An adaptive variant of jSO with multiple crossover strategies employing Eigen transformation

Patrik Kolenovsky
Department of Informatics and Computers
Faculty of Science, University of Ostrava
patrik.kolenovsky@osu.cz

Petr Bujok
Department of Informatics and Computers
Faculty of Science, University of Ostrava
petr.bujok@osu.cz
ORCID: 0000-0003-2956-1226

***Abstract*—In this paper, new strategy options are developed for the adaptive jSO algorithm. The proposed variant of jSO is based on the competition of a binomial and exponential crossover. Moreover, an Eigen transformation approach is employed in the selected crossover with a given probability. The proposed variant of jSO is applied to the CEC 2022 benchmark set, which contains 12 functions with dimensionality $D = 10,\ 20$. The proposed algorithm found the optima values in seven problems out of 24. When comparing the new variant of jSO with the original jSO algorithm, nine functions were improved, where two of them significantly.**

***Index Terms*—jSO, multiple crossover strategies, Eigen transformation, experiments, test problems**

I. Introduction

The global optimisation problem offers many opportunities for improving robustness, efficiency, and accuracy of proposed solutions. For the objective function f, $f(x), x = (x_1,\ x_2,\ \ldots, x_D) \in \mathbb{R}^D$ and the domain of feasible solutions Ω constrained by bounds, a lower limit (a_j) and an upper limit (b_j), $\Omega = \prod_{j=1}^{D}[a_j,\ b_j]$, $a_j < b_j$, $j = 1,\ 2,\ \ldots, D$, the global minimum point $\vec{x}^*$ satisfying condition $f(x^*) \leq f(x),\ \forall x \in \Omega$ is the solution of the problem.

The problem of global optimisation occurs in many industries and researchers. Therefore, it is necessary to develop a sufficiently robust approach that finds suitable solutions according to the specified conditions. There are several ways to solve the global optimisation problem. Among one the most well-known approaches are algorithms that model animal behaviour or more general evolutionary algorithms (EAs) that model the natural approach of Charles Darwin's theory. The main feature of these approaches lies in the possibilities of variations of different strategies that try to achieve the best results regardless of the complexity of the functions.

These variations enable to create and modify new algorithms with many combinations and completely new approaches to finding solutions. This has allowed for extensive development and the search for new avenues for further development. This creates space for new solutions for solving the global optimisation problem on the new CEC 2022 benchmark set.

There are many successful and popular EAs which achieved good results on competitions and which are frequently applied on real-world problems. One of the most efficient is the adaptive variant of Differential Evolution called jSO. The aim of the paper is to show how the competition of crossover variants and Eigen transformation used before crossover operation increase the efficiency of the successful jSO variant.

II. jSO algorithm

Many experimental studies show that the jSO algorithm appears to be a successful optimiser [1] and therefore, it makes sense to extend this method with research and development. The jSO algorithm is based on an extension of the iL-SHADE algorithm [2] with a change in the mutation strategy, where it shows improvement in many parts of its results. Both the algorithms are based on SHADE [3] and L-SHADE [4] variants that are derived from the original Differential Evolution (DE) algorithm, which was first introduced in 1995 and is very often developed and applied for its efficiency and simplicity [5], [6]. In a previous experiment, a variant of jDE100 [7] with the development of parameter adaptation has been shown to be effective [8].

One way of developing DE was the SHADE algorithm. First introduced in 2013 [9]. Its principle is based on the adaptation of *CR* and F parameters. With the addition of the Linear Population Size Reduction (LPSR) property which causes a gradual decrease of the population depending on the linear function a new algorithm known as L-SHADE was created [3]. The iL-SHADE algorithm is therefore further development of L-SHADE [2]. They differ in several principles:

- Higher values for *CR* parameter generated by inner optimisation process.
- Storing historical memory values M_{CR} and M_F.
- Very high values of F and low values of *CR* are forbidden in an early stage.
- After each generation g, p are computed as follows:

$$p = (\frac{P_{max} - P_{min}}{max\_nfes}) \ . \ nfes + p_{min}, \tag{1}$$

'where *nfes* is the current number of objective function evaluations and *max_nfes* is maximum possible number of evaluations.' [1]

The jSO algorithm differs from iL-SHADE in a mutation strategy for generating a trial vector. Where L-SHADE and iL-SHADE apply DE/current-to-pBest/1 mutation strategy for generating of a trial vector:

$$\vec{v}_{i,g} = \vec{x}_{i,g} + F(\vec{x}_{pBest,g} - \vec{x}_{i,g}) + F(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}), \tag{2}$$

The principle of trial vector generation for jSO is in the new DE/current-to-pBest-w/1 approach, which is formed as follows:

$$\vec{v}_{i,g} = \vec{x}_{i,g} + F_w(\vec{x}_{pBest,g} - \vec{x}_{i,g}) + F(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}), \quad (3)$$

where F_w is defined by:

$$F_w = \begin{cases} 0.7 * F, & \textit{nfes} < 0.2\ \textit{max\_nfes} \\ 0.8 * F, & \textit{nfes} < 0.4\ \textit{max\_nfes} \\ 1.2 * F, & \text{otherwise}, \end{cases} \quad (4)$$

'The aim of the presented weighted mutation strategy is to apply a smaller factor F_w to multiply difference of vectors in which $\vec{x}_{pBest,g}$ appears at early stages of the evolutionary process, while in later stages higher factor F_w is used. With factor F_w the vector $\vec{x}_{pBest,g}$, might have a lower and/or higher influence' [1].

