Adaptive differential evolution with a new joint parameter adaptation method

Miguel Leon¹ · Ning Xiong¹

Published online: 20 July 2020
© The Author(s) 2020

Abstract

Differential evolution (DE) is a population-based metaheuristic algorithm that has been proved powerful in solving a wide range of real-parameter optimization tasks. However, the selection of the mutation strategy and control parameters in DE is problem dependent, and inappropriate specification of them will lead to poor performance of the algorithm such as slow convergence and early stagnation in a local optimum. This paper proposes a new method termed as Joint Adaptation of Parameters in DE (JAPDE). The key idea lies in dynamically updating the selection probabilities for a complete set of pairs of parameter generating functions based on feedback information acquired during the search by DE. Further, for mutation strategy adaptation, the Rank-Based Adaptation (RAM) method is utilized to facilitate the learning of multiple probability distributions, each of which corresponds to an interval of fitness ranks of individuals in the population. The coupling of RAM with JAPDE results in the new RAM-JAPDE algorithm that enables simultaneous adaptation of the selection probabilities for pairs of control parameters and mutation strategies in DE. The merit of RAM-JAPDE has been evaluated on the benchmark test suit proposed in CEC2014 in comparison to many well-known DE algorithms. The results of experiments demonstrate that the proposed RAM-JAPDE algorithm outperforms or is competitive to the other related DE variants that perform mutation strategy and control parameter adaptation, respectively.

Keywords Differential evolution · Evolutionary computation · Control parameter adaptation · Mutation strategy adaptation

1 Introduction

Differential evolution (DE) is a population-based algorithm that belongs to the Evolutionary Algorithms family (Storn and Price 1997; Xiong et al. 2015). In DE, the search is driven by calculating differences between vectors. DE is an attractive tool for real-parameter optimization due to its high performance and easy implementation. It has been successfully employed in many real-world applications, such as furnace optimized control system (Leon et al. 2015), process modelling for greenhouses (Perez-Gonzalez et al. 2018), enhancing the contrast and brightness of satellite images

(Suresh and Lal 2017), optimal control and design of electric systems (Bakare et al. 2007; Leon et al. 2016).

However, DE, as many other metaheuristic algorithms, have two main issues: a) Stagnation into local optima; b) slow convergence speed (Leon and Xiong 2019). If the algorithm exploits promising regions too much by having a greedy setup, it will more likely stagnate into local optima. On the other hand, if the algorithm, with the purpose of avoiding stagnation into local optima, excessively explores the search space, it will encounter the problem of slow convergence speed. For this reason, a proper trade-off between exploration and exploitation is very much needed in DE.

The exploration/exploitation behaviour of DE is largely dependent on its two control parameters: mutation factor (F) and crossover rate (CR) (Yildiz et al. 2015). Manually finding the best values of the parameters for real applications usually involves a time consuming and trial-and-error procedure. Moreover, in different stages of the search, we would need different parameters to suite the characteristics of the varying landscape. Inappropriate setting of the control parameters

Communicated by A. Di Nola.

✉ Miguel Leon
miguel.leonortiz@mdh.se

Ning Xiong
ning.xiong@mdh.se

¹ School of Innovation, Design and Engineering, Malardalen University, Box 833, Hogskoleplan 1, 721 Vasteras, Sweden

will lead to low convergence speed or stagnation into local optima during the search.

Adaptation of the control parameters during the execution of DE has received much attention for more than one decade. In the works presented in (Qin et al. 2009; Zhang and Sanderson 2009; Islam et al. 2012), the F and CR values for each individual of the population are generated by Gaussian and Cauchy probability density functions separately. The mean values of these probability functions are updated based on the successful F and CR values that succeeded in producing better trial solutions (than target vectors) in the previous generation. The SHADE algorithm (Tanabe and Fukunaga 2013) relies on the previous experience to calculate the weighted Lehmer means of successful F values and the weighted arithmetic mean of successful CR values, which are then utilized to replace an entity of the memories for F and CR generation, respectively.

This paper proposes a new method called Joint Adaptation of Parameters in DE (JAPDE), which emphasizes more reliable assessment of the combined effect of F and CR parameter generating functions. Like in SADE (Qin et al. 2009), JADE (Zhang and Sanderson 2009) and SHADE algorithms, JAPDE also uses two different probability functions to generate the F and CR values, respectively. What is novel is that JAPDE deals with pairs of mean values of the two probability functions instead of a single means with separation of the other. A pair of mean values is randomly selected from a pool of candidates for each target vector in running of JAPDE. We use a matrix to represent the selection probabilities for a complete set of pairs of the two mean values. This probability matrix is then dynamically updated via performance evaluation using feedback acquired during the search, such that those pairs that were more successful in producing better trial solutions in the last period will reinforce their probabilities to be selected and vice versa.

Further JAPDE has been combined with the Rank-Based Mutation Adaptation (RAM) method, which was proposed in our recent study (Leon and Xiong 2018). The reason for this combination is that, the different mutation strategies will also highly affect the exploration/exploitation behaviour of DE. The main idea of RAM is to maintain and adapt multiple probability distributions such that individuals of distinct fitness ranks can obtain different probabilities in selecting the mutation strategy. The combination of RAM and JAPDE gives rise to the RAM-JAPDE algorithm, which offers a new and probably more synergistic manner for simultaneously adapting the selection probabilities for mutation strategies and pairs of control parameters in DE. RAM-JAPDE has been tested on the set of benchmark problems from CEC'14 (Liang et al. 2013a), and it has been shown to outperform or be competitive to many state-of-the-art DE variants that also adapt their control parameters and mutation strategies in the optimization process.

The rest of the paper is organized as follows. First, a description of the conventional DE is given in Sect. 2. In Sect. 3, a review of well-known DE algorithms with adaptation in both control parameters and mutation strategies is given. In Sect. 4, all the details of the proposed RAM-JPADE algorithm are presented. The experiments and results are given in Sect. 5. Finally, the conclusion is given in Sect. 6.

