An insight into algorithms' results' variability provides the Box-plots in Figures 1-6. In most cases, the algorithms provided the same results in the one engineering problem, illustrated by no blue box and no outlined red crosses. Therefore six problems with more variable results are selected. The biggest variability of results is observed for jDE variants, especially jDE with $N = 50$ and F sampled from $(0.1, 0.5)$.

To better insight into algorithms' performance, the Friedman test is applied. The medians of achieved minimum values at the end of the search ($FES = 9{,}000$ or less) are used. The mean ranks are presented in Table 8, where a lower rank means better performing DE, and the algorithms are ordered from the best to the worst. The null hypothesis on the equivalent efficiency of the methods is rejected with $p < 5 \times 10^{-6}$.

The best overall results provides $SHADE4_{50}$ variant followed by LSHADE4 and jDE with $N = 50$, and F sampled from $(0.1, 0.9)$. Surprisingly, adaptive variants

Table 2: Median values of the compared algorithms - 1.

Prob.	$DE_{b50F08CR05}$	$DE_{b50F08CR08}$	$DE_{b10F08CR05}$	$DE_{b100F08CR05}$	$DE_{b50F01CR05}$
1	2994.43	2994.43	**2994.42**	2994.775	2999.41
2	0.0127851	0.01266605	0.0127182	0.01309825	0.0135867
3	6062.21	**6059.71**	**6059.71**	6096.48	6410.09
4	**263.896**	**263.896**	**263.896**	**263.896**	**263.896**
5	1.166E-10	2.30782E-11	2.30631E-10	1.35559E-10	9.74565E-10
6	1.34097	1.34022	1.33998	1.345355	1.3434
7	0.0130809	**0.0130741**	**0.0130741**	0.01314945	0.0131296
8	**26.4864**	**26.4864**	**26.4864**	26.4874	**26.4864**
9	11.98405	8.41505	167.473	61.76575	12.7772
10	6.84298	**6.84296**	**6.84296**	6.84779	6.85033
11	22.8826	22.86645	**22.843**	23.1212	23.0776
12	1.733675	1.725075	**1.72485**	1.771525	1.86113
13	**359.208**	**359.208**	**359.208**	**359.208**	**359.208**

Table 3: Median values of the compared algorithms - 2.

Prob.	$DE_{b50F09CR05}$	$DE_{b50F08CR01}$	$DE_{b50F08CR09}$	$DE_{b200F08CR05}$	$DE_{b1000F08CR05}$
1	2994.43	2994.43	2994.43	3000.305	3069.71
2	0.0128465	0.01325465	0.0126653	0.0133211	0.0143878
3	6070.34	6223.55	**6059.71**	6176.89	6874.33
4	**263.896**	263.8995	**263.896**	263.897	263.919
5	1.545E-10	1.36165E-09	**2.70086E-12**	1.08005E-10	1.16612E-10
6	1.341525	1.348465	1.34001	1.37191	1.908215
7	0.0130897	0.0138714	**0.0130741**	0.0133145	0.01415825
8	**26.4864**	26.54615	**26.4864**	26.5033	26.6294
9	8.55837	8.704045	87.94291	65.10925	241.1185
10	6.84308	6.879495	**6.84296**	6.91793	7.34611
11	22.93365	23.0946	22.86015	23.5581	24.8954
12	1.73716	1.832565	1.7249	1.82632	2.069355
13	**359.208**	359.212	**359.208**	359.212	359.251

Table 4: Median values of the compared algorithms - 3.