In 2020, Sun et al. proposed adaptive DE with a combined strategy, CSDE [10]. Two mutation variants - DE/current-to-$p$best/1 and DE/$p$best-to-rand/1 compete to be selected in reproduction, based on success in previous generations. Results achieved from the CEC2014 benchmark set show that CSDE performs better compared to seven adaptive DE variants.

In 2020, Shuijia et al. proposed an adaptive variant of JADE (EJADE) to optimise photovoltaic models [11]. In EJADE, there are two enhancements - selection of sorted CR values based on function values and linear population size reduction. The proposed EJADE is used for the optimisation of five photovoltaic models, where results are compared with eight various adaptive DE variants. The proposed variant of EJADE provides very good results compared to other optimisers in the study. It illustrates the efficiency of very simple mechanisms in practice.

In 2021, Shen et al. proposed a modified variant of jSO applied to constrained problems [12]. The proposed MjSO introduces three advanced approaches – symmetry-search-process parameter control, cosine-based parameter adaptation, and weighted opposition-based restart mechanism. The results achieved on the CEC 2017 benchmark set illustrate the better performance of MjSO compared to five various adaptive DE variants.

In 2021, Goji et al. proposed an adaptive regeneration framework for the DE algorithm [13]. The values of *CR* are computed for each individual based on the function evaluations of the current, the best, and the worst points of the population. Moreover, the regeneration of stagnated individuals is controlled and a renewal strategy to generate new solutions is employed. The framework was implemented in eight DE variants, where it achieved $80 - 100$ % of optimisation rate without significantly increased complexity.

III. The new variant of jSO: jSObeE

The original jSO algorithm is a very successful optimiser, but it uses only one variant of crossover which restricts the possibility to cope with various optimisation problems. Previous experimental studies showed that the combination of several mutations or crossover variants enables to achieve the better results in various optimisation problems [14], [15]. Therefore, the competition of two well-known crossover variants is employed in the original jSO to extend it and increase its performance. Moreover, an Eigen transformation approach from a variant of CoBiDE is used to increase performance in problems represented by rotated objective functions. So, the newly presented variant of jSO is called *jSObeE*, where 'be' represents the combination of binomial and exponential crossover, and 'E' denoted the Eigen transformation (see pseudo-code in Algorithm 1).

Algorithm 1 A new variant of jSO: jSObeE

```
initialise population P = {x_1, x_2, ..., x_N}
evaluate all individuals by a goal function
set both probabilities equally q_{1,2} = 0.5
while stopping condition not reached do
  if rand(0,1) < peig then
    Eigen coordinate system is used
    Eigenvectors B are updated
    for i = 1, 2, ..., N do
      select a proper crossover by roulette (5)
      create a new trial point y_i
      evaluate f(y_i)
      if f(y_i) ≤ f(x_i) then
        insert y_i into P
      else
        insert x_i into P
      end if
    end for
  else
    standard coordinate system is used
    for i = 1, 2, ..., N do
      select a proper crossover by roulette (5)
      create a new trial point y_i
      evaluate f(y_i)
      if f(y_i) ≤ f(x_i) then
        insert y_i into P
      else
        insert x_i into P
      end if
    end for
  end if
  update population size N
  update jSO parameters
end while
```

A. Competition of crossover variants

At the beginning of jSObeE, a population of N individuals is initialised and evaluated by a cost function f. The probabilities to use of both crossover variants are set equally to $q_h = 0.5$ (generally, it is $1/H$). It means that each crossover has the same probability to be applied when the mutation vector $\vec{v}_{i,g}$ is combined with the parent vector $\vec{x}_{i,g}$. The crossover variant producing new individuals ($\vec{y}_{i,g}$) more successfully (better

than parent individuals, ie. $f(\vec{y}_{i,g}) \leq f(\vec{x}_{i,g})$), increases the probability to be used in future generations

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}, \tag{5}$$

where n_h is the count of the success of the hth crossover variant, and $n_0 > 0$ is a constant preventing a significant change of q_h when random success of any crossover occurs. Moreover, the values of q_h are re-initialised to a equal values (0.5) if any probability q_h is lower than the given limit $\delta > 0$, where $\delta = 1/(5 \times H)$. This provided a protection against huge performance of some crossover variant. This approach enables to use more successful crossover strategy (producing high quality solutions) in various problems and stages of the optimisation process. More detail about this approach is provided in [16].

