sciendo

TATRA MOUNTAINS
Mathematical Publications

DOI: 10.2478/tmmp-2018-0027
Tatra Mt. Math. Publ. **72** (2018), 155–165

IS DIFFERENTIAL EVOLUTION ROTATIONALLY INVARIANT?

Hana Zámečníková — Daniela Einšpiglová — Radka Poláková — Petr Bujok

University of Ostrava, Ostrava, CZECH REPUBLIC

ABSTRACT. In this paper, we study a problem of the control parameter settings in Differential Evolution algorithm and test a novel variant of the algorithm called CoBiDE. Although Differential Evolution with basic setting (i.e., $CR = 0.5$; $F = 0.5$) works quite well, it starts to fail on rotated functions. In general, we want to improve the convergence of algorithm primarily on rotated functions. It is done by adapting crossover parameter CR whereas parameter F is fixed to 0.5. There is a recommendation to set $CR = 1$ for rotated functions. It means that trial vectors are essentially composed from mutant. However, it is not easy task to set the parameters appropriately for solving optimization problem but it is crucial for obtaining good results. Moreover, the quality of points produced in evolution is highly affected by the coordinate system. In CoBiDE, the authors proposed a new coordinate system based on the current distribution of points in the population. We test these two approaches by running both algorithms on six pairs of rotated and non-rotated functions from CEC 2013 benchmark set in two levels of dimension space. This experimental study aims to reveal if such algorithm's setting is invariant under a rotation.

Introduction

Global optimization deals with an objective function

$$f\colon D \to \mathbb{R},\ D \subseteq \mathbb{R}^d.$$

We are searching for global minimum point $\mathbf{x}^* \in D$ where $f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in D$

© 2018 Mathematical Institute, Slovak Academy of Sciences.
2010 Mathematics Subject Classification: 68T20.
Keywords: global optimization, differential evolution, CEC2013 benchmark (set).
This work was supported by University of Ostrava from the internal grant projects SGS01/UVAFM/2017 and SGS06/UVAFM/2018.

and D is continuous space with boundary constrains

$$D = \prod_{i=1}^{d} [a_i, b_i], \qquad a_i < b_i, \quad i = 1, 2, \ldots, d.$$

As it is well-known, global optimization is NP-hard issue therefore some heuristic algorithm is often used for its solving. One of the most popular optimization algorithm is Differential Evolution (DE) since it is simple algorithm producing good results. During last years many authors tried to improve DEby changing of parameters' setting or they transformed its structure and consequently its code. The researchers experimented with control parameters' setting [5], diversity of population [3] or elimination of drift bias [4]. Moreover, successful DE variants were published such as EIG-L-SHADE [1] or CoBiDE [9].

This paper is focused on discussing two proposed approaches that should improve this algorithm and mainly make it more general-purpose. The paper has two experimental parts both aimed on the problem of rotated functions on which DE, as such as another heuristic algorithms, often fails. The first part is focused on setting of the parameter CR according to recommendation in [5], testing this setting and comparison with mainly used one. The second part deals with improved version of DE proposed in [9]. It is enhanced by covariance matrix learning and bimodal distribution parameter setting. The same experiments and comparison were done on both version of DE, original [5], [8] and enhanced [9].

1. Differential Evolution

Differential Evolution is population-based optimization algorithm that was developed by Rainer Storn and Kenneth Price in 1997 [8] and has only few control parameters. It is simple stochastic evolutionary algorithm using heuristic search in continuous space in order to find global optimum of an objective function. DE uses evolutionary operators, such as mutation, crossover, and selection.

1.1. Creating a new generation

First of all, a population of N random points in a search space D is uniformly generated ($\mathbf{x}_i, i = 1, 2, \ldots, N$). And then according to the pseudocode it is proceeded, as below.

It is important to say that the number of steps is limited. Usually DE is repeated until difference between the worst and the best function value is very small, often 1×10^{-8} or the number of function evaluations is greater than predefined fixed value.