Prob.	$DE_{b5F08CR05}$	$DE_{e30F08CR05}$	$DEDMNA_5$	$SHADE4_{100}$	$SHADE4_{50}$
1	2994.46	**2994.42**	2998.375	2994.43	**2994.42**
2	0.013521	0.0126675	0.01338255	0.0126798	**0.0126652**
3	6410.09	6059.735	6167.205	6060.055	**6059.71**
4	**263.896**	**263.896**	263.902	**263.896**	**263.896**
5	6.353E-09	1.08005E-10	0.2097295	9.93988E-11	2.30782E-11
6	1.34093	1.34065	1.365205	1.340525	**1.33996**
7	0.0131789	0.0130751	0.0132624	0.0130743	**0.0130741**
8	**26.4864**	**26.4864**	26.5023	**26.4864**	**26.4864**
9	175.1045	**8.4127**	176.7815	8.413195	**8.4127**
10	6.84297	**6.84296**	6.85941	6.843065	**6.84296**
11	23.00725	22.88155	23.3265	22.85095	**22.843**
12	1.90241	1.72539	1.77365	1.72667	**1.72485**
13	360.743	**359.208**	359.4555	**359.208**	**359.208**

of cSO, jSO, etc., outperform several standard DE variants with $N = 50$ and high (more progressive) F, CR settings. The variant of jDE with $N = 5$ and F generated from $(0.1, 0.5)$ is the worst-performing method in the comparison. So, lower values of F provide worse results when solving engineering problems.

More detailed results can provide the Kruskal-Wallis test applied to the results of each problem independently. The null hypothesis was rejected for each engineering problem (the significance level was less than

Table 5: Median values of the compared algorithms - 4.

Prob.	$SHADE4_{10}$	LSHADE4	cSO	$jDE100_{200\_20}$	$jDE100_{500\_25}$
1	**2994.42**	**2994.42**	**2994.42**	**2994.42**	**2994.42**
2	0.0127055	**0.0126652**	**0.0126652**	0.0126801	0.01268835
3	6523.22	**6059.71**	**6059.71**	6059.72	6059.77
4	**263.896**	**263.896**	**263.896**	**263.896**	**263.896**
5	2.358E-09	0.06180385	0.220024	2.30782E-11	2.30782E-11
6	1.340095	**1.33996**	**1.33996**	1.340025	1.34033
7	**0.0130741**	**0.0130741**	**0.0130741**	0.01307485	0.0130769
8	**26.4864**	**26.4864**	**26.4864**	**26.4864**	**26.4864**
9	111.77205	**8.4127**	167.473	8.438285	22.6592
10	**6.84296**	**6.84296**	**6.84296**	**6.84296**	**6.84296**
11	22.8443	22.8432	22.8432	22.84835	22.85855
12	**1.72485**	**1.72485**	**1.72485**	1.725185	1.72731
13	**359.208**	**359.208**	**359.208**	**359.208**	**359.208**

1×10^{-70}). The number of problems where the methods achieve the best result, the second-best result, the third-best result, and the worst result are provided in Table 9. The algorithms are ordered from left top to right bottom based on the numbers. The best results achieved cSO, followed by jSO, $SHADE4_{50}$, and LSHADE4. The standard DE with $N = 10$, $F = 0.8$, and CR=*0.5* and $N = 50$, $F = 0.8$, and CR=*0.5* outperformed most of adaptive DE variants in the comparison. The variant of jDE with $N = 5$ and F generated from $(0.1, 0.5)$ is the worst-performing algorithm.

Because the $SHADE4_{\mathbf{50}}$ provided the best results in the experiments, it was selected as the reference method for applying the Wilcoxon rank-sum tests. This test provides a comparison of all the algorithms in the experiment with the reference (best) method to show more detail. Due to the clarity, the number of the problems where the reference method ($SHADE4_{\mathbf{50}}$) performs significantly better, similarly, and significantly worse than the second method are provided in Table 10. The methods are ordered from the best (non-reference) algorithm to the worst method. The second best method is LSHADE4, which performs worse than the reference $SHADE4_{\mathbf{50}}$ only in two problems, and it is never better, followed by the cSO with three worse results and zero better cases. The jDE with $N = 5$ and $F \in (0.1, 0.5)$ performs significantly worse than the reference method in all 13 engineering problems.

Each algorithm stopped when it achieved 9,000 of the function evaluations or when the difference between the founded solution and the optimal solution is less than 1×10^{-8}. Most of the algorithms in the comparison achieved maximal provided *FES*, and only six were able to provide the solution earlier. Table 11 illustrates the fastest algorithms, where CoBiDE and DEDMNA variants provided bad results for the problems.