2 Differential evolution

Differential Evolution (DE) was proposed by R. Storn and K. Price (Storn and Price 1995) in 1995. The population X is formed by ps (population size) individuals, in which the i -th individual is represented as $X_i = \{x_1, x_2, \dots, x_j, \dots, x_n\}$ where n is the dimension of the problem and $x_j \in \mathbb{R}$ is inside a range dependent on the specifications of the problem. The initial population of DE is randomly generated inside the search space. These individuals are transformed in evolution by three different operations: Mutation, Crossover and Selection. Conducting all these operations sequentially constitutes one cycle of DE algorithm, which leads to a new generation. These operations are described as follows:

MUTATION: The first step inside the DE cycle is called Mutation. In Mutation, a new population of mutated individuals, V , is generated by using differences of individuals of the population. There are different ways of calculating the i -th individual of V in generation g ($V_{i,g}$). Some common mutation strategies are described below. More variants of mutation strategies can be found in the literature (Leon and Xiong 2014; Zhang and Sanderson 2009; Leon and Xiong 2017).

– DE/rand/1

$$V_{i,g} = X_{r_1,g} + F \cdot (X_{r_2,g} - X_{r_3,g}) \quad (1)$$

– DE/best/1

$$V_{i,g} = X_{best,g} + F \cdot (X_{r_1,g} - X_{r_2,g}) \quad (2)$$

– DE/current-to-rand/1

$$V_{i,g} = X_{i,g} + F \cdot (X_{r_1,g} - X_{i,g} + X_{r_2,g} - X_{r_3,g}) \quad (3)$$

– DE/current-to-best/1

$$V_{i,g} = X_{i,g} + F \cdot (X_{best,g} - X_{i,g} + X_{r_1,g} - X_{r_2,g}) \quad (4)$$

where r_1, r_2 and r_3 represent random indexes between 1 and the maximum number of individuals in the population (ps), $X_{best,g}$ represents the best individual from the population at generation g and $F \in [0, 2]$. Additionally, the last position

in the name of the mutation strategies indicates the number of subtractions between random individuals from the population.

CROSSOVER: The second step is called Crossover. In Crossover, a mixture of the population, X , and the mutated population, V , is performed in order to generate the set of offspring, U . To obtain the j -th parameter of the i -th offspring, $U_{i,g}[j]$, in generation g , the following calculation is performed:

$$U_{i,g}[j] = \begin{cases} V_{i,g}[j] & \text{if } rand < CR \text{ or } j = j_{rand} \\ X_{i,g}[j] & \text{otherwise} \end{cases} \quad (5)$$

where $rand$ is a uniform random number in the range $[0, 1]$. CR , called crossover rate, is a value inside the range $[0, 1]$, which decides the probability of selecting values from the mutated individual. Additionally, j_{rand} is an index between 1 and n , to ensure that at least one value is taken from V_i .

SELECTION: The last step is called Selection. In selection, the fitness of the i -th offspring ($f(U_i)$) is compared against the fitness of the i -th individual from the population ($f(X_i)$) and the best one will become a member of the population for the next generation $g + 1$. This operation is decided by Eq. (6) for a minimization problem.

$$X_{i,g+1} = \begin{cases} U_{i,g} & \text{if } f(U_{i,g}) < f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \quad (6)$$

3 Related work

In this section, an overview of different DE algorithms that adapt the control parameters (Sect. 3.1) and DE algorithms that adapt different mutation strategies (Sect. 3.2) is presented.

3.1 Adaptation of parameters

As previously mentioned, finding the best combination of parameters in DE is an expensive task. According to (Eiben et al. 1999), there are three types of methods to control the parameters:

- **Deterministic parameter control:** A deterministic rule is applied to the different parameters in order to change them through the evolutionary process.
- **Adaptive parameter control:** Feedback from the search is used to adapt the different parameters to respond to changing properties in various stages of the search
- **Self-adaptive parameter control:** The parameters are embedded into the individuals, and thereby, the parameters will undergo different evolutionary operations.

The most common type is adaptive parameter control. The algorithms can be divided into two subcategories: individual parameter control and joint parameter control. Within the first subcategory, the first algorithm, SaDE (Qin et al. 2009), creates random F and CR values using two normal distributions, respectively. The novelty of this algorithm is that the mean of the normal distribution for CR , $meanCR$, is adapted. To do so, the successful CR values during the last learning period are used to calculate the new value of $meanCR$. A similar algorithm was proposed by Zhang et al. (Zhang and Sanderson 2009), stated as JADE. Differently from SaDE, both the Cauchy distribution for F and the normal distribution for CR are adapted. In this case, the successful F and CR values in the last generation are utilized to calculate their Lehmer and arithmetic means, respectively, for updating the mean values of both distributions. Similarly, in MDE_{pBX} (Islam et al. 2012) the Lehmer mean is replaced by a pow mean. A different algorithm was proposed by Tanabe et al. (Tanabe and Fukunaga 2013), stated as SHADE, which maintains memories of F and CR values calculated as weighted Lehmer mean and weighted arithmetic mean of successful F and CR values from the last generation. An improved version of SHADE was proposed in 2016 (Viktorin et al. 2016), in which a multi-chaotic framework is used to select the parents that will be used during the mutation phase.

Within the second subcategory there are not as many algorithms as in the first one. The well-known algorithm that keeps good combinations of F and CR values is EPSDE (Mallipeddi et al. 2011). EPSDE creates a pool with CR values from the range 0.1–0.9 and with F values from the range 0.4–0.9 in steps of 0.1. Random combinations are assigned to different individuals. If a combination successfully creates an offspring that is better than its parent, the same combination will be used for that offspring during the next generation. It will also be added to the pool of successful combinations. On the contrary, if the offspring is not better, then a new combination is assigned to that individual from either the pool of all combinations or the pool of successful combinations. A similar algorithm was proposed in (Wang et al. 2011), but with a reduced number of combinations in consideration. A different alternative was proposed in (Fan and Yan 2016), stated as ZEPDE, in which the total region of parameters is divided into 4 zones. Then, the combinations of values inside these zones are changed using a weighted mean of the successful combinations within each zone.