Algorithm 1 Pseudocode of DE

```
evaluate f in each x_i, i = 1, 2, ..., N
while stopping condition not achieved do
    Q = ∅
    for i := 1 to N do
        generate mutation vector u_i
        generate a new trial vector y_i by crossover of u_i and x_i
        evaluate f in y_i
        if f(y_i) ≤ f(x_i) then
            insert y_i into Q
        else
            insert x_i into Q
        end if
    end for
    P := Q
end while
```

1.2. Mutation

Generating vector $\mathbf{u}$ (a linear combination of several points from population) represents the mutation in DE algorithm. There are several different types of mutation. The most frequently used is *rand*/1/ type, which generates vector $\mathbf{u}$ from three different points randomly selected from the population ($\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$), not equal to $\mathbf{x_i}$, as below.

$$\mathbf{u} = \mathbf{r}_1 + F(\mathbf{r}_2 - \mathbf{r}_3), \tag{1}$$

$F > 0$ is input parameter of mutation.

1.3. Crossover

The mutant vector $\mathbf{u}$ is one of the parental vectors for crossover. The second parent is current $\mathbf{x_i}$ from the actual generation. There are two types of crossover in DE – binomial and exponential, in this paper we use only the binomial one. In binomial crossover, new trial point $\mathbf{y}$ is created by mixing coordinates of both parental vectors as follows

$$y_j = \begin{cases} u_j, & R_j \leq CR \ \vee \ j = I, \\ x_{ij}, & R_j > CR \ \wedge \ j \neq I, \end{cases} \tag{2}$$

where I is random integer from $\{1, 2, \ldots, d\}$, $CR \in [0, 1]$ is crossover constant and $R_j \in (0, 1)$ is chosen randomly and independently for each $j \in \{1, \ldots, d\}$.

1.4. Selection

If the new point $\mathbf{y}$ is fulfilling the condition $f(\mathbf{y}) \leq f(\mathbf{x}_i)$, then $\mathbf{y}$ proceeds to the next generation and replaces the current vector $\mathbf{x}_i$. If not, then $\mathbf{x}_i$ stays for the next generation.

2. DE for rotated functions

Main goal of our paper is to explore the behavior of DE with setting recommended by authors of the original paper [5] where version DE/*rand*/1/*bin* of DE algorithm was used. Two settings of DE algorithm are compared here, most frequently used $CR = 0.5$ and $CR = 1$ recommended by [5]. The assumption was, that especially on rotated functions, $CR = 1$ should provide better solutions [5].

Benchmark test suite CEC 2013 was taken for this experiment, twelve functions were chosen in two levels of dimension ($d = 10, 30$) and 51 repeated runs for each function, dimension, and each of both versions of parameter CR ($CR = 1$, $CR = 0.5$) were done. Population size in DE was always dependent on dimension ($N = 5d$) and F was fixed on 0.5.

2.1. DE Results

The new setting of crossover parameter was proposed and implemented into the mostly used version od DE/*rand*/1/*bin* and tested on 12 functions in two dimensions. Table 1 illustrates how the recommended setting of CR ($CR = 1$) lead against to the classic one. Final comparison was done by Wilcoxon two-sample test for each function and each dimension with significance level 0.05. Symbol "+" means that setting $CR = 1$ was significantly better than that one with $CR = 0.5$, analogously "−" means that $CR = 1$ was worse that the classic one

TABLE 1. Results of statistical comparison for DE with $CR = 1$ and DE with $CR = 0.5$.

Number	Function	$d = 10$	$d = 30$
1	Rosenbrock's Function	−	−
2	Rotated Rosenbrock's Function	≈	−
3	Ackley's Function	−	−
4	Rotated Ackley's function	−	−
5	Griewank's Function	−	−
6	Rotated Griewank's Function	≈	−
7	Rastrigin's Function	−	−
8	Rotated Rastrigin's Function	≈	+
9	Schwefel's Function	−	−
10	Rotated Schwefel's Function	≈	≈
11	Lunacek BiRastrigin Function	−	−
12	Rotated Lunacek BiRastrigin Function	−	−

($CR = 0.5$), and on functions labeled with "≈", both compared versions were not significantly different. We got 24 pairs for comparison, but unfortunately positive effect is minimal, improvement occurs just for Rotated Rastrigin's Function in higher tested dimension ($d = 30$).