Each algorithm in the comparison produced a new trial solution, which replaced the original (old) solution if it was better. In these cases, the success of the algorithm is increased by one. More information about the performance of the algorithms provides in Table 12, presenting the median values of algorithms' success regarding all 13 engineering problems. Notice that these values are computed in view of the achieved number of function evaluations because some algorithms provided the results early before final *FES*=9,000. Algorithms are ordered from the more successful in the reproduction process to the less successful. The most successful algorithm is CoBiDE, with more than 87 % of success. Although this algorithm is very fast, its performance is not high as for other adaptive DE variants.

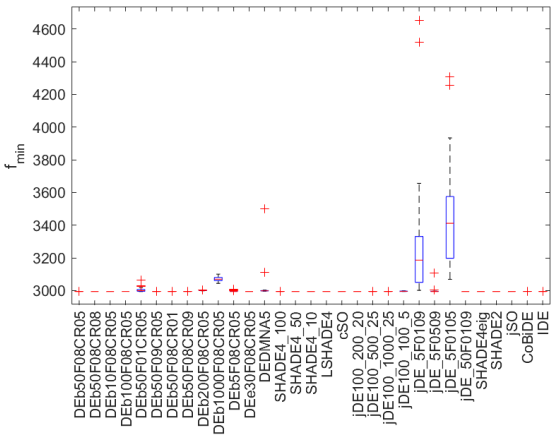

Figure 1: Results of the algorithms on the problem 1.

6 Conclusion

In this paper, 31 variants of the DE algorithm (12 standard DE and 19 adaptive DE) are compared when solving 13 real-world engineering problems. The final results of the algorithms were statistically compared with several interesting conclusions.

Only several adaptive DE variants perform significantly better than the standard DE with a proper setting regarding all 13 problems, $SHADE4_{50}$, LSHADE4, and $jDE_{50F0109}$.

Table 6: Median values of the compared algorithms - 5.

Prob.	$jDE100_{1000\_25}$	$jDE100_{100\_5}$	jDE_{5F0109}	jDE_{5F0509}	jDE_{5F0105}
1	2994.43	2995.855	3186.36	2994.5	3411.645
2	0.0127131	0.01277805	0.0159849	0.0135403	0.06142775
3	6060.14	6086.715	7787.165	6820.41	11304.85
4	**263.896**	**263.896**	264.0585	**263.896**	265.5115
5	1.356E-10	2.30782E-11	2.5669E-08	2.35764E-09	5.06474E-07
6	1.34036	1.35059	2.8573	1.34326	4.92805
7	0.0130791	0.01314905	0.0144837	0.01309835	0.04178755
8	**26.4864**	**26.4864**	26.4866	**26.4864**	29.18905
9	10.769645	15.6576	477.7895	201.779	901.4045
10	6.84297	6.851765	7.574705	6.845435	10.23008
11	22.8693	23.0622	25.7113	23.2106	27.1776
12	1.73058	1.749345	3.063145	1.849895	3.84543
13	**359.208**	**359.208**	362.4805	360.733	369.966

Table 7: Median values of the compared algorithms - 6.

Prob.	$jDE_{50F0109}$	SHADE4eig	SHADE2	jSO	CoBiDE	IDE
1	**2994.42**	**2994.42**	**2994.42**	**2994.42**	2994.73	2994.45
2	0.0126771	0.0126856	0.01271165	0.0126655	0.0126834	0.01831
3	**6059.71**	6410.09	6343.295	**6059.71**	6060.135	6111.1
4	**263.896**	**263.896**	**263.896**	**263.896**	**263.896**	**263.896**
5	2.308E-11	1.12777E-09	1.26338E-09	0.222494	2.30782E-11	0.31135
6	**1.33996**	1.340125	1.34031	**1.33996**	1.342395	1.34607
7	0.0130742	**0.0130741**	**0.0130741**	**0.0130741**	0.0130772	0.01309
8	**26.4864**	**26.4864**	**26.4864**	**26.4864**	**26.4864**	**26.4864**
9	8.419635	**8.4127**	167.473	167.473	10.1846	167.567
10	**6.84296**	**6.84296**	**6.84296**	**6.84296**	6.843655	**6.84296**
11	22.8433	23.1845	23.1845	22.8433	22.95775	22.9222
12	1.724945	**1.72485**	**1.72485**	**1.72485**	1.728155	1.72656
13	**359.208**	**359.208**	**359.208**	**359.208**	**359.208**	**359.208**

Table 8: Mean ranks from the Friedman test of median values.

