FOUNDATIONS

A novel mutation operator based on the union of fitness and design spaces information for Differential Evolution

Hossein Sharifi-Noghabi1,2 · Habib Rajabi Mashhadi1,2,3 · Kambiz Shojaee1,2

Published online: 20 September 2016
© Springer-Verlag Berlin Heidelberg 2016

Abstract Differential Evolution (DE) is one of the most successful and powerful evolutionary algorithms for global optimization problem. The most important operator in this algorithm is mutation operator, in which parents are selected randomly to participate in it. Recently, numerous papers are tried to make this operator more intelligent by selection of parents for mutation intelligently. The intelligent selection for mutation vectors is performed by applying design space (also known as decision space) criterion or fitness space criterion; however, in both cases, half of valuable information of the problem space is disregarded. In this article, a Union Differential Evolution (UDE) is proposed which takes advantage of both design and fitness spaces criteria for intelligent selection of mutation vectors. The experimental analysis on UDE are performed on CEC2005 benchmarks and the results stated that UDE significantly improved the performance of DE in comparison with other methods that only use one criterion for intelligent selection.

Keywords Differential Evolution · Union Differential Evolution · Intelligent selection · Ranking-based mutation · Proximity-based mutation

1 Introduction

In order to solve complicated computational problems, researchers and scientists are taking advantage of nature and inspiring from it for finding an intelligent, innovative, and effective solutions, and evolutionary computation is one of the most important results of this inspiration (Bäck 1996; Michalewicz 1996). In an iterative process, an evolutionary algorithm moves a single individual or a population toward the global optimum. Algorithms such as genetic algorithm (GA), particle swarm optimization (PSO), and simulated annealing (SA) are examples of evolutionary and meta-heuristic algorithms (Thomas et al. 1997; Boussaïd et al. 2013).

Among these methods, Differential Evolution (DE) is one of the most effective and powerful algorithms for continuous global optimization problem (Price 1999). DE was originally proposed in 1995 by Storn and Price (1997). DE is based on population and similar to other evolutionary methods, and it has some operators for handling exploration and exploitation in its optimization process. There are numerous advantages about DE including, its simplicity in implementation, a few control parameters, and the fact that it is intersection of classic methods such as Nelder–Mead and controlled random search and early evolutionary algorithms such as GA and SA (Price et al. 2005; Das and Suganthan 2011; Neri and Tirronen 2010).

The performance of DE significantly relays on two things: First, the strategy of generating new offspring which is performed by mutation and crossover operators, and second,
the mechanism of controlling the parameters of the algorithm including, number of population, scaling factor, and crossover rate (Brest et al. 2008). Generally, in the process of evolution and optimization, the first step (generating new offspring) takes place and after that parameters of the algorithm will be tuned, and these steps are repeating until the algorithm reaches its stopping condition (Das et al. 2014; Wei-Jie et al. 2014).

In order to enhance the performance of DE, in recent years, intelligent selection of mutation vectors received a lot of attention in DE literature (Das and Suganthan 2011; Biswas et al. 2014; Gong and Cai 2013). Usually, parent vectors in mutation are selected randomly from the population, and since this strategy only deals with exploration, applying the best member of population (Price et al. 2005), p% best (Jingqiao and Sanderson 2009), current members, are suggested later to provide DE with exploitation as well as exploration (Matej et al. 2013).

Intelligent selection methods generally take advantage of information in either design or fitness space (Das and Suganthan 2011), for instance, Kaelo and Ali applied tournament selection in fitness space (Kaelo and Ali 2006), Epitropakis et al. (2011) defined the selection criterion in design space and Gong and Cai (2013) proposed a ranking approach in fitness space for intelligent selection for mutation vectors. Although all of these methods have their own merits, the main drawback of them is that they disregard half of the valuable information for determination of mutation vectors. In nature, good species have good information, and consequently they can guide the whole population to the global optimum (Gong and Cai 2013). Based on this fact, in this article, Union Differential Evolution (UDE) is proposed that makes the best use of information in both design and fitness spaces in mutation operator for selection of parent vectors. Among five parent vectors in the proposed mutation, two of them are selected with fitness space criterion, two of them are selected randomly, and the last one is chosen based on design space criterion.