Results are not very satisfying, so it has been confirmed that rotated problems could not be solved satisfactory with the basic versions of DE algorithm. Further possibility how to increase efficiency of DE in rotated problems is using of CoBiDE.

3. CoBiDE

In 2014, Yong Wang et al. [9] published a novel DE. In brief, it is DE employing covariance matrix learning and bimodal distribution parameter setting named CoBiDE. Authors promise better performance of CoBiDE in comparison with original algorithm. As the results of first part of our experiment showed, to set DE parameters to appropriate fixed values is not an easy task but it is crucial for obtaining good results. Moreover, the fact that the quality of points produced in evolution is highly affected by the coordinate system is usually ignored. Authors in [9] proposed two significant modifications in solving these issues. Firstly, covariance matrix C of a part of population with less function values is computed. The matrix C reflects current population diversity and interactions among variables. Then by using eigenvector decomposition of C, the new coordinate system for search space is established in which the trial vector is generated. The crossover is done in the new coordinate system. From Fig. 1, it is obvious that in the new coordinate system the trial vector $\mathbf{y}$ can be closer to global minimum. All details including procedure of the covariance matrix learning can be found in [9].

Second modification is bimodal distribution of DE parameters setting, composed of Cauchy distributions as follows

$$F = \begin{cases} rand_{cauchy}(0.65, 0.1), & rand(0,1) < 0.5, \\ rand_{cauchy}(1.0, 0.1), & otherwise, \end{cases}$$

$$CR = \begin{cases} rand_{cauchy}(0.1, 0.1), & rand(0,1) < 0.5, \\ rand_{cauchy}(0.95, 0.1), & otherwise. \end{cases}$$

There are two new parameters ps and $peig$, both are real numbers from $[0, 1]$. The ps parameter determines proportion of population used for computing the covariance matrix needed for the coordinate system transformation, whereas the number $peig$ is a probability of applying this approach to computing new generation of population. See all the changes in pseudocode below.

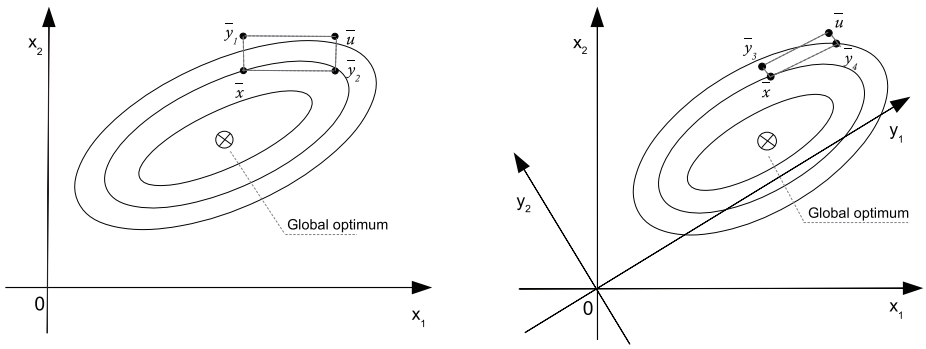

FIGURE 1. Search space and possible trial points in old and new coordinate system.