Even though adaptive parameter control within differential evolution algorithms is more common, there are some algorithms that belong to deterministic parameter control or self-adaptive parameter control. One good example of the former is sinDE (Draa et al. 2015), which uses sinusoidal functions to change F and CR values through the time without any feedback from the search. Similarly, a deterministic rule is proposed in (Sun et al. 2018), that also uses a sinu-

soidal function whereas for F and CR calculations dependent on individual performance ranking. More algorithms belongs to the latter. The first algorithm belonging to this category is jDE (Brest et al. 2006). In jDE, F and CR are embedded into the individuals. If a new offspring successfully outperforms its parent, then its associated parameters will survive as well, so that good combinations of parameters are available within the population. A different algorithm is proposed in (Teo 2006), stated as DESAP, in which an additional parameter, the population size, is adapted along with F and CR . The three parameters will undergo mutation with the individuals where they are embedded. Then, instead of crossover, a small perturbation is applied to the new values.

3.2 Adaptation of mutation strategies

There are different DE algorithms that use different mutation strategies. In SaDE (Qin et al. 2009), 4 different mutation strategies are used: DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin and DE/current-to-rand/1. Each of them will have a probability of being selected for mutation. After a learning period, these probabilities of the mutation strategies are adjusted according to their success rates. Similarly to SaDE, ZEPDE (Fan and Yan 2016) adapts the selection among 5 different mutation strategies: DE/rand/1, DE/current-to-best/2, DE/current-to-best/1, DE/best/2 and DE/rand/2, each of which will have a probability of being selected. After each generation, these probabilities are adjusted based on how much each created offspring improves with respect to the worst one in the population. Note that all this is performed after a certain number of generations (G_s) since at the beginning of the search, only DE/rand/1 is used.

Differently from SaDE and ZEPDE, CoDE (Wang et al. 2011) uses three different mutation strategies (DE/rand/1/bin, DE/rand/2/bin and DE/current-to-rand/1) at the same time to create three different offspring per parent. Then, the best offspring will compete with the parent to become a member of the population. An improved version of the algorithm was proposed in (Deng et al. 2019), which introduced a new operation before applying mutation. An even simpler adaptation of mutation strategies was proposed by Mallipeddi et al. (Mallipeddi et al. 2011), in which DE/best/2/bin, DE/rand/1/bin, DE/current-to-rand/1/bin are used as mutation strategies. They are then randomly selected for being assigned to the different individuals. If a mutation strategy successfully creates an offspring that will replace its parent, then the same mutation strategy will be used for that individual in the next generation. If the mutation strategy is not successful, a new mutation strategy will be randomly assigned.

More recently, SA-SHADE as an improved version of SHADE was proposed (Dawar and Ludwig 2018). SA-SHADE uses 5 different mutation strategies: DE/rand/1,

DE/rand/2, DE/best/2, DE/current-to-pbestWithArchive/ and DE/current-rand-to-pBest, all of them are described in (Dawar and Ludwig 2018). Similar as the parameter adaptation in SHADE, a memory of successful mutation strategies is maintained. In order to create an offspring, a random mutation strategy is selected from the memory. After each generation, the memory is updated with the mutation strategy that was successful for a highest number of times. In (Mohamed and Suganthan 2018), two mutations strategies are used as candidates: DE/rand/1 and the new triangular mutation strategy proposed to balance the global exploration. DE/rand/1 is selected with higher probability at the beginning of the search while both are used with the same probability at the end.

4 RAM-JAPDE: rank-based Adaptation of Mutation strategies with Joint Adaptation of Parameters in Differential Evolution

This paper proposes a new method that jointly adapts F and CR generations inside DE, which is referred to as Joint Adaptation of Parameters in DE (JAPDE). JAPDE is further integrated with the method of Rank-Based Adaptation of Mutation Strategies (RAM) (Leon and Xiong 2018), creating a new adaptive DE algorithm stated as RAM-JAPDE.

4.1 JAPDE: Joint Adaptation of Parameters in Differential Evolution

In most other adaptive DE algorithms, the parameters F and CR are considered separately when calculating the new centres of Normal/Cauchy distributions. However, F and CR are inherently related, e.g. for some functions, only small values of F work, if big values of CR are taken.

Like in some other adaptive DE algorithms, the F and CR values are generated in JAPDE using Cauchy and Normal distributions, respectively. Thus the i -th mutant vector F_i and crossover rate CR_i are created by following Eq. (7) and Eq. (8).

$$F_i = \text{Cauchy}(\text{mean}F_i, 0.05) \quad (7)$$

$$CR_i = \text{Normal}(\text{mean}CR_i, 0.05) \quad (8)$$

where $\text{Cauchy}(\text{mean}F_i, 0.05)$ is a random value generated by the Cauchy distribution with $\text{mean}F_i$ as its mean and std equal to 0.05. Similarly, $\text{Normal}(\text{mean}CR_i, 0.05)$ is a random value generated by the Normal distribution with $\text{mean}CR_i$ as its mean and std equal to 0.05. If F_i is greater than 1 then it is set to 1. On the contrary, F_i is generated again if it is smaller than 0. CR_i is set to 0 or 1 if it is outside the range $[0, 1]$. Differently from other variants of adaptive

DE, JAPDE maintains a probability distribution matrix (M) based on which to select pairs of $meanF_i$ and $meanCR_i$. The size of M is equal to 11×11 , in which the columns represent the different $meanF$ s (from 0 to 1 in the step size of 0.1) and the rows represent the different $meanCR$ s (from 0 to 1 in the step size of 0.1). The probability of selecting the pair of cr -th $meanCR$ and the f -th $meanF$ is given by $M_{cr,f}$.