Some other DE variants use more than one mutation in similar idea of concepts like boosting or ensemble learning. Although such methods are beyond the score of this paper, a brief overview can be helpful. Self-adaptive DE, SaDE (Qin et al. 2009), uses both multiple trial vector strategies, i.e., multiple mutations and at the same time a learning mechanism to update parameters associated with them. Wang et al. proposed composite DE, CoDE (Yong et al. 2011), which improved performance of DE by applying three mutation operators and three control parameter settings. Mallipeddi and Suganthan (2010) proposed EPSDE which has an ensemble mutation and crossover strategies along with a pool for controlling parameters. Later, Iacca et al. (2014) introduced EPSDE-LS with a pool of local search algorithms and proposed a new framework for ensemble DE. Noghabi et al. (2015) proposed DE with generalized mutation operator, GMDE, which has two mutation pools with four trial vector generating strategies in each and applied it for parameter optimization in gene selection problem (Mohammadi et al. 2016). More interested readers can refer to Das et al. (2016).

The rest of the paper is organized as follows: Sect. 2 is the brief review of previous research about DE and intelligent selection, Sect. 3 deals with the proposed method, and Sect. 4 is about experimental results and analysis.

2 Related work

In this section, first, the original DE is briefly introduced and then related works regarding intelligent selection are discussed.

Without loss of generality, throughout this paper, the following numerical optimization problem is considered:

Minimize \( f(x), \quad x \in \text{CS} \) \hspace{1cm} (1)

where \( \text{CS} \subset \mathbb{R}^D \) is a compact set, \( x = [x_1, x_2, \ldots, x_D]^T \), and \( D \) is the dimension of the problem space. Moreover:

\[
\text{LB} \leq x_j \leq \text{UB}, \quad j = 1, 2, \ldots, D
\] \hspace{1cm} (2)

where LB and UB are lower bound and upper bound, respectively.

2.1 Original DE

Similar to other evolutionary computation methods, DE starts with an initial population that generally initialized randomly. After determining the population, a new candidate individual is generated by applying mutation and crossover operators (Price et al. 2005). This candidate then becomes the input of selection operator, and through a hard selection mechanism between the candidate and the current member of the population, if the candidate is better than the current member, it will enter the next generation; otherwise, the current member remains in the population (Das and Suganthan 2011). Algorithm 1 is the pseudocode of the original DE. In this algorithm, NP is the population size, \( D \) is size of dimension, \( F \) is the scaling factor, \( CR \) is the crossover rate, \( \text{randint}(1, D) \) is a random integer in \([1, D]\), and \( \text{rand} \) is a random real number in \([0, 1]\).
2.2 Mutation operators in DE

The main operator in DE is mutation operator and a lot of mutations have been proposed in DE literature from 1997 up to now (Das and Suganthan 2011; Neri and Tirronen 2010). These mutations take advantage of diverse mechanisms and strategies for generating the donor vector. In order to distinguish between DE variants, notation DE/X/Y/Z proposed by Storn and Price where DE denotes Differential Evolution, X denotes the base vector, Y denotes number of difference vectors and finally Z determines the type of crossover (Storn and Price 1997). In this paper, since in all of the cases, the binomial crossover (bin) is applied, and therefore, Z is omitted from the notation.

Some of the well-known mutations are listed as follows:

DE/rand/1

\[ X_{\text{new},g} = X_{r1,g} + F \cdot (X_{r2,g} - X_{r3,g}) \]  

(3)

DE/best/1

\[ X_{\text{new},g} = X_{\text{best}} + F \cdot (X_{r1,g} - X_{r2,g}) \]  

(4)

DE/rand/2

\[ X_{\text{new},g} = X_{r1,g} + F \cdot (X_{r2,g} - X_{r3,g}) + F \cdot (X_{r4,g} - X_{r5,g}) \]  

(5)

DE/best/2

\[ X_{\text{new},g} = X_{\text{best}} + F \cdot (X_{r1,g} - X_{r2,g}) + F \cdot (X_{r3,g} - X_{r4,g}) \]  

(6)

DE/current-to-best/1

\[ X_{\text{new},g} = X_{\text{current},g} + F \cdot (X_{\text{best}} - X_{\text{current},g}) + F \cdot (X_{r2,g} - X_{r3,g}) \]  

(7)

DE/rand-to-best/1

\[ X_{\text{new},g} = X_{r1,g} + F \cdot (X_{\text{best}} - X_{r1,g}) + F \cdot (X_{r2,g} - X_{r3,g}) \]  

(8)

DE/current-to-rand/1 \((F' = k_i \cdot F)\)

\[ X_{\text{new},g} = X_{\text{current},g} + k \cdot (X_{r1,g} - X_{\text{current},g}) + F' \cdot (X_{r2,g} - X_{r3,g}) \]  

(9)

where in these mutations, \(X_{\text{new},g}\) is the newly generated member, \(X_{\text{best}}\) indicates the best individual, \(X_{\text{current}}\) denotes the current member of the population at generation \(g\), \(X_{r1} \neq X_{r2} \neq X_{r3} \neq X_{r4} \neq X_{r5} \neq X_{\text{current}}\) and \(F\) is the scaling factor. \(K\) is a constant weight between the engaged terms.