$SHADE4_{50}$	LSHADE4	$jDE_{50F0109}$	$DE_{b50F08CR08}$	$DE_{b50F08CR09}$
6.2	8.0	8.2	9.4	9.4
$jDE100_{200\_20}$	cSO	$DE_{b10F08CR05}$	jSO	$DE_{e30F08CR05}$
9.6	9.7	10.0	10.2	10.7
$jDE100_{500\_25}$	SHADE4eig	$SHADE4_{100}$	$SHADE4_{10}$	SHADE2
11.6	11.9	12.2	12.4	13.8
$jDE100_{1000\_25}$	CoBiDE	$DE_{b50F08CR05}$	$DE_{b50F09CR05}$	$jDE100_{100\_5}$
14.5	14.8	15.5	16.2	17.9
IDE	$DE_{b100F08CR05}$	$D\ddot{E}_{b50F01CR05}$	$DE_{b5F08CR05}$	jDE_{5F0509}
18.7	20.0	20.3	21.6	22.8
$DE_{b50F08CR01}$	$DE_{b200F08CR05}$	$DEDMNA_5$	$DE_{b1000F08CR05}$	jDE_{5F0109}
23.0	24.2	25.8	27.5	29.2
jDE_{5F0105}				
30.6				

The optimal settings of the standard DE for the engineering problems that are represented by constrained functions with a small dimensionality are $N = 50$, $F = 0.8$, and $CR = 0.8,\ 0.9$. The success of the higher values of F and CR means that more distant mutation individuals are able to find the optimal solution. Big population size ($N = 200,\ 1000$) and small population size ($N = 5$) performed worse. The performance of the

Table 9: Number of 1st, 2nd, 3rd, and the last positions of all algorithms from the Kruskal-Wallis tests.

cSO 10/10/8/1	jSO 8/8/9/0	$SHADE4_{50}$ 7/8/10/0	LSHADE4 7/6/6/0	$DE_{b10F08CR05}$ 6/5/5/0
$DE_{b50F08CR09}$ 6/5/5/0	$DE_{e30F08CR05}$ 5/6/5/0	$SHADE4_{10}$ 5/5/5/0	$jDE_{50F0109}$ 5/5/5/0	SHADE4eig 5/5/5/0
SHADE2 5/5/5/0	$jDE100_{200\_20}$ 4/4/4/0	CoBiDE 3/4/3/0	$DE_{b50F08CR05}$ 3/3/3/0	$DE_{b50F08CR08}$ 3/3/3/0
$DE_{b50F09CR05}$ 3/3/3/0	$SHADE4_{100}$ 3/3/3/0	$jDE100_{500\_25}$ 3/3/3/0	$jDE100_{1000\_25}$ 3/3/3/0	IDE 3/3/3/0
$jDE100_{100\_5}$ 2/2/2/0	$DE_{b100F08CR05}$ 1/1/1/0	$DE_{b50F01CR05}$ 0/0/0/0	$DE_{b50F08CR01}$ 0/0/0/0	$DE_{b200F08CR05}$ 0/0/0/0
$DE_{b5F08CR05}$ 0/0/0/0	$DEDMNA_5$ 0/0/0/0	jDE_{5F0109} 0/0/0/0	jDE_{5F0509} 0/0/0/0	$DE_{b1000F08CR05}$ 0/0/0/1
jDE_{5F0105} 0/0/0/11				

Table 10: Number of worse, similar, and better results of all algorithms compared to $SHADE4_{50}$ and the significant differences from the Wilcoxon rank-sum tests.