The pseudocode of how a pair of $meanF_i$ and $meanCR_i$ is selected is given in Algorithm 1. In this algorithm, $zeros(ps,1)$ creates an array of ps zeros, $sumAll(M)$ will add all the values of M and $sumPerRow(M)$ will return an array, in which each value is the sum of the elements of an entire row of M .

Algorithm 1 function meanValuesSelection(M)

```

1: meanF = zeros(ps,1);
2: meanCR = zeros(ps,1);
3: for  $i = 1$  to  $ps$  do
4:   probabilityPerF = sumPerRow(M);
5:    $r = rand(0,1)$ ;
6:   cumulativeP = probabilityPerF(0);
7:    $f = 0$ ;
8:   while cumulativeP <  $r$  do
9:      $f++$ ;
10:    cumulativeP = cumulativeP + probabilityPerF( $f$ )
11:  end while
12:  totalProbabilityF = probabilityPerF( $f$ )
13:   $r = rand(0,totalProbabilityF)$ ;
14:   $cr = 0$ ;
15:  cumulativeP =  $M_{f,cr}$ ;
16:  while cumulativeP <  $r$  do
17:     $cr++$ ;
18:    cumulativeP = cumulativeP +  $M_{f,cr}$ 
19:  end while
20:   $meanF(i) = f/10$ ;
21:   $meanCR(i) = cr/10$ ;
22: end for
23: return meanF, meanCR;
```

M is updated after a certain number of generations (LP_{PA} generations), also called learning period, following Eq. (9) and Eq. (10).

$$M_{cr,f} = (1 - E_{PA}) \cdot M_{cr,f} + E_{PA} \cdot \Delta M_{cr,f} \quad (9)$$

$$\Delta M_{cr,f} = \left(\frac{\left(\frac{successPC_{cr,f} \cdot EP_f}{triesPC_{cr,f}} \right)}{\sum_{cr=0}^{10} \sum_{f=0}^{10} \left(\frac{successPC_{cr,f} \cdot EP_f}{triesPC_{cr,f}} \right)} \right) \quad (10)$$

where E_{PA} is the evaporation rate used to update the parameters, $successPC_{cr,f}$ is the number of times in which the combination f - cr (f - index representing the closest candidate to the used F , cr - index representing the closest candidate to the used CR) was successful, $triesPC_{cr,f}$ is the number of times in which the combination f - cr was tried and EP_f is a value from the exploration plane. EP_f is

calculated as indicated in Eq. (11). If f is equal to 0, then 0.01 is used instead.

$$EP_f = \frac{f}{10} \cdot e^{-\left(\frac{NFES}{NFES_{MAX}}\right)^3} \quad (11)$$

where $NFES$ is the current number of evaluations and $NFES_{MAX}$ is the maximum number of evaluations. The reason for using the evaporation rate and the exploration plane is to avoid fast convergence to the pairs of $meanF$ and $meanCR$ that can work well in early stages of the search but they will become inappropriate in later stages.

4.2 Adaptation of mutation strategies

JAPDE is combined with RAM, a method that adapts the selection between different mutation strategies. As previously mentioned, there are many different mutation strategies with various exploration-exploitation behaviour. The highly exploitative mutation strategies include DE/best/1 and DE/current-to-best, which utilize the best individual in the population in creating a new mutant vector. The highly exploratory strategies include DE/rand/1 and DE/current-to-rand/1, which completely rely on randomly selected individuals from the population in the mutation operation. Further by considering the structure of mutation strategies, DE/best/1 can be viewed as highly exploitative as opposed to DE/rand/1, and DE/current-to-best as highly exploitative as opposed to DE/current-to-rand/1. A compromise between two opposite mutation strategies can be implemented by two new strategies: DE/ p best/1 (Eq. (12)) and DE/current-to- p best/1 (Eq. (13)), which can be used to produce an effect that lies in between highly exploratory and exploitative behaviours. How much these two new mutation strategies are going to explore or exploit the search space is determined by the parameter p . If p is set to a very high percentage, these two parameterized strategies will behave closely to the exploratory mutation strategies, otherwise they will function more similarly to exploitative mutation strategies. The relation between the 6 different mutation strategies can be found in Fig. 1.

$$V_{i,g} = X_{pBest,g} + F_i \cdot (X_{r1,g} - X_{r2,g}) \quad (12)$$

$$V_{i,g} = X_{i,g} + F_i \cdot (X_{pBest,g} - X_{i,g} + X_{r1,g} - X_{r2,g}) \quad (13)$$

where $X_{pBest,g}$ is a random individual selected from the p best individuals, with p being a percentage value of the population size. $X_{r1,g}$ and $X_{r2,g}$ are randomly selected individuals from the population.

In RAM-JAPDE, both strategies, DE/ p best/1 and DE/current-to- p best/1, are used. Firstly, the parameter p is decided by the state of the search. A high value of p is enforced at the beginning of the search to enforce the algo-

p		
1%	⇒ 20% ⇒ 50% ⇒ 80% ⇒	100%
DE/best/1	DE/pBest/1	DE/rand/1
DE/current-to-best/1	DE/current-to-pBest/1	DE/current-to-rand/1

Fig. 1 Relation between p and 6 different mutation strategies. These percentages correspond to a population size equal to 100

rithm to more explore the search space, while a small value of p is used at the end of the search process to enable the algorithm to converge faster. How p changes during the search process is given in Eq. (14).

$$p = \max \left(1 - \frac{NFES}{NFES_{MAX}}, \frac{1}{ps} \right) \quad (14)$$

where ps is the population size, $NFES$ is the current number of evaluations and $NFES_{MAX}$ is the maximum number of evaluations. The evaluation maximum is used to ensure that at least one individual from the population is always taken.