2.3 Intelligent selection for mutation

Intelligent selection has been under consideration since the original DE was proposed, in fact, some methods such as DE/best1 and DE/current-to-best/1 can be considered examples of this category of mutations. Although Das and Suganthan (2011) introduced intelligent selection as one of the promising area in DE, more serious and deep studies started from 2006 by Kaelo and Ali (2006), and they proposed DERL which applies tournament selection to select three random member of population and then the best of them is placed as the base vector and the other two as difference vectors in DE/rand/1 mutation. Jingqiao and Sanderson (2009) proposed DE/current-to-pbest/1 in JADE that uses \(p\%\) of the best members of population and moreover; it also applies an archive of failed answers to participate in mutation for more diversity. The same approach also introduced in Baatar et al. (2013). Epitropakis et al. (2011), presented a proximity-based mutation that takes advantage of design space criterion for intelligent selection. In this method, first
affinity matrix \((NP \times NP)\) is calculated based on Euclidean distance of individuals from each other and then the selection probabilities are calculated. The closer an individual to the current member is, the more probability it has. Finally, on each row of the probabilities, matrix individuals are chosen by roulette wheel selection. In Garcia-Martínez et al. (2011), Garcia et al. proposed the role differentiation and malleable mating for mutation. In this approach, the vectors in the population are grouped into four groups, including receiving, placing, leading, and correcting groups. For mutation and crossover operators, vectors are selected from corresponding groups, instead of the whole population. Gong and Cai (2013) presented ranking-based mutation which uses fitness space as the criterion for intelligent selection. In ranking, individuals are sorted and then each individual receives a rank based on its fitness. The probability of selection for each individual is calculated according to its corresponding rank in the population. For a mutation such as DE/rand/1 two of the three vectors are chosen by ranking criterion via a proportional selection method. In Liang et al. (2014), Liang et al. proposed an intelligent selection with fitness Euclidean distance ratio (FER) for multimodal optimization that probabilities of selection are calculated based on this measure.

3 Proposed method

In this paper, we proposed a novel mutation operator, Union DE that has both merits of design and fitness spaces in parent vectors. Since good individuals in the population have valuable information, they can guide the rest of the population much better, and thus, it is important to use valuable information of both spaces. In this section, first the proposed mutation, UDE is introduced and then some definitions and criteria are stated regarding it.

3.1 UDE

In the spirit of DE/rand/2 mutation, UDE is defined as follows:

\[
v_i = X_{FS1} + F \cdot (X_{FS2} - X_{r1}) + F \cdot (X_{DS} - X_{r2}) \tag{10}
\]

where \(v_i\) is the donor vector, \(F\) is the scaling factor, \(X_{FS1}\) is the parent vector chosen by fitness space criterion, \(X_{DS}\) is the vector selected by design space criterion, and \(X_{ri}\) is a randomly selected vector.

Terminology 1: **Base role**: a vector has base role if it is in the first position of mutation operator. For example, \(X_{FS1}\) has the base role in Eq. (10).

Terminology 2: **Leading role**: a vector has leading role if it is in the first place of a difference vector. In Eq. (10), \(X_{FS2}\) and \(X_{DS}\) have this role.

Terminology 3: **Terminal role**: a vector has terminal role if it is in the second position of a difference vector. For example, \(X_{r1}\) and \(X_{r2}\) in Eq. (10) are terminal vectors.

Remark 1 In contrast with ranking and proximity mutations, UDE takes advantage of intelligent selection for determination of base and leading roles in mutation and this approach provides the algorithm with better performance.

3.2 Fitness space criterion

In order to select \(X_{FS1}\) and \(X_{FS2}\) vectors, following steps are required:

First, sort the population in increasing order (from best to worst) according to their fitness value.