LSHADE4 2/11/0	cSO 3/10/0	$DE_{b10F08CR05}$ 4/9/0	jSO 4/9/0	$jDE_{50F0109}$ 5/8/0
SHADE4eig 5/8/0	$DE_{b50F08CR09}$ 6/7/0	$DE_{b50F08CR08}$ 6/7/0	$SHADE4_{10}$ 6/7/0	SHADE2 6/7/0
$DE_{e30F08CR05}$ 7/6/0	$jDE100_{200\_20}$ 7/6/0	$jDE100_{500\_25}$ 7/6/0	$jDE100_{100\_5}$ 9/4/0	CoBiDE 9/4/0
IDE 9/4/0	$DE_{b50F08CR05}$ 10/3/0	$DE_{b50F01CR05}$ 10/3/0	$DEb_{50F09CR05}$ 10/3/0	$SHADE4_{100}$ 10/3/0
$jDE100_{1000\_25}$ 10/3/0	$DE_{b100F08CR05}$ 11/2/0	$DE_{b5F08CR05}$ 11/2/0	jDE_{5F0509} 11/2/0	$DE_{b50F08CR01}$ 13/0/0
$DE_{b200F08CR05}$ 13/0/0	$DE_{b1000F08CR05}$ 13/0/0	$DEDMNA_5$ 13/0/0	jDE_{5F0109} 13/0/0	jDE_{5F0105} 13/0/0

Table 11: Median of algorithms' number of function evaluations necessary to solve all 13 problems.

Algorithm	FES
CoBiDE	1047
$DEDMNA_5$	5756
$DE_{b10F08CR05}$	7860
LSHADE4	8206
SHADE4eig	4245
jSO8eig	8130

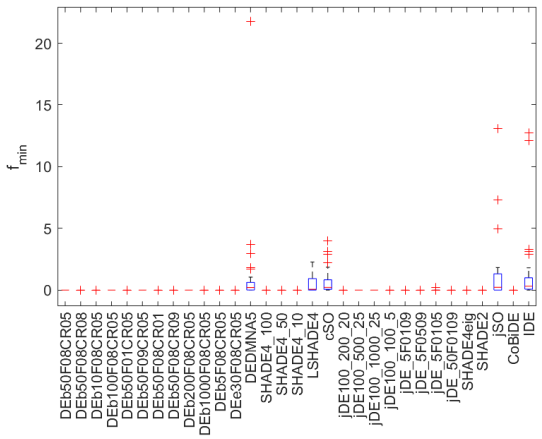

Figure 2: Results of the algorithms on the problem 5.

standard DE decreases with decreasing value of *CR*.

Although the adaptive DE variants use an enhanced approach to adapt the setting of the control parameters to achieve success in the currently solved problem, it is surprising that only some adaptive DE outperform the standard DE. The DE variants using fixed population size (CoBiDE, IDE, jDE, jDE100, SHADE4, and SHADE4eig) are underprivileged compared to versions with an adaptation of N ($DEDMNA_5$, jSO, jSO8eig, and LSHADE4). Therefore it is interesting that variants of $SHADE4_{50}$ or $jDE_{50F0109}$ are on the top positions in the comparison. The common denomina-

Table 12: Median of algorithms' success (%) regarding all 13 problems.