Secondly, in order to decide which strategy is used to create the different mutant vectors, a method that we previously proposed, called Rank-based Mutation Adaptation (RAM), is used (Leon and Xiong 2018). In RAM, the individuals from the population are first ordered according to their fitness values, and then the population is divided into different groups. The number of groups (gr) is a parameter to be decided by the user. The groups size (GS) is calculated as stated in Eq. (15).

$$GS = \frac{ps}{gr} \text{ where } GS \in \mathbb{Z}^+ \quad (15)$$

where ps is the population size.

The number of probabilities per group is set depending on the number of mutation strategies used by the algorithm, which is two in this algorithm ($pBest/1$ and current-to- $pBest/1$). The total probability profile is stated as P , where $P_{k,l}$ stands for the probability of selecting the l -th mutation strategy for individuals of the k -th group. The pseudocode showing how the mutation strategies are selected is given in Algorithm 2. In Algorithm 2, $ceil(i/GS)$ refers to the group id to which the i -th individual belongs, $zeros(ps, 1)$ creates a vector of zeros with size ps , and $rand(0, a)$ returns a random value in the range $[0, a]$.

The different probabilities are updated according to its performance during a period of time, called learning period (LP_{RAM}). Hence, the new probability of using the l -th mutation strategy in the k -th group ($P_{k,l}$) is calculated as

$$P_{k,l} = (1 - E_{RAM}) \cdot P_{k,l} + E_{RAM} \cdot \Delta P_{k,l} \quad (16)$$

$$\Delta P_{k,l} = \left(\frac{\left(\frac{successMS_{k,l}}{triesMS_{k,l}} \right)}{\sum_{l=1}^2 \left(\frac{successMS_{k,l}}{triesMS_{k,l}} \right)} \right) \quad (17)$$

Algorithm 2 function selectMutationStrategy(GS, P)

```

1: mutationToUse = zeros( $ps, 1$ );
2: for  $i = 1$  to  $ps$  do
3:   cumulativeP = 0;
4:    $k = ceil(i/GS)$ ;
5:    $r = rand(0, 1)$ 
6:   for  $l = 1$  to 2 do
7:     cumulativeP = cumulativeP +  $P_{k,l}$ ;
8:     if  $r \leq cumulativeP$  && mutationToUse( $i$ ) == 0 then
9:       mutationToUse( $i$ ) =  $j$ ;
10:    end if
11:  end for
12: end for
13: return mutationToUse;
```

where E_{RAM} is the evaporation rate. This is used in order to avoid a big change on the probabilities. $successMS_{k,l}$ and $triesMS_{k,l}$ are, respectively, the number of times in which the l -th mutation strategy was successfully used for the k -th group in creating an offspring better than the parent, and the number of times that the same mutation strategy was used for the same group. A complete pseudocode of RAM-JAPDE is provided in Algorithm 3.

5 Experiments and results

In this section, the strength of RAM-JAPDE is tested. First, in Sect. 5.1, the experimental settings are described. Second, a study of the different parameters and parts of RAM-JAPDE is performed in Sect. 5.2. In Sect. 5.3, the strength of the JAPDE method is studied. Forth, in Sect. 5.4, the performance of RAM-JAPDE is compared with other adaptive DE algorithms. Fifth, the convergence speed of RAM-JAPDE is compared with other DE algorithms in Sect. 5.5. Sixth, the performance of RAM-JAPDE in problems of different dimensions is given in Sect. 5.6. Seventh, a discussion on the evolution of M is carried out in Sect. 5.7. Eighth, a comparison of the F and CR values used by RAM-JAPDE and SHADE is performed in Sect. 5.8. Finally, RAM-JAPDE is compared with L-SHADE, the winner of the CEC2014 competition, in Sect. 5.9.

5.1 Experimental settings and comparative measures

In this paper, the benchmark proposed in CEC'14 conference is used for comparisons. This benchmark is composed of 30 functions, which are grouped into four categories: Unimodal functions (F1–F3), multimodal functions (F4–F16), Hybrid Functions (F17–F22) and Composition functions (F23–F30). For each function, all the algorithms compared in this paper have been tested 50 independent times with a maximum number of evaluations of $10000 \times dimension$. If an algorithm

Algorithm 3 RAM-JAPDE

```

1: Initialize  $M$  with  $1/121$ ;
2: Initialize  $P$  with  $1/2$ ;
3: Calculate GS as in Eq. (15);
4: Initialize population  $X$ ;
5:  $NFES = 0$ ,  $G = 0$ ;
6:
7: //Main loop
8: while  $NFES < NFES_{MAX}$  do
9:   Calculate  $p$  as in Eq. (14);
10:   $mutationToUse = selectMutationStrategies(GS, P)$ ;
11:   $[meanF, meanCR] = meanValuesSelection(M)$ ;
12:  for  $i = 1$  to  $ps$  do
13:     $F_i = Cauchy(meanF(i), 0.05)$ 
14:     $CR_i = Normal(meanCR(i), 0.05)$ 
15:
16:    //Mutation
17:    if  $mutationToUse(i) == 1$  then
18:       $V_i = X_{pBest} + F_i \cdot (X_{r1} - X_{r2})$ ;
19:    else
20:       $V_i = X_i + F_i \cdot (X_i - X_{pBest} X_{r1} - X_{r2})$ 
21:    end if
22:
23:    //Crossover
24:    for  $j = 1$  to  $dimension$  do
25:      if  $rand(0,1) < CR_i$  or  $j == j_{rand}$  then
26:         $U_i[j] = V_i[j]$ 
27:      else
28:         $U_i[j] = X_i[j]$ ;
29:      end if
30:    end for
31:
32:    //Selection and counting success and trials
33:     $f = round(F_i * 10)$ ,  $cr = round(CR_i * 10)$ ;
34:     $k = ceil(i / GS)$ ,  $l = mutationToUse(i)$ ;
35:    if  $f(U_i) < f(X_i)$  then
36:       $X_i = U_i$ ;
37:       $successPC_{cr, f}++$ ;
38:       $successMS_{k, l}++$ ;
39:    end if
40:     $triesPC_{cr, f}++$ ;
41:     $triesMS_{k, l}++$ ;
42:
43:  end for
44:   $G++$ ;
45:   $NFES = NFES + ps$ ;
46:
47:  //Update M and P
48:  if  $G \% LP == 0$  then
49:    Update M as in Eq. (9);
50:     $successMS = triesMS = 0$ ;
51:    Update P as in Eq. (16);
52:     $successPC = triesPC = 0$ ;
53:  end if
54: end while

```

obtained an error below $1.00E-08$, the optimal solution was considered to be found.