Second, calculate the selection probability for each individual by:

\[
P_i = \frac{NP - i}{NP}, \quad i = 1, 2, \ldots, NP \tag{11}
\]

where \(P_i\) is the selection probability for the \(i\)-th member and NP is number of population.

After measuring selection probabilities, two members are chosen by roulette wheel and randomly assigned to stated roles for fitness space vectors. Therefore, the criterion for fitness space is calculated.

3.3 Design space criterion

For design space selection, based on Euclidean distance between all of the individuals in the population, the distance matrix, \(DM\) is obtained as follows:

\[
DM = \begin{pmatrix}
    \|X_1 - X_1\| & \cdots & \|X_1 - X_{NP}\| \\
    \vdots & \ddots & \vdots \\
    \|X_{NP} - X_1\| & \cdots & \|X_{NP} - X_{NP}\|
\end{pmatrix} \tag{12}
\]

Due to symmetric property of distance, \(DM\) is a symmetric and only its upper triangular part requires to be calculated. Based on \(DM\), we calculate the probability matrix, \(PM\), by the following equation:

\[
PM(i, j) = 1 - \frac{DM(i, j)}{\sum_k DM(i, k)} \tag{13}
\]

In this matrix, \(PM(i, j)\) represents the probability between \(i\)-th and \(j\)-th individual with respect to the \(i\)-th row of \(PM\). This probability is inversely proportional to the distance of the
j-th member of the population which means the member with minimum distance to i-th member will have the maximum probability. In order to choose $X_{DS}$, roulette wheel selection without replacement is performed on every row of PM matrix (for each member of population).

Remark 2 Whenever a new member is entered to the population, only row and column associated with this new individual are required to be updated, and it is not necessary to calculate the entire matrices.

Remark 3 Applying Eq. (13) can be considered as a local search as well because it exploits the regions around a pre-defined member by assigning the higher probability to closer individuals. Hence, this approach makes UDE competitive for multimodal problems too. Algorithm 2 is pseudocode for UDE.

Algorithm 2 Union DE (UDE)
1: Generate an initial population consisting of $NP$ individuals and evaluate them.
2: While (termination criterion is not satisfied)
3:    For $i = 1: NP$
4:        Select randomly $r_1 \neq r_2 \neq i$
5:        select $X_{FS1}$ and $X_{FS2}$ according to Eq. (11) and roulette wheel.
6:        select $X_{DS}$ according to Eqs. (12-13) and roulette wheel.
7:        Perform UDE mutation according to Eq. (10).
8:        Perform binomial crossover similar to Algorithm 1.
9:    End for
10:   End if
11:   End for
12: Evaluate the offspring $x_i$
13: If $f(u_i) < f(x_i)$ then
14:    Replace $x_i$ with $u_i$
15: End if
16: End for
17: End while

4 Experimental results and analysis

In this section, in order to investigate and study the performance of the presented UDE, comprehensive experiments are performed. We applied 25 benchmark functions from CEC2005 competition (Suganthan et al. 2005). These functions are categorized into four groups: (1) F1–F5: These functions are unimodal, (2) F5–F12: These functions are basic multimodal functions, (3) F13 and F14 are expanded multimodal functions and finally (4) F15–F25 are hybrid composition functions.

4.1 Parameter settings

Since the value of parameters is totally influential on the performance of the algorithm, in all cases, we utilized jDE control parameter mechanism to have fair situation for all of the algorithms. jDE (Brest et al. 2006). Control scaling factor and crossover rate are as follows:

$$F_{i,G+1} = \begin{cases} F_{1,G} + \text{rand1} \ast F_u & \text{rand2} < \tau_1 \\ F_{1,G} & \text{otherwise} \end{cases}$$

$$C_{r_i,G+1} = \begin{cases} \text{rand3} & \text{rand4} < \tau_2 \\ C_{r_i,G} & \text{otherwise} \end{cases}$$

where $\text{rand1}, \text{rand2}, \text{rand3}, \text{rand4} \in [0,1]$, $\tau_1 = \tau_2 = 0.1$, $F_1 = 0.1$, $F_u = 0.9$

This approach makes $F \in [0.1, 0.9]$ and $Cr \in [0, 1]$.

The other parameters are initialized as follows: $NP = 50$; Maximum run = 50; $D$ (dimensions) = 30 and maximum number of function evaluation = $D \ast 10,000$.