Algorithm	Succ.	Algorithm	Succ.	Algorithm	Succ.
CoBiDE	87.2	$SHADE4_{50}$	25.4	$DE_{b100F08CR05}$	14.5
$DE_{b1000F08CR05}$	37.3	LSHADE4	23.9	$DE_{b50F08CR08}$	17.3
jSO8eig	32.5	$jDE100_{100\_5}$	23.6	$DE_{e30F08CR05}$	13.4
$DEDMNA_5$	30.1	jSO	24.4	$DE_{b50F08CR05}$	12.1
SHADE4eig	38.9	IDE	21.4	$DE_{b50F09CR05}$	11.5
$jDE100_{1000\_25}$	27.9	$jDE100_{200\_20}$	23.2	$DE_{b50F08CR01}$	8.6
$DE_{bin50F01CR05}$	22.6	$DE_{b200F08CR05}$	21.1	$DE_{b10F08CR05}$	11.1
$DE_{bin50F01CR05}$	22.6	$DE_{b200F08CR05}$	21.1	$DE_{b10F08CR05}$	11.1
$DE_{b50F01CR05}$	22.6	$DE_{b200F08CR05}$	21.1	$DE_{10F08CR05}$	11.1
$SHADE4_{10}$	35.5	$jDE_{50F0109}$	17.1	jDE_{5F0109}	7.7
$SHADE4_{100}$	26.8	jDE_{5F0509}	16.7	$DE_{b5F08CR05}$	5.1
$jDE100_{500\_25}$	26.1	$DE_{b50F08CR09}$	19.8	jDE_{5F0105}	4.1
SHADE2	33.3				

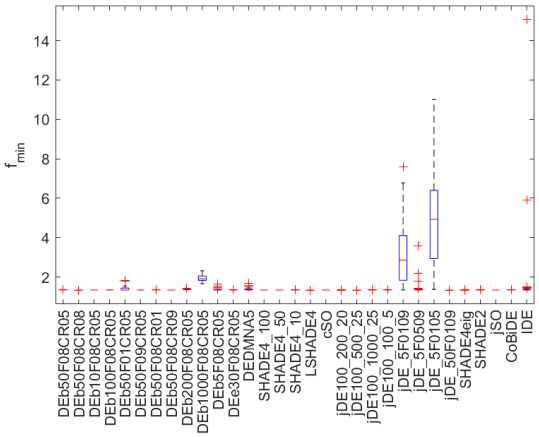

Figure 3: Results of the algorithms on the problem 6.

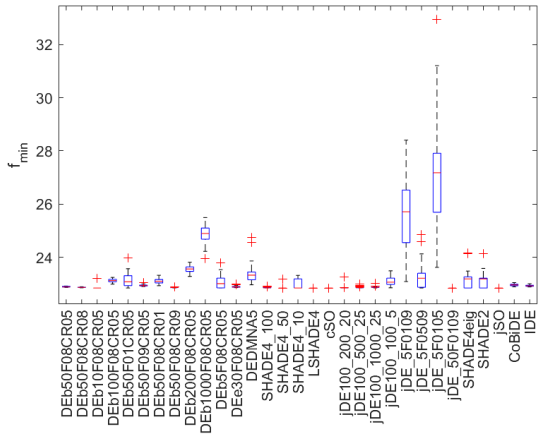

Figure 5: Results of the algorithms on the problem 11.

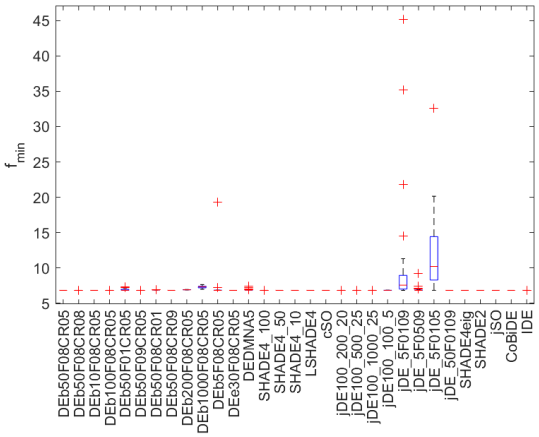

Figure 4: Results of the algorithms on the problem 10.