Algorithm 2 Pseudocode of CoBiDE

```
Generate an initial population P, (x_i, i = 1, 2, ..., N).
Evaluate f in each x_i, i = 1, 2, ..., N.
Generate initial values of F and CR for each element of the population.
while stopping condition not achieved do
    Q = ∅
    for i := 1 to N do
        apply the mutation operator to produce mutant vector u_i for the target vector x_i
    end for
    if rand(0, 1) < peig then
        for i := 1 to N do
            implement the crossover operator according to covariance matrix learning and produce trial vector y_i
        end for
    else
        for i := 1 to N do
            implement the crossover operator according to original coordinate system and produce y_i
        end for
    end if
    for i := 1 to N do
        evaluate function f value of y_i
        if f(y_i) ≤ f(x_i) then
            insert y_i into Q
            F and CR remain the same;
        else
            insert x_i into Q
            generate new F and CR
        end if
    end for
end while
```

4. CoBiDE results

CoBiDE was tested on the same 6 pairs of non-rotated and rotated functions from CEC 2013 benchmark set as it was used in the first part of our experiment, 51 independent runs were performed for each dimension, problem, and parameters' setting. The setting was following: dimensions $d = 10; 30$, population size $N = 5d$. Our goal was to test CoBiDE and its setting on rotated problems and compare its results with results of original DE on these problems. At first, we set up the parameters *ps* and *peig* according to values recommended in [9] $ps = 0.5$ and $peig = 0.4$. In order to explore the setting and possibilities of CoBiDE more, we also tested extreme values of parameters *ps* and *peig*: 0.1 and 0.9. We combined the extreme values with the recommended values and obtained more permutations. All settings and results (medians of function value errors) are shown in tables 2, 3, 4, and 5. It is necessary to add that errors smaller than 1×10^{-8} were replaced by 0. Tested functions are numbered according to Table 1, non-rotated functions are labelled by odd numbers and rotated functions by even numbers.

It seems that for functions 1–6 in dimension 10, the setting of *ps* and *peig* does not change the performance whatsoever. Also, errors for non-rotated and rotated version of the functions are very similar. It appears that CoBiDE is rotationally invariant on two of these three pairs of functions (Rosenbrock's and Ackley's).

TABLE 2. Medians of results for dimension 10, functions 1–6.

		Function number					
ps	*peig*	1	2	3	4	5	6
0.5	0.4	0	0	20.577	20.336	0	0.0369
0.5	0.1	0	0	20.477	20.364	3.91E-08	0.039
0.5	0.9	0	0	20.748	20.354	3.76E-08	0.027
0.1	0.1	0	0	20.524	20.362	3.872E-08	0.052
0.1	0.4	0	0	20.591	20.355	3.85E-08	0.043
0.1	0.9	0	0	20.743	20.337	3.814E-08	0.044
0.9	0.1	0	0	20.496	20.35	3.86E-08	0.0418
0.9	0.4	0	0	20.574	20.339	3.86E-08	0.032
0.9	0.9	0	0	20.68	20.349	3.728E-08	0.027
DE, $CR = 1$		8.25	31.241	20.477	17.539	0.598	0.154
DE, $CR = 0.5$		9.812	3.348	20.36	0	0.426	0

TABLE 3. Medians of results for dimension 10, functions 7–12.

		Function number					
ps	*peig*	7	8	9	10	11	12
0.5	0.4	0	6.96	0	460.51	10.122	18.459
0.5	0.1	0	7.96	0	517.76	10.122	17.687
0.5	0.9	0	5.97	66.87	403.13	13.862	16.658
0.1	0.1	0	7.96	0	472.6	10.122	17.649
0.1	0.4	0	6.965	0	469.04	10.122	16.398
0.1	0.9	0	6.965	62.34	405.56	13.695	16.902
0.9	0.1	0	7.96	0	444.4	10.122	17.518
0.9	0.4	0	6.97	0	394.17	10.122	17.346
0.9	0.9	0	5.97	66.19	369.38	13.283	16.548
DE, $CR = 1$		21.101	21.569	1347.5	1416.01	39.166	37.389
DE, $CR = 0.5$		0	22.42	0.258	1355.35	10.122	33.636

TABLE 4. Results for dimension 30, functions 1–6.