B. Eigen transformation

The competition of crossover variants promises a good performance on various optimisation problems because two different approaches are combined. Moreover, the Eigen transformation approach introduced in CoBiDE [17] is employed to increase the efficiency of the proposed method in problems defined by rotated objective functions. Similarly, Guo et al. proposed the Eigen transformation for crossover in DE [18].

In the beginning of each generation g, the original CoBiDE computed Eigenvalues (matrix $\vec{D}$) and Eigenvectors (matrix $\vec{B}$) from the covariance matrix ($\vec{C}$) computed from a *ps* part of a better individuals of population.

$$\vec{C} = \vec{B}\vec{D}^2\vec{B}^T. \tag{6}$$

After the Eigen transformation, a new solution $\vec{y}_{i,g}$ is produced in an Eigen coordinate system:

$$\vec{x}'_{i,g} = \vec{B}^{-1}\vec{x}_{i,g} = \vec{B}^T\vec{x}_{i,g}, \tag{7}$$

$$\vec{u}'_{i,g} = \vec{B}^{-1}\vec{u}_{i,g} = \vec{B}^T\vec{u}_{i,g}. \tag{8}$$

Then, a binomial crossover is applied to produce a new solution $\vec{y}'_{i,g}$ which is finally transformed back into a standard coordinate system:

$$\vec{y}_{i,g} = \vec{B}\vec{y}'_{i,g}. \tag{9}$$

This approach is used for a whole generation if the randomly generated number is lower than the control parameter *peig*, which controls the frequency of the Eigen transformation.

These two enhancements are gradually added to the original jSO, and two new variants of jSObe (using only competition of two crossover variants) and jSObeE (employing also the Eigen transformation) are applied on the CEC 2022 set and compared with the original jSO.

IV. Experiments

The experiment was performed on the CEC 2022 benchmark set which contains 12 test functions. Those functions were proposed with a special session and competition on Single Objective Bound Constrained Real-Parameter Numerical Optimisation as a part of Evolutionary Computation (CEC) 2022. There are unimodal problems, multimodal problems, hybrid problems, and composed problems. This session was established to test the newly developed EAs algorithms. The test functions can be examined in two basic settings for the number of possible dimensions, namely for $D = 10$ and $D = 20$. The original jSO algorithms is applied in the original setting [1]. The proposed variants of jSObe and jSObeE are using following settings. The number of competing crossover variants is $H = 2$, parameter of $\delta = 0.1$, and parameter of $n_0 = 2$. Moreover, the Eigen transformation is controlled by two parameters, $ps = 0.5$ and $peig = 0.4$. All algorithms are implemented in Matlab 2020b where the experiments were subsequently conducted. The experiments were performed on a PC (Windows 10) CPU Intel(R) Core(TM)i7-9700 3.0 GHz, 16 GB RAM.

V. Results

The proposed variants of jSObe and jSObeE are applied on the CEC 2022 benchmark set. The basic characteristics of better performing jSObeE (based on error values) are in Table I and II. We can see that jSObeE finds the solution in nine problems out of 24. Better results are achieved for lower dimension $D = 10$, where jSOEbeE solve six problems.

TABLE I
Basic characteristics of jSObeE for $D = 10$

F	min	max	med	mean	std
1	1.68E-09	9.79E-09	8.40E-09	7.68E-09	1.77E-09
2	6.94E-09	8.9161	3.98658	5.16823	2.40979
3	6.52E-09	9.93E-09	9.04E-09	8.72E-09	1.04E-09
4	1.98992	4.9748	2.98488	3.21704	8.13E-01
5	3.68E-09	9.79E-09	8.63E-09	8.04E-09	1.47E-09
6	1.60E-03	3.58E-01	1.45E-02	4.36E-02	7.30E-02
7	4.34E-09	6.38E-06	1.12E-08	3.50E-07	1.19E-06
8	3.54E-02	3.51E-01	1.12E-01	1.31E-01	7.94E-02
9	229.284	229.284	229.284	229.284	8.67E-14
10	100.146	100.258	100.186	100.1931	2.39E-02
11	7.05E-09	9.99E-09	9.17E-09	9.04E-09	7.83E-10
12	162.183	164.931	162.7	163.362	9.18E-01

The estimated time-complexity, achieved by predefined computations [19] is provided in Table III