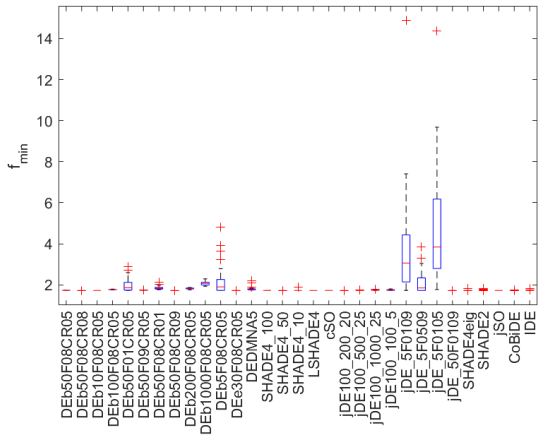

Figure 6: Results of the algorithms on the problem 12.

tor of these algorithms is the 'middle' population size, $N = 50$. Variants with higher or smaller population sizes perform worse. Adapting the DE control parameters provided better results than the standard DE; nevertheless, the variants with fixed parameters need tuning to be usable in engineering practice. Achieved conclusions enable future research on a new algorithm with good accuracy and lower time complexity.

References

[1] Bayzidi, H., Talatahari, S., Saraee, M., and Lamarche, C.-P. Social network search for solving engineering optimization problems. *Computational Intelligence and Neuroscience 2021* (2021).

[2] Brest, J., Greiner, S., Bošković, B., Mernik, M., and Žumer, V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation 10* (2006), 646–657.

[3] Brest, J., Maučec, M. S., and Bošković, B. Single objective real-parameter optimization: Algorithm jso. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (2017), pp. 1311–1318.

[4] Brest, J., Maučec, M. S., and Bošković, B. The 100-digit challenge: Algorithm jDE100. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), pp. 19–26.

[5] Bujok, P. Competition of strategies in jso algorithm. In *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing* (Cham, 2020), A. Zamuda, S. Das, P. N. Suganthan, and B. K. Panigrahi, Eds., Springer, pp. 113–121.

[6] Bujok, P. The real-life application of differential evolution with a distance-based mutation-selection. *Mathematics 9*, 16 (2021).

[7] Bujok, P., and Tvrdík, J. A comparison of various strategies in differential evolution. In *MENDEL, 17th International Conference on Soft Computing, Brno, Czech Republic* (2011), R. Matoušek, Ed., pp. 48–55.

[8] Bujok, P., and Tvrdík, J. New adaptive variant of differential evolution and real-world optimization problems. In *Proceedings of BIOMA 2016, he 7th International Conference on Bioinspired Optimization Methods and their Applications*. 2016. accepted.

[9] Bujok, P., Tvrdík, J., and Poláková, R. Evaluating the performance of shade with competing strategies on CEC 2014 single-parameter test suite. In *IEEE Congress on Evolutionary Computation (CEC) 2016* (2016), pp. 5002–5009.

[10] Kononova, A. V., Vermetten, D., Caraffini, F., Mitran, M.-A., and Zaharie, D. The importance of being constrained: Dealing with infeasible solutions in differential evolution and beyond. *Evolutionary Computation* (may 2023), 1–46.

[11] Kudela, J. A critical problem in benchmarking and analysis of evolutionary computation methods. *Nature Machine Intelligence 4*, 12 (2022), 1238–1245.

[12] Polakova, R., Tvrdik, J., and Bujok, P. Evaluating the performance of l-shade with competing strategies on cec2014 single parameter-operator test suite. In *2016 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC)* (2016), IEEE Congress on Evolutionary Computation, IEEE; IEEE Computat Intelligence Soc; Int Neural Network Soc; Evolutionary Programming Soc; IET; IEEE BigData Initiat; Gulf Univ Sci & Technol, pp. 1181–1187. IEEE Congress on Evolutionary Computation (CEC) held as part of IEEE World Congress on Computational Intelligence (IEEE WCCI), Vancouver, CANADA, JUL 24-29, 2016.

[13] Storn, R., and Price, K. V. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11* (1997), 341–359.

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

[15] Tanabe, R., and Fukunaga, A. S. Improving the search performance of shade using linear population size reduction. In *IEEE Congress on Evolutionary Computation (CEC) 2014* (2014), pp. 1658–1665.

[16] Tang, L., Dong, Y., and Liu, J. Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation 19*, 4 (2015), 560–574.

[17] Wang, Y., Li, H.-X., Huang, T., and Li, L. Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. *Applied Soft Computing 18* (2014), 232–247.