		Function number					
ps	*peig*	1	2	3	4	5	6
0.5	0.4	0.098	8.7681	21.096	20.951	3.5E-05	4.9E-05
0.5	0.1	0.109	12.254	21.06	20.963	5.02E-05	0.149
0.5	0.9	0.0758	6.302	21.14	20.944	3.16E-06	0
0.1	0.1	0.114	13.422	21.058	20.954	7.12E-05	0.342
0.1	0.4	0.107	12.711	21.087	20.945	1.09E-04	8.25E-03
0.1	0.9	0.0918	13.939	21.152	20.954	7.61E-05	7.41E-03
0.9	0.1	0.11	12.327	21.06	20.944	5.05E-05	0.136
0.9	0.4	0.0956	8.0909	21.075	20.948	2.66E-05	3.98E-05
0.9	0.9	0.0752	6.4325	21.17	20.96	1.07E-05	0
DE, $CR = 1$		117.67	126153	20.97	20.95	108.15	1.02
DE, $CR = 0.5$		16.213	21.711	20.939	20.51	8.723	0

TABLE 5. Results for dimension 30, functions 7–12.

		Function number					
ps	*peig*	7	8	9	10	11	12
0.5	0.4	3.2826	102.81	519.24	5822.03	47.44	205.814
0.5	0.1	1.63E-04	119.98	135.612	5781.25	35.575	217.59
0.5	0.9	71.542	83.232	3593.72	5706.96	138.81	193.58
0.1	0.1	1.84E-04	114.842	138.446	5757.39	35.485	213.285
0.1	0.4	4.2195	105.644	509.678	5887.88	48.067	212.68
0.1	0.9	74.409	82.826	3682.78	5646.68	144.23	204.553
0.9	0.1	1.50E-04	115.4	133.56	5748.27	35.401	215.402
0.9	0.4	3.093	105.28	490.43	5811.19	48.154	210.28
0.9	0.9	71.399	91.549	3695.32	5703.98	141.29	192.62
DE, $CR = 1$		169.87	174.71	7080.64	7277.77	225.17	233.50
DE, $CR = 0.5$		107.26	193.69	4485.57	7337.83	141.525	223.7

However, in dimension 30 we can observe difference in results for Rosenbrock's function (no. 1) and its rotated version (no. 2). It also has slightly better results for function no. 2 when the parameter $peig = 0.9$. Surprisingly, errors for functions 3–6 in dimension 30 are not distinctively different from results in dimension 10. There is a potential of algorithm's stability in bigger dimensions in this particular case.

Interesting observation is that the setting of *ps* and *peig* made some evident changes in function errors for functions 7–12. For instance, in dimension 10, the setting $peig = 0.9$ caused big errors in comparison with smaller *peig* for Schwefel's function. For Rotated Schwefel's function (no. 9), this approach is clearly unsuitable, although for its non-rotated function with proper setting it can find global minimum. For dimension 30, the results are even more interesting. See the best result for $peig = 0.1$ for non-rotated Rastrigin's function (no. 7), and the worst for Rotated Rastrigin's function (no. 8). While in dimension 10 for Rastrigin's function the error was always 0, in dimension 30 for $peig = 0.9$ there are distinct errors. The Rotated Rastrigin's function has smaller but still significant differences in errors. In similar way for the pair of Schwefel's functions, it is true that the worst setting $peig = 0.9$ for the non-rotated is the best for the rotated even though here the errors are very similar. The Lunacek BiRastrigin pair (no. 11, 12) has similar errors as the Schwefel's functions. Clearly solution of rotated problems requires individual approach and setting in this particular method.

If we compare CoBiDE with simple DE, then CoBiDE has often better results. For dimension 10, the errors of DE are obviously bigger, especially for setting $CR = 1$, for instance, for Schwefel's function, the error is 1347.5 whereas for CoBiDE it is at most 66.87 and could be even 0. However, we cannot miss the fact that for Ackley's functions (no. 3, 4) all results are almost the same except for DE with setting $CR = 0.5$ in dimension 10. This setting was the only successful at least for the rotated Ackley's function (no. 4). In dimension 30, the difference between classical DE and its novel is even visible. CoBiDE is mostly able to provide smaller errors than DE but it is clear that for some functions such as Rotated Schwefel's or Rotated Lunacek, it fails regardless the *ps*/*peig* setting. To sum up the results, we observe that CoBiDE despite its better perfomance than DE is not rotationally invariant since there are functions on which it fails. However, the setting of parameters *ps* and *peig* promises positive changes in the field of optimization and give a reason for further research.