TABLE II
Basic characteristics of jSObeE for $D = 20$

F	min	max	med	mean	std
1	5.69E-09	9.89E-09	9.13E-09	8.82E-09	9.51E-10
2	44.8955	49.0845	44.8955	45.0351	7.65E-01
3	7.37E-09	9.96E-09	9.36E-09	9.12E-09	6.96E-10
4	3.97984	11.9395	8.95463	8.921461	1.72298
5	7.37E-09	9.96E-09	9.39E-09	9.07E-09	7.60E-10
6	2.82E-02	4.56E-02	6.36E-02	9.88E-02	1.02E-01
7	1.30713	20.5019	3.297115	5.39987	5.46911
8	2.71E-01	20.6289	19.70435	15.3202	7.67055
9	180.781	180.781	180.781	180.781	2.89E-14
10	100.223	100.361	100.303	100.300	3.59E-02
11	300	400	300	306.667	25.37081
12	228.859	232.26	231.17	230.819	1.15283

The success of the newly applied competition of two crossover variants is presented in Table IV, where column

'bin' represents the original binomial crossover and 'exp' the exponential one. We can see that newly employed exponential crossover is more successful in most test problems (18 out of 24). It seems that the exponential crossover copes with these problems well.

TABLE III
TIME-COMPLEXITY OF THE NEW JSOBEE.

D	T0	T1	T2	(T2-T1)/T0
10	0.0619	0.28	1.28	16.16
20	0.0610	0.32	3.05	44.75

Further, the success of the second newly used approach called the Eigen transformation is assessed. In the column 'std.' the success of the standard type of crossover is presented and 'Eig.' is for the newly used Eigen transformation. We can see that the original approach is more successful in 17 out of 24 problems (especially for $D = 20$). Necessary to note that these values only estimate the real performance and usability of the newly incorporated enhancements.

TABLE IV
EFFICIENCY OF EIGEN TRANSFORMATION AND TWO CROSSOVER VARIANTS IN JSOBEE.

D	F	std.	Eig.	bin	exp
10	1	6764	**6834**	**7182**	6305
10	2	8914	**8955**	**9148**	7925
10	3	11103	**6120**	6672	**10415**
10	4	**4822**	3536	2589	**5533**
10	5	**7381**	5155	5400	**7111**
10	6	4374	**8496**	5098	**8077**
10	7	**7941**	3601	1221	**10274**
10	8	**4370**	2558	1797	**5218**
10	9	13608	**8781**	10700	**11776**
10	10	**2464**	1964	1743	**2698**
10	11	**14607**	9713	10822	**13452**
10	12	**10646**	7544	8248	**9827**
20	1	19449	**23795**	17984	**25282**
20	2	**38021**	27404	**37173**	28035
20	3	**46674**	19915	9937	**56806**
20	4	**20179**	14152	6491	**26697**
20	5	**24529**	17544	**21610**	20432
20	6	12300	**37926**	12660	**36828**
20	7	**23184**	11283	5066	**29149**
20	8	**14228**	6747	3555	**18319**
20	9	**35445**	28632	**33179**	30642
20	10	**7133**	5906	3614	**9438**
20	11	**61431**	38826	**52426**	46356
20	12	**30068**	24594	9510	**43615**

Two newly proposed variants of jSO are compared with the original jSO based on the error values and also based on the function evaluations, for each dimension independently. At first, the mean ranks from the Friedman tests are provided in Table V. The mean rank values represent the efficiency of each method regarding all test problems, where a lower mean rank means a better performing method.

It is obvious that the variant of jSObeE proposes better error values for $D = 10$, whereas for $D = 20$ is better jSObe. Regarding the speed of the optimisation, the original jSO performs faster for both dimensions. It is caused by no more enhancements, which make the newly proposed methods slightly slower. Column 'avg' represents the average mean rank of each algorithm regarding accuracy and speed.

TABLE V
MEAN RANKS FROM THE FRIEDMAN TESTS.

	Accuracy			Speed	
Alg.	D=10	D=20	avg	D=10	D=20
jSO	2.04	2.08	1.93	**1.75**	**1.83**
jSObeE	**1.96**	2	2.01	2.08	2
jSObe	2	**1.92**	2.06	2.17	2.17

TABLE VI
MEDIAN VALUES AND ACHIEVED SIGNIFICANCE FROM THE KRUSKAL-WALLIS TESTS.