4.2 Statistical analysis

In this paper, we used Wilcoxon signed-rank test at $\alpha = 0.05$ for comparing proposed UDE with other methods. According to this analysis, win denotes number of functions that UDE performed better than the compared method, lose denotes number of functions that UDE was not successful compared to the corresponding method, and tie denotes number of functions that UDE and the compared method are equal.

4.3 Results and experiments

In order to study and performance of the presented UDE, we selected DERL, proximity-based and ranking-based mutations to compare with. All of these methods take advantage of DE/rand/2 mutation that developed by their own approaches. Each method executed 50 times independently for the stated settings and the average value of the errors for each function is presented in Table 1. In this table, boldface indicates that for the corresponding function that algorithm achieved the best result among the other methods. According to this table, with respect to other methods, UDE significantly overcomes the other methods in most of the cases. Compared to DERL, UDE wins in 16 functions, ties in 2, and lost in 7 functions which most of successes of DERL were in unimodal and multimodal functions. UDE also wins in 15 functions in comparison with Proximity-based mutation, ties in 3, and lost in 7 functions similar to DERL. In comparison with ranking-based mutation, UDE again wins in 16 cases.
Table 1  Comparison of the performance between UDE, DERL, proximity and ranking for functions F1–F25 at D = 30 and 300,000 function evaluation for 50 independent runs

| Fun. | DERL mean | Proximity mean | Ranking mean | UDE mean |
|------|-----------|----------------|--------------|----------|
| F1   | 3.26E–28  | 5.05E–31       | 0            | 0        |
| F2   | 1.31E–23  | 3.61E–10       | 0.0162176    | 2.92E–10 |
| F3   | 74,543.26 | 164,945.8      | 379,023.67   | 142,905.61 |
| F4   | 254,116.04| 0.8932487      | 8.6211449    | 0.013536 |
| F5   | 1806.9985 | 476.52098      | 1043.8789    | 278.8043 |
| F6   | 13.329272 | 0.5722474      | 34.920037    | 0.318399 |
| F7   | 4696.2886 | 4696.2886      | 4692.478     | 4696.2886 |
| F8   | 20.93327  | 20.935417      | 20.954167    | 20.91975 |
| F9   | 4.7360039 | 0              | 0            | 0        |
| F10  | 56.23388  | 49.691529      | 58.571746    | 46.94091 |
| F11  | 23.833349 | 27.718178      | 27.698503    | 27.41882 |
| F12  | 12,518.51 | 16,496.466     | 16,000.211   | 14,155.216 |
| F13  | 12.23211  | 12.912602      | 12.948416    | 12.910522 |
| F14  | 370.1511  | 352.17827      | 242.8361     | 376      |
| F15  | 137.22539 | 69.01356       | 103.23722    | 68.16309 |
| F16  | 154.06222 | 133.9956       | 152.5687     | 148.4484 |
| F17  | 911.45675 | 904.5121       | 904.78488    | 904.4761 |
| F18  | 912.80443 | 904.5984       | 904.67019    | 904.71551 |
| F19  | 912.33533 | 904.94622      | 904.81682    | 904.4761 |
| F20  | 539.8539  | 512.00004      | 500          | 500      |
| F21  | 891.98448 | 881.90812      | 878.71601    | 874.3584 |
| F22  | 589.56758 | 534.16419      | 534.1641     | 534.16425 |
| F23  | 200       | 200            | 200          | 200      |
| F24  | 1617.594  | 1626.0676      | 1631.2113    | 1626.8149 |
| F25  | 1617.594  | 1626.0676      | 1631.2113    | 1626.8149 |