Additionally, in order to make a fair comparison between the different algorithms, two measures and one statistical test have been used:

- F.A.R.: the average ranking according to Friedman statistical test.
- S.E.R.: the summation of the relative errors with respect to the worst per function.
- Wilcoxon's signed rank statistical test ($\alpha < 0.05$).

F.A.R. is a N to N comparison, in which a smaller value represents a better algorithm. In order to calculate the F.A.R. values, the following two steps have to be followed: 1) Calculate the rank of each algorithm on each separate problem and 2) calculate the average rank value.

Similar to F.A.R., S.E.R. is a N to N comparison in which a smaller value represents a better algorithm. When calculating the S.E.R. value, two steps have to be followed: 1) The performance of each algorithm on each problem is divided by the performance of the worst performing algorithm on the exact same problem and 2) The summation of the values across all functions is performed.

Wilcoxon's signed rank statistical test is a 1 to 1 statistical test. The first step is to calculate the differences between the two compared algorithms. Then, these differences are ranked from smaller to larger. The ranks are equal to points that the algorithms will receive if they are better. In order to conclude that one algorithm is better than the other, the p value has to be smaller than the significance values ($\alpha = 0.05$ in our case), then the differences between the algorithms are statistically relevant. We have used the singrank function in MATLAB.

More information can be found in (Derrac et al. 2011)

5.2 Parameters study of RAM-JAPDE

In order to create RAM-JAPDE, 4 different parts have been combined together. Firstly, a matrix containing selection probabilities of all combinations of parameters is introduced. Secondly, RAM method is used to adapt the selection between two different mutation strategies. Thirdly, an exploration plane is added to the probability matrix. Lastly, a linear reduction of the parameter p is used. In this section, a study of the different components is performed using $PS = 100$ and $gr = 10$.

There are four constants that will affect the adaptation of the probability matrix and the usage of RAM: Learning Period of RAM (LP_{RAM}), learning period of parameters adaptation (LP_{PA}), evaporation rate of RAM (E_{RAM}) and evaporation rate of parameter adaptation (E_{PA}). A comparison between different combinations of parameters, on the benchmark proposed in CEC'13 conference (Liang et al. 2013b), is performed (Fig. 2). In order to make an easier comparison, we will assume that $E_{RAM} = E_{PA}$ (E) and $LP_{RAM} = LP_{PA}$ (LP). As it is expected, if we have an aggressive evaporation with a small learning period, the matrix will change too much and it will change too quickly from one combination of parameters to another. On the con-

LP \ E	0.01	0.05	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
1	0.724	0.839	0.840	0.870	0.876	0.885	0.894	0.914	0.927	1.000	0.958
10	0.714	0.692	0.687	0.806	0.845	0.823	0.854	0.887	0.862	0.896	0.895
20	0.736	0.644	0.664	0.765	0.757	0.853	0.802	0.839	0.868	0.870	0.891
30	0.806	0.625	0.614	0.674	0.698	0.809	0.812	0.837	0.817	0.871	0.862
40	0.756	0.659	0.642	0.652	0.726	0.696	0.747	0.785	0.807	0.847	0.864
50	0.773	0.693	0.663	0.640	0.675	0.695	0.752	0.773	0.795	0.792	0.805
60	0.806	0.675	0.626	0.637	0.624	0.707	0.658	0.734	0.775	0.793	0.838
70	0.822	0.668	0.657	0.650	0.624	0.651	0.703	0.734	0.738	0.738	0.787
80	0.852	0.740	0.693	0.597	0.625	0.665	0.688	0.711	0.723	0.757	0.767
90	0.828	0.729	0.685	0.643	0.627	0.622	0.637	0.679	0.743	0.672	0.752
100	0.823	0.759	0.653	0.604	0.615	0.608	0.663	0.667	0.738	0.715	0.683
110	0.783	0.747	0.669	0.613	0.652	0.628	0.607	0.675	0.706	0.684	0.724
120	0.856	0.741	0.706	0.610	0.662	0.610	0.695	0.638	0.683	0.694	0.711
130	0.870	0.743	0.729	0.664	0.611	0.637	0.606	0.613	0.667	0.675	0.712
140	0.852	0.686	0.677	0.675	0.668	0.621	0.600	0.607	0.660	0.679	0.655
150	0.865	0.771	0.727	0.642	0.620	0.625	0.633	0.632	0.631	0.691	0.721

Fig. 2 Relation between different configurations of Evaporation rate (E) and Learning Period (LP) on the CEC'13 benchmark problems (Liang et al. 2013b). The values represent the normalized average value between F.A.R. (average ranking according to Friedman statistical test) and S.R.E. (summation of the relative errors with respect to the worst). The best result is represented with white, while the worst is represented with dark blue

trary, if we set a small evaporation rate with a long learning period, it will shift too slowly to a new combination of parameters. According to the results, the best combination is $E = 0.2$ and $LP = 80$.