D	F	sig.	jSO	jSObe	jSObeE
10	1	≈	8.32E-09	8.64E-09	8.40E-09
10	2	≈	3.98658	3.98658	3.98658
10	3	≈	9.04E-09	8.79E-09	9.04E-09
10	4	*	**2.98488**	2.98488	2.98488
10	5	≈	8.79E-09	8.58E-09	8.63E-09
10	6	***	2.78E-01	2.90E-01	**1.45E-02**
10	7	≈	9.85E-09	8.60E-09	1.12E-08
10	8	***	1.82E-01	2.33E-01	**1.12E-01**
10	9	≈	229.284	229.284	229.284
10	10	≈	100.188	100.2	100.186
10	11	≈	8.99E-09	8.90E-09	9.17E-09
10	12	≈	162.7	162.7	162.7
20	1	≈	9.15E-09	9.36E-09	9.13E-09
20	2	≈	44.896	44.896	44.896
20	3	≈	9.49E-09	9.32E-09	9.36E-09
20	4	***	**6.96471**	9.14471	8.95463
20	5	≈	8.89E-09	8.74E-09	9.39E-09
20	6	***	4.96E-01	4.99E-01	**6.36E-02**
20	7	≈	2.69638	2.61427	3.29712
20	8	≈	20.3003	19.6057	19.7044
20	9	≈	180.781	180.781	180.781
20	10	***	**100.224**	100.289	100.303
20	11	≈	300	300	300
20	12	***	232.26	231.737	**231.17**

More detail of the comparison provide results of the Kruskal.Wallis test from Table VI. Column 'sig.' represents achieved significance, and it is clear that only in seven problems algorithms performs differently. The original jSO is better in three problems, whereas, the variant of jSObeE is better in four problems.

TABLE VII
COUNTS OF THE BEST, SECOND BEST AND LAST POSITIONS OF THE COMPARED ALGORITHMS.

position	jSObeE	jSObe	jSO
1st	10	6	6
2nd	5	9	8
3rd	7	7	8

Table VII provides number of the best achieved error values for each algorithm and both dimensions. Obviously, jSObeE is better in ten problems, and jSObe and jSO in six problems.

Finally, better performing newly proposed jSObeE is compared with jSO and jSObe variants using the Wilcoxon rank-

TABLE VIII
MEDIAN VALUES WITH ACHIEVED SIGNIFICANCE FROM THE WILCOXON RANK-SUM TESTS.

D	F	jSObeE	sig.	jSO	sig.	jSObe
10	1	8.40E-09	≈	8.32E-09	≈	8.64E-09
10	2	3.98658	≈	3.98658	≈	3.98658
10	3	9.04E-09	≈	9.04E-09	≈	8.79E-09
10	4	2.98488	**	**2.98488**	≈	2.98488
10	5	8.63E-09	≈	8.79E-09	≈	8.58E-09
10	6	**1.45E-02**	***	2.77E-01	***	2.90E-01
10	7	1.12E-08	≈	9.85E-09	≈	8.60E-09
10	8	**1.12E-01**	**	1.82E-01	***	2.33E-01
10	9	229.284	≈	229.284	≈	229.284
10	10	100.186	≈	100.188	≈	100.2
10	11	9.17E-09	≈	8.99E-09	≈	8.90E-09
10	12	162.7	≈	162.7	≈	162.7
20	1	9.13E-09	≈	9.15E-09	≈	9.36E-09
20	2	44.8955	≈	44.8955	≈	44.8955
20	3	9.36E-09	≈	9.49E-09	≈	9.32E-09
20	4	8.95463	***	**6.96471**	≈	9.14471
20	5	9.39E-09	≈	8.89E-09	*	8.74E-09
20	6	**6.36E-02**	***	4.96E-01	***	4.99E-01
20	7	3.297115	≈	2.696375	≈	2.614265
20	8	19.7044	≈	20.30025	≈	19.6057
20	9	180.781	≈	180.781	≈	180.781
20	10	100.303	***	**100.224**	≈	100.289
20	11	300	≈	300	≈	300
20	12	**231.17**	***	232.26	*	231.737

sum test, for each problem and dimension independently VIII. We can see, the variant of jSObeE is significantly better than the original jSO in four problems and worse in three problems. jSObeE is significantly better than jSObe in four problems and it is never worse performing.

Better insight into algorithms' performance provide linear convergence plots in Figure 1 and 2. Mostly, all three methods converges similarly, there are only small differences in ability to minimise the function error values.

Finally, the values of q_h for binomial (dash) and exponential (solid) crossover variants are depicted in Figure 3 for each problem and both dimensions. Mostly, binomial crossover is prioritised in initial stages, whereas exponential crossover is selected more frequently in late stages.

VI. CONCLUSION

Newly proposed variants of jSO based on competition of two crossover variants and the Eigen transformation are applied on the CEC 2022 benchmark set and compared with the original jSO. The results show that the variant of jSObeE using both competitions of crossover and the Eigen transformation performs well. The accuracy of both newly proposed methods is better than the original jSO, whereas convergence speed is better for jSO. Both newly employed approaches were used in the recommended settings, therefore, more experimental studies in future research can provide better results.

ACKNOWLEDGMENT

The research was supported by the University of Ostrava from the project SGS17/PřF-MF/2022.

REFERENCES

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

[2] M. M. S. Brest J. and B. B., "il-shade: Improved l-shade algorithm for single objective real-parameter optimization." Vancouver, BC, Canada: IEEE, 2016.