Win: 16 Win: 15 Win: 16 Lose: 7 Lose: 7 Lose: 5 Tie: 2 Tie: 3 Tie: 4

and ties in 4, and only loses in 5 functions. In case of proximity and ranking, they appeared to be more competitive in hybrid functions (F15–F25). From another point of view, in functions 1–5 (unimodals) in three of them including F1, F4 and F5 UDE had the best answer among the others, ranking find the best answer in one of the cases (F1) and DERL was the best in two among the other methods (F2 and F3), thus, for unimodal functions, utilizing both design and fitness spaces leads to better performance. For basic multimodal functions, ranking had the best performance in two functions including F7 and F9, proximity found the best answer in F9 too and DERL only found the best answer in F12. For this functions, again UDE had the best performance and obtained the best answer in five functions (F6, F8, F9, F10, and F11). Hence, for this group, UDE with its hybrid approach for intelligent selection tremendously overcomes the other methods. For DERL and proximity were better than other methods and won in F14 and F13 respectively. In case of hybrid composition functions, all of the methods performed the same for F24, so, beside this function, DERL was successful only in F25 ranking achieved the best answer among others in F15, F21, and F23. However, proximity performed generally better in comparison with ranking and DERL and found the best answer in F17, F18, and F19. Although both ranking and proximity relatively performed well for hybrid composition functions especially proximity, UDE again obtained the best results among other methods and found the best answer in F16, F20, F21, and F22. In addition to its superiority, for rest of the functions and in most of them, UDE achieved closer answers that were extremely close to the best ever found for that function. Therefore, hybrid approach for intelligent selection significantly improved the performance of the algorithm for basic multimodal and hybrid composition functions. Although UDE found relatively adequate minimums, it did not act well enough in expanded multimodal functions. It also appeared for complex functions such as F15–F25, after hybrid approach presented in UDE, design space criterion (proximity) performed better than fitness space approach.

In summary, among 24 functions (without considering F24) DERL found the best answer in five of the cases,
proximity performs the same as DERL and found the best answer in five functions but statistically, proximity performs significantly better than DERL. Ranking found the best in six functions but, statistically, there is no meaningful difference between proximity and ranking. UDE achieved the best performance in 12 functions and more importantly, statistical analysis regarding its performance indicates that in all of comparisons, UDE is better than the other methods and there is significant and meaningful difference between them in favor of UDE. Table 2 is presented the statistical details regarding UDE. In this table, MR− indicates mean of negative ranks, MR+ indicates mean of positive ranks, SR− denotes sum of negative ranks, SR+ denotes sum of positive ranks. P value is the measure that determines the difference between algorithms is significant or not and finally in the last column, “+” indicates that UDE is significantly better than the compared method, “−” indicates that UDE is significantly worse than the compared method and “=” indicates that there is no meaningful difference between UDE and the corresponding method.

As illustrated in this table, in all of the cases, UDE is significantly better than the other algorithms which means that there is a meaningful difference between the proposed method and the other algorithms in favor of UDE. Therefore, the proposed method, make the best use of good information in design and fitness spaces and this usage is leading to an effective, robust and accurate global optimization method which is capable of working with diverse kinds of problems.

### 4.4 Comparison with two basic mutations

UDE might be considered similar to DE/best/2 in a way. Therefore, one interesting study can be comparison of the proposed method with such a basic mutation. Results for this study are tabulated in Table 3 and as expected, UDE had significantly better performance than DE/best/2 method in most of the cases. In order to save space and avoid unnecessary large tables, we only report the results of the statistical analysis for this study.

### 5 Conclusion and future works

In this paper, we studied the impact of intelligent selection for mutation operator. Intelligent selection is a new generation in DE literature which is trying to instead of choosing parent vectors randomly, select them intelligently. Current methods utilize design space or fitness space criteria; however, the consequent of such a selection is losing half of the valuable information available in the problem space. In this paper, we proposed Union Differential Evolution (UDE) which takes advantage of both stated spaces in a single mutation strategy. In the presented mutation, there are five vectors, two of them are selected according to fitness space criterion, another two vectors are selected randomly and the last one is chosen by design space criterion; therefore, both exploration and exploitation are provided in UDE.

The results are illustrated that UDE is significantly better than methods that solely use fitness or design space criteria in their evolution process. All extensive experiments were performed on 25 benchmark functions from CEC2005 competition.

In future, we intend to study the impact of population sizes on the performance of the proposed method, more importantly, the proposed method should be examined with other control parameter approaches. Moreover, it is essential to investigate the performance of the proposed method for higher dimensions as well.

### Acknowledgements

The authors would like to thank Center of Excellence on Soft Computing and Intelligent Information Processing (SCIPP) for kind supports and Dr. Y. Wang for making the MATLAB code of jDE available online.