As was mentioned above, RAM is used to adapt the selection between DE/current-to-pBest/1 (DE/ctpb/1) and DE/pBest/1. A comparison between RAM-JAPDE and JAPDE using only one of the two mutation strategies is performed (Fig. 3). In order to perform this comparison, the previously found parameters are used ($LP = 80$ and $E = 0.2$). From the figure, it can be observed that using RAM to adapt the selection between the two different mutation strategies did not obtained the best performance on every function. However, its results on each function is closer to the mutation strategy with better results. The good performance of RAM-JAPDE is also verified by the results shown in Table 1, demonstrating that using RAM provides a good coupling between the two mutation strategies.

Additionally, two extra parts are included into the algorithm: the exploration plane and the linear reduction of p . RAM-JAPDE has been tested without the exploration plane (RAM-JAPDE no plane) and with a random p instead of the linear reduction (RAM-JAPDE random p). The Wilcoxon's statistical test, together with F.A.R. and S.R.E, shows that adding the exploration plane and having a linear reduction of p provide a better performance due to a good exploration/exploitation balance (Table 1).

5.3 Comparison of JAPDE with other parameter adaptation methods

In order to evaluate the strength of the proposed joint parameter adaptation method, JAPDE is compared with 4

Func	JAPDE DE/ctpb/1	RAM-JAPDE	JAPDE DE/pBest/1
F1	3.59E+03	5.75E+03	1.15E+04
F2	0.00E+00	0.00E+00	0.00E+00
F3	0.00E+00	0.00E+00	0.00E+00
F4	9.56E-03	0.00E+00	3.01E-05
F5	2.03E+01	2.03E+01	2.03E+01
F6	1.02E+01	6.78E-01	9.08E-01
F7	0.00E+00	0.00E+00	2.47E-04
F8	0.00E+00	0.00E+00	0.00E+00
F9	2.97E+01	2.34E+01	3.36E+01
F10	2.95E-03	7.63E-03	3.96E-02
F11	1.66E+03	1.40E+03	1.40E+03
F12	3.19E-01	2.89E-01	1.85E-01
F13	2.03E-01	1.93E-01	1.56E-01
F14	1.97E-01	2.10E-01	2.38E-01
F15	3.94E+00	2.23E+00	2.64E+00
F16	9.60E+00	8.95E+00	8.58E+00
F17	3.57E+02	3.73E+02	4.79E+02
F18	1.52E+01	1.64E+01	1.23E+01
F19	4.69E+00	3.37E+00	2.48E+00
F20	1.09E+01	8.48E+00	7.86E+00
F21	2.05E+02	1.52E+02	1.53E+02
F22	1.25E+02	1.13E+02	1.17E+02
F23	3.15E+02	3.15E+02	3.15E+02
F24	2.24E+02	2.24E+02	2.24E+02
F25	2.02E+02	2.03E+02	2.03E+02
F26	1.00E+02	1.00E+02	1.00E+02
F27	3.55E+02	3.56E+02	3.62E+02
F28	8.30E+02	8.12E+02	8.11E+02
F29	7.22E+02	7.02E+02	7.23E+02
F30	7.47E+02	8.82E+02	1.05E+03

Fig. 3 Comparison between using Rank-based adaptation of mutation strategies (RAM) and using DE/current-to-pBest/1 (DE/ctpb/1) or DE/pBest/1 independently. The best results per function is represented in white, while the worst result per function is represented in dark blue

other adaptation methods. They are taken from SaDE (Qin et al. 2009), jDE (Brest et al. 2006), JADE (Zhang and Sanderson 2009) and SHADE (Tanabe and Fukunaga 2013). These algorithms were evaluated under the same conditions in dimension 10, 30, 50 and 100. The conditions are the same as given in the experimental settings. Additionally, DE/rand/1/bin was used as the basic mutation strategy on which the algorithms were tested. In order to stress that only the methods of parameter adaptation of the algorithms are tested, a suffix “-p” has been added to the name of the algorithms. Additional parameters that are concerned are listed below:

- JAPDE: $PS = 100$, $LP_{PA} = 80$ and $E_{PA} = 0.20$.
- SaDE-p: $PS = 100$ and $LP = 20$;
- JADE-p: $PS = 100$ and $c = 0.1$.
- jDE: $PS = 100$,
- SHADE-p: $PS = 100$ and $H = 100$.

Table 1 Comparison of RAM-JAPDE and its different parts using the three measures: (1) Wilcoxon's statistical test ($\alpha = 0.05$) between RAM-JAPDE and its different parts, (2) the average ranking according to Friedman statistical test (F.A.R.) and (3) the summation of the relative errors with respect to the worst (S.R.E.)

Algorithms	Wilcoxon's stat. Test		Other measures	
	<i>p</i> value	stat. Dif?	F.A.R.	S.R.E.
RAM-JAPDE	–	–	2.28	16.09
JAPDE DE/ctpb/1	1.34E–01	No	3.02	18.17
JAPDE DE/pBest/1	3.67E–01	No	3.52	16.79
RAM-JAPDE no plane	7.45E–04	Yes	3.22	18.69
RAM-JAPDE random p	4.79E–04	Yes	3.97	24.48

The results (Table 2) show that JAPDE is stronger in all the dimensions than its counterparts, except jDE, which is statistically similar to JAPDE in dimensions 30, 50 and 100. Besides, JAPDE and SHADE are statistically similar in dimension 30.

The reason of stronger performance of JAPDE lies in its ability to adapt the control parameters by treating *F* and *CR* in a combined manner, while the other adaption methods (except jDE) adjust the two parameters separately. In jDE, *F* and *CR* are evolved at the same time in the running of the algorithm. That is why jDE obtained competitive results to JAPDE in the experiments.

5.4 Comparison of RAM-JAPDE with other adaptive Differential Evolution Algorithms

In order to evaluate the strength of RAM-JAPDE, two different comparisons in dimension 30 have been performed.