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

[4] ——, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *IEEE Congress on Evolutionary Computation 2013*. IEEE Computational Intelligence Society, 2013, pp. 1952–1959.

[5] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 27–54, 2011.

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

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

[8] K. P. Bujok P. and J. V., "Eigenvector crossover in jde100 algorithm." Glasgow, UK: IEEE, 2020.

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

[10] G. Sun, B. Yang, Z. Yang, and G. Xu, "An adaptive differential evolution with combined strategy for global numerical optimization," *SOFT COMPUTING*, vol. 24, no. 9, SI, pp. 6277–6296, MAY 2020.

[11] S. Li, Q. Gu, W. Gong, and B. Ning, "An enhanced adaptive differential evolution algorithm for parameter extraction of photovoltaic models," *ENERGY CONVERSION AND MANAGEMENT*, vol. 205, FEB 1 2020.

[12] Y. Shen, Z. Liang, H. Kang, X. Sun, and Q. Chen, "A modified jso algorithm for solving constrained engineering problems," *Symmetry*, vol. 13, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2073-8994/13/1/63

[13] G. Sun, C. Li, and L. Deng, "An adaptive regeneration framework based on search space adjustment for differential evolution," *NEURAL COMPUTING & APPLICATIONS*, vol. 33, no. 15, pp. 9503–9519, AUG 2021.

[14] P. Bujok and J. Tvrdík, "Adaptive differential evolution: Shade with competing crossover strategies," in *ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING, PT I*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds., vol. 9119. Polish Neural Network Soc; Univ Social Sci; Czestochowa Univ Technol, Inst Computat Intelligence; IEEE Computat Intelligence Soc, Poland Chapter, 2015, pp. 329–339, 14th International Conference on Artificial Intelligence and Soft Computing ICAISC, Zakopane, POLAND, JUN 14-18, 2015.

[15] P. Bujok, J. Tvrdík, and R. Poláková, "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.

[16] J. Tvrdík, "Competitive differential evolution," in *MENDEL 2006, 12th International Conference on Soft Computing*, R. Matoušek and P. Ošmera, Eds. Brno: University of Technology, 2006, pp. 7–12.

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

[18] 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 *IEEE Congress on Evolutionary Computation (CEC)*, 2015, pp. 1003–1010.

[19] A. Kumar, K. V. Price, A. W. Mohamed, A. A. Hadi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2022 special session and competition on single objective bound constrained numerical optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China And Technical Report, Nanyang Technological University, Singapore, Tech. Rep., 2021, github.com/P-N-Suganthan.