5. Conclusion

Differential Evolution became popular during last years because it is efficient and it has a simple code. In order to improve the performance of DE, many authors attempted to transform the algorithm by particular control parameter's setting or by changing the code, e. g. CoBiDE. Our goal was to test anticipated performance of modified algorithm according to [5] and [9]. Main motivation was to find rotationally invariant setting of DE. Experiment was performed on six pairs benchmark test functions developed for CEC 2013 competition.

First part of our experiment, where $CR = 1$, did not confirm our expectation. Improvement occurred only on one function and on the other functions there was no effect or the results were even worse. This suggestion is not rotationally invariant therefore in the second part we tested several parameter settings of CoBiDE algorithm in order to verify parameter sensitivity. In this case, the results were visibly better, however, the algorithm is not rotationally invariant. CoBiDE established new way how to solve optimization problems of rotated functions since particular parameter setting provided promising results. Proper setting and using of covariance-matrix-based crossover should be further studied.

Acknowledgement. The authors sincerely thank the reviewers for their constructive and very helpful comments and suggestions.

REFERENCES

[1] GUO, S.-M.—YANG, C.-C.— TSAI, J. S.-H.—HSU, P.-H.: *A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful framework on CEC 2015 benchmark set.* In: Proceedings of the IEEE Congress on Evolutionary Computation—CEC '15, Sendai, Japan, 2015, IEEE, New York, pp. 1003–1010.

[2] LIANG, J. J.—QU, B-Y.—SUGANTHAN, P. N.—HERNÁNDEZ-DÍAZ, A. G.: *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization*. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, Tech. Rep. 2012, `http://www.ntu.edu.sg/home/epnsugan/`.

[3] POLÁKOVÁ, R.—TVRDÍK, J.—BUJOK, P.: *Population-size adaptation through diversity-control mechanism for differential evolution*, MENDEL 2016, Brno University of Technology, 2016. pp. 49–56.

[4] PRICE, K.: *Eliminating drift bias from the differential evolution algorithm*. In: *Advances in Differential Evolution, Studies in Computational Intelligence*, Vol. 143, Springer-Verlag, Berlin, 2008, pp. 33–88.

[5] ——— *How symmetry constrains evolutionary optimizers*. In: Proc. of the IEEE Congress on Evolutionary Computation—CEC '17, Donostia-San Sebastián, Spain, 2017, IEEE, New York, pp. 1712–1719.

[6] PRICE, K.—STORN, R. M.—LAMPINEN, J. A.: *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[7] STORN, R.: *Differential evolution research—trends and open questions*. In: *Advances in Differential Evolution*, Springer-Verlag, Berlin, 2008, pp. 1–31.

[8] STORN, R.—PRICE, K.: *Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces*. J. Global Optim. **11** (1997), 341–359.

[9] WANG, Y.—LI, H.-X.—HUANG, T.—LONG, L.: *Differential evolution based on covariance matrix learning and bimodal distribution parameter setting*, Appl. Soft Comput. **18** (2014), 232–247.

Received December 5, 2017

Hana Zámečníková *
Daniela Einšpiglová **
Department of Matematics
Faculty of Science
E-mail: p18113@student.osu.cz *
p18111@student.osu.cz **

Radka Poláková
Centre of Excellence IT4Innovations
Institute for Research and
Applications of Fuzzy Modeling
E-mail: radka.polakova@osu.cz

Petr Bujok
Department of Informatics and Computers
Faculty of Science
E-mail: petr.bujok@osu.cz

University of Ostrava
30. dubna 22
CZ–701-03 Ostrava
CZECH REPUBLIC