### Compliance with ethical standards

**Conflict of interest** Authors have nothing to disclose regarding conflict of interests.
References

Baatar N, Dianhai Z, Chang-Seop K (2013) An improved differential evolution algorithm adopting λ-best mutation strategy for global optimization of electromagnetic devices. IEEE Trans Magn 2097–2100. doi:10.1109/TMAG.2013.2240284

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford

Biswas S, Kundu S, Das S (2014) An Improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution. IEEE Trans Cybern 1726–1737. doi:10.1109/TTCYB.2013.2292971

Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization meta-heuristics. Inf Sci 82–117. doi:10.1016/j.ins.2013.02.041

Brest J, Zamuda A, Boškovic B, Greiner S, Žumer V (2008) An analysis of the control parameters’ adaptation in DE. In: Chakraborty U (ed) Advances in differential evolution. Springer, Berlin, pp 89–110

Brest J, Greiner S, Boskovic B, Mernik M, Žumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evolut Comput 646–657. doi:10.1109/TEVC.2006.872133

Das S, Mandal A, Mukherjee R (2014) An adaptive differential evolution algorithm for global optimization in dynamic environments. IEEE Trans Cybern 966–978. doi:10.1109/TTCYB.2013.2278188

Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution—an updated survey. Swarm Evolut Comput 1–30. doi:10.1016/j.swevo.2016.01.004

Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evolut Comput 4–31. doi:10.1109/TEVC.2010.2059031

Epitropakis MG, Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2011) Enhancing differential evolution utilizing proximity-based mutation operators. IEEE Trans Evolut Comput 99–119. doi:10.1109/TEVC.2010.2083670

García-Martínez C, Rodríguez F, Lozano M (2011) Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation. Soft Comput 2109–2126. doi:10.1007/s00500-010-0641-8

Gong W, Cai Z (2013) Differential evolution with ranking-based mutation operators. IEEE Trans Cybern 2066–2081. doi:10.1109/TTCYB.2013.2239988

Iacca G, Neri F, Caraffini F, Suganthan PN (2014) A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms. In: Esparcia-Alcázar IA, Mora MA (eds) Applications of evolutionary computation: 17th European conference, EvoApplications 2014, Granada, Spain, April 23–25, 2014, revised selected papers. Springer, Berlin, pp 615–626

Jingqiao Z, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evolut Comput 945–958. doi:10.1109/TEVC.2009.2014613

Kaelo P, Ali MM (2006) A numerical study of some modified differential evolution algorithms. Eur J Oper Res 1176–1184. doi:10.1016/j.ejor.2004.08.047

Liang JJ, Qu BY, Mao XB, Niu B, Wang DY (2014) Differential evolution based on fitness Euclidean-distance ratio for multimodal optimization. Neurocomputing 252–260. doi:10.1016/j.neucom.2013.03.069

Mallipeddi R, Suganthan PN (2010) Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies. In: Panigrahi BK, Das S, Suganthan PN, Dash SS (eds) Swarm, evolutionary, and memetic computing: first international conference on swarm, evolutionary, and memetic computing, SEMCCO 2010, Chennai, India, December 16–18, 2010 proceedings. Springer, Berlin, pp 71–78

Matej, C, Liu S-H, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput Surv 1–33. doi:10.1145/2480741.2480752

Michalewicz Z (1996) Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, Berlin

Mohammadi M, Sharifi Noghabi H, Abed Hodtani G, Rajabi Mashhadi H (2016) Robust and stable gene selection via maximum–minimum correntropy criterion. Genomics 83–87. doi:10.1016/j.ygeno.2015.12.006

Neri F, Tirronen V (2010) Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 61–106. doi:10.1007/s10462-009-9137-2

Noghabi HS, Mashhadi HR, Shohaji K (2015) Differential evolution with generalized mutation operator. arXiv preprint arXiv:1510.02516

Price KV (1999) An introduction to differential evolution. In: David C, Marco D, Fred G, Dipankar D, Pablo M, Riccardo P, Kenneth VP (eds) New ideas in optimization. McGraw-Hill Ltd, UK, pp 79–108

Price K, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization (natural computing series). Springer, New York

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolut Comput 398–417. doi:10.1109/TEVC.2008.927706

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 341–359. doi:10.1023/A:1008202821328

Suganthan PN, Hansen N, Liang JJ, Deb Y-PCK, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization. Nanyang Technol University, Singapore

Thomas B, David BF, Zbigniew M (1997) Handbook of evolutionary computation. IOP Publishing Ltd, Bristol

Wei-Jie Y, Mei S, Wei-Neng C, Zhi-Hui Z, Yue-Jiao G, Ying L, Ou L, Jun Z (2014) Differential evolution with two-level parameter adaptation. IEEE Trans Cybern 1080–1099. doi:10.1109/TCYB.2013.2279211

Yong W, Zixing C, Qingfu Z (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evolut Comput 55–66. doi:10.1109/TEVC.2010.2087271