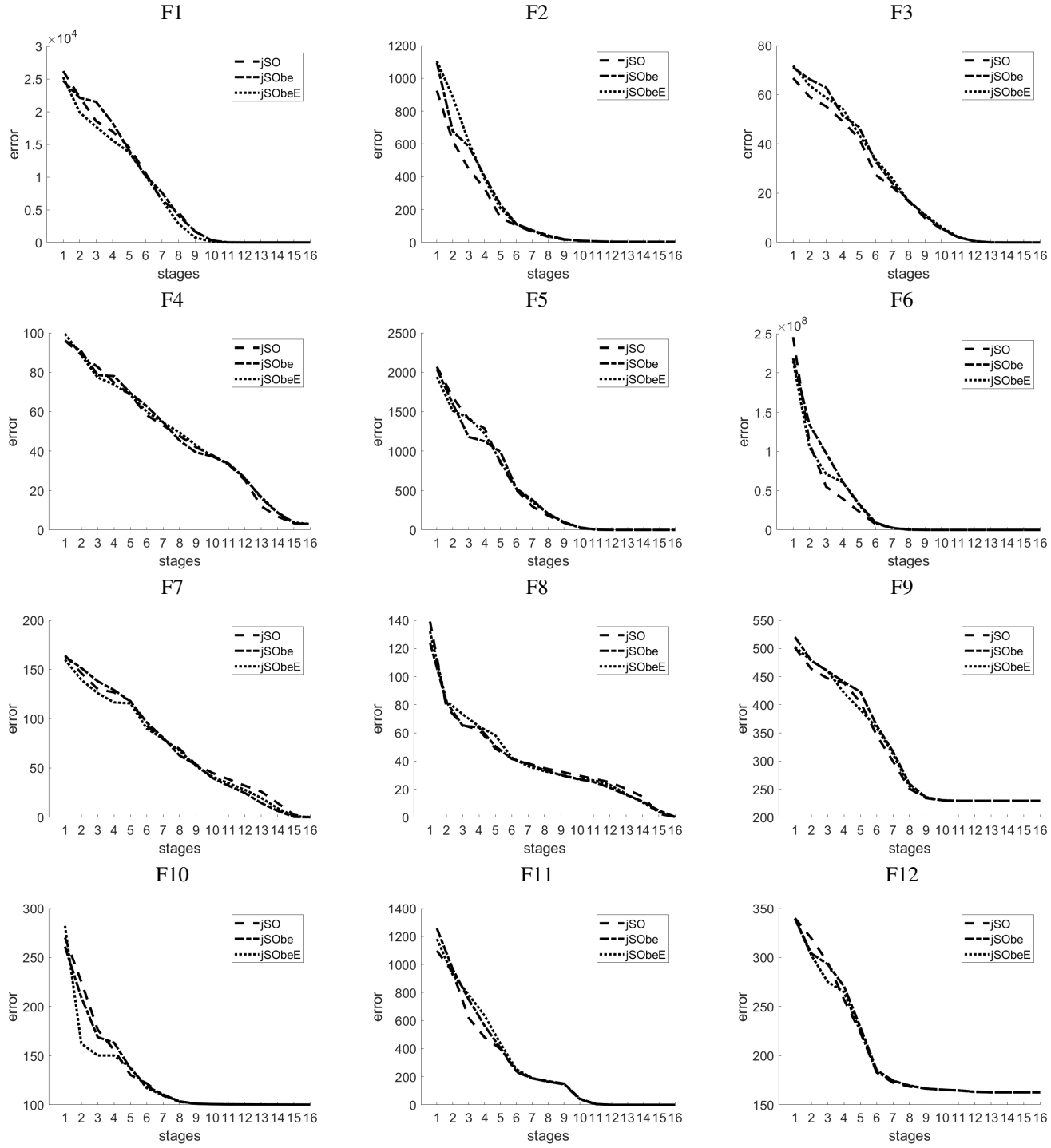

Fig. 1. Convergence error-lines of compared algorithms, $D = 10$.

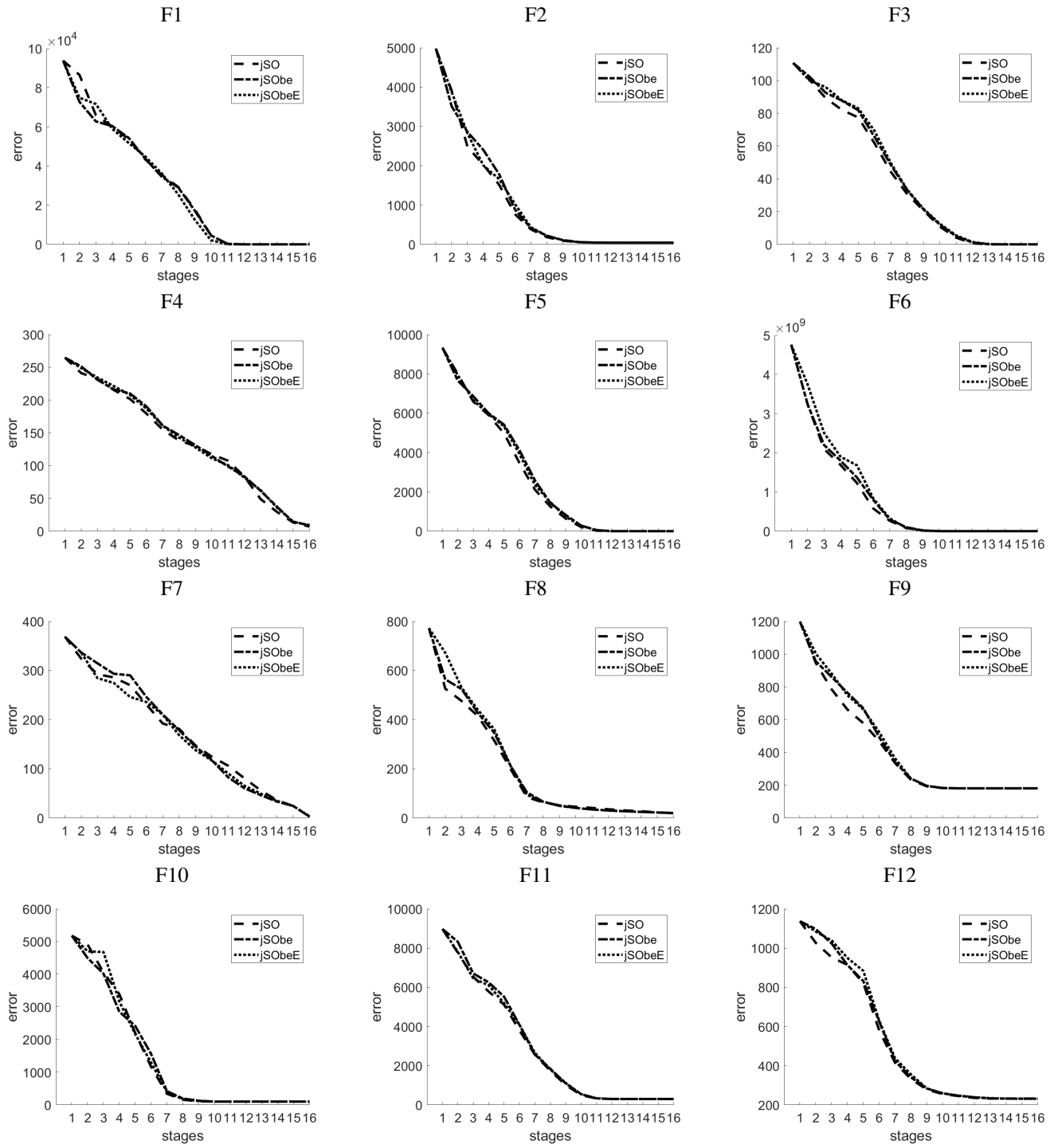

Fig. 2. Convergence error-lines of compared algorithms, $D = 20$.

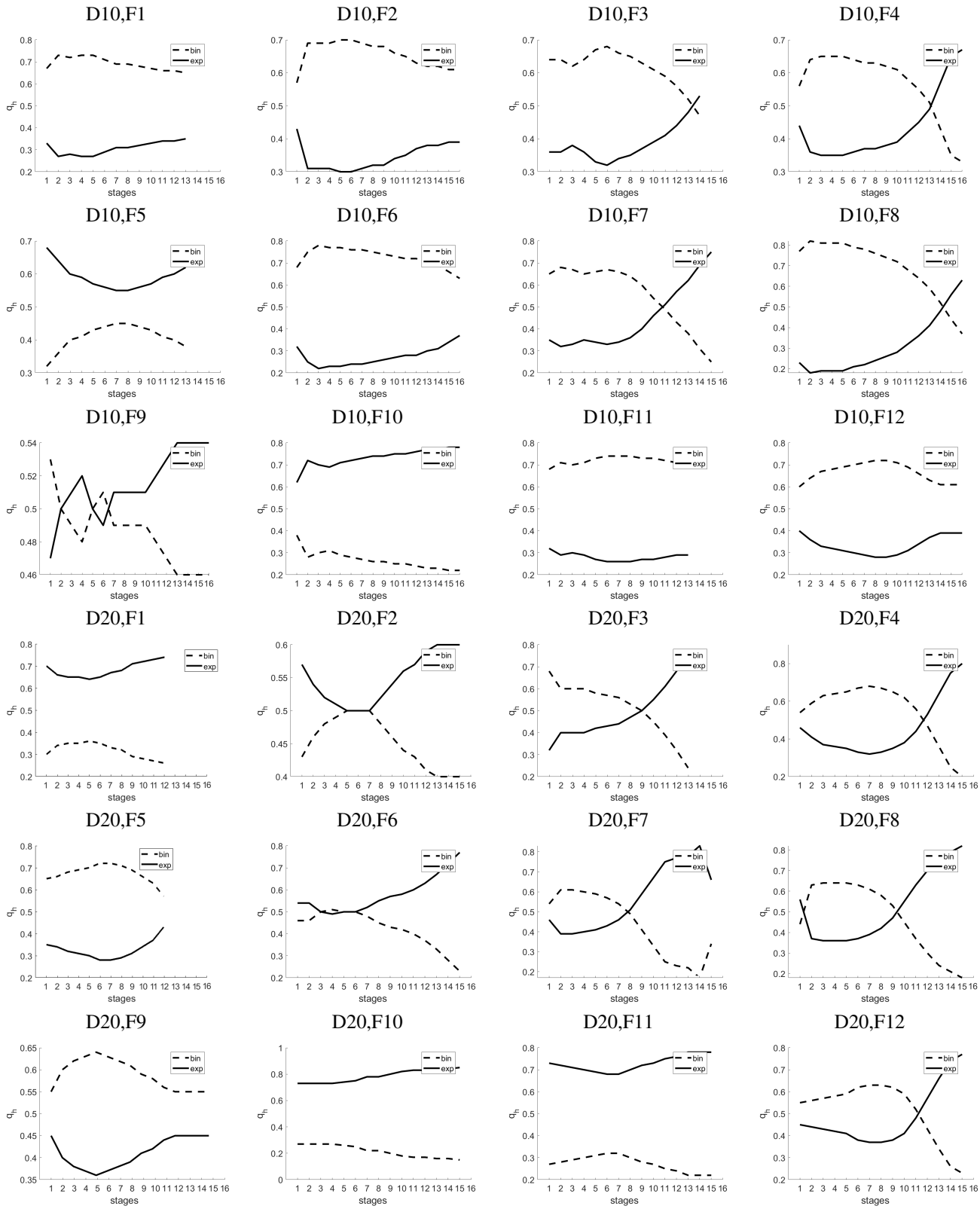

Fig. 3. Values of q_h for binomial (dash) and exponential (solid) crossover variants.