Firstly, RAM-JAPDE have been compared with 6 different DE algorithms that also adapt the selection between different mutation strategies: SaDE (Qin et al. 2009), CoDE (Wang et al. 2011), EPSDE (Mallipeddi et al. 2011), ZEPDE (Fan and Yan 2016), SA-SHADE (Dawar and Ludwig 2018) and EFADE (Deng et al. 2019). The parameters used in all the algorithms are the same as given in their respective papers:

- SaDE: $PS = 100$ and $LP = 20$.
- CoDE: $PS = 30$
- EPSDE: $PS = 50$
- ZEPDE: $PS = 100$, $G_{max} = 3000$, $G_s = 0.175 \times G_{max}$ and $G_b = 0.35 \times G_{max}$
- SA-SHADE: $PS = 100$ and $H = 100$.
- EFADE: $PS = 50$
- RAM-JAPDE: $PS = 100$, $gr = 10$, $LP_{RAM} = LP_{PA} = 80$ and $E_{RAM} = E_{PA} = 0.2$

In Table 3, a comparison between RAM-JAPDE and the previously related algorithms is performed. It can be seen that RAM-JAPDE obtained the best results in half of all the functions. Additionally, Wilcoxon's statistical test has been

performed, showing that RAM-JAPDE is statistically better than its counterparts. Moreover, RAM-JAPDE obtained the best F.A.R. and S.E.R. scores. It is good to mention that CoDE, EPSDE and ZEPDE also adapt *F* and *CR* in a joint manner and that RAM-JAPDE outperforms all of them.

Secondly, RAM-JAPDE have been compared with other well-known DE algorithms: ODE (Rahnamayan et al. 2006), JADE (Zhang and Sanderson 2009), SHADE (Tanabe and Fukunaga 2013), jDE (Brest et al. 2006), IDDE (Sun et al. 2018), MC-SHADE (Viktorin et al. 2016), η -CODE (Deng et al. 2019) and sinDE (Draa et al. 2015). The parameters used in the algorithms are the same as described in their respective papers and are listed below:

- ODE: $PS = 100$, $F = 0.5$, $CR = 0.9$ and $J_r = 0.3$
- JADE: $PS = 100$, $A = 100$, $c = 0.1$ and $p = 0.1$.
- SHADE: $PS = 100$ and $H = 100$.
- jDE: $PS = 100$, $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$ and $F_u = 0.9$.
- IDDE: The results were taken from (Sun et al. 2018).
- MC-SHADE: The results were taken from (Viktorin et al. 2016).
- η -CoDE: The results were taken from (Deng et al. 2019).
- sinDE: $PS = 40$, $freq = 0.25$ and configuration 2.
- RAM-JAPDE: $PS = 100$, $gr = 10$, $LP_{RAM} = LP_{PA} = 80$ and $E_{RAM} = E_{PA} = 0.2$

The results of this comparison are presented in Table 4. It can be observed from the table that RAM-JAPDE is statistically better than its counter parts, except η -CODE, which is statistically similar. Additionally, if F.A.R. and S.E.R. scores are considered, RAM-JAPDE obtained the best results among all the compared algorithms.

5.5 Convergence speed analysis

In this subsection, the convergence speed of RAM-JAPDE is analysed and compared with algorithms that either jointly adapt the parameters of DE, as CoDE, EPSDE, ZEPDE or are well-known adaptive DE algorithms such as SaDE, JADE, jDE and SHADE.

Table 2 Comparison between JAPDE and four adaptive Differential Evolution algorithms in dimensions 10, 30, 50 and 100 using three measures: (1) Wilcoxon's statistical tests ($\alpha = 0.05$) of JAPDE vs. other algorithms, (2) the average ranking according to Friedman statistical

test (F.A.R.) and (3) the summation of the relative errors with respect to the worst (S.R.E.). *p* values in boldface means that there are statistical differences in favour of JAPDE

	Dimension 10			Dimension 30			Dimension 50			Dimension 100		
	<i>p</i> value	F.A.R.	S.R.E.	<i>p</i> value	F.A.R.	S.R.E.	<i>p</i> value	F.A.R.	S.R.E.	<i>p</i> value	F.A.R.	S.R.E.
JAPDE	–	1.57	11.32	–	2.05	12.27	–	1.95	12.69	–	2.07	12.81
SaDE-p	7.45E–08	3.38	16.46	8.28E–04	3.27	19.25	1.19E–05	3.37	18.41	3.50E–03	3.23	18.6
JADE-p	9.43E–07	4.43	23.58	1.86E–08	4.65	24.32	5.72E–07	4.55	24.38	9.04E–06	4.27	24.41
jDE	4.09E–05	3.38	17.39	3.67E–01	2.95	17.02	6.70E–01	2.7	18.02	3.81E–01	2.8	17.61
SHADE-p	1.20E–02	2.23	14.01	9.21E–01	2.08	13.9	2.80E–03	2.43	16.06	1.21E–02	2.63	16.52

Table 3 Mean error comparison between RAM-JAPDE, CoDE, EPSDE, ZEPDE, SaDE, SA-SHADE and EFADE in dimension 30. Additionally, the results of Wilcoxon's statistical test ($\alpha = 0.05$), the average ranking according to Friedman statistical test (F.A.R.) and the

summation of the relative errors with respect to the worst (S.R.E.) are given at the bottom of the table. *p* values in boldface means that there are statistical differences in favour of RAM-JAPDE

Func.	RAM-JAPDE	CoDE	EPSDE	ZEPDE	SaDE	SA-SHADE	EFADE
F1	5.75E+03	9.68E+04	4.18E+06	1.31E+06	3.70E+05	3.91E+03	1.29E+04
F2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F3	0.00E+00	0.00E+00	0.00E+00	3.25E–08	0.00E+00	0.00E+00	0.00E+00
F4	0.00E+00	2.57E–01	5.71E–01	8.02E+00	2.83E+00	0.00E+00	1.45E–02
F5	2.03E+01	2.01E+01	2.09E+01	2.08E+01	2.09E+01	2.03E+01	2.05E+01
F6	6.78E–01	3.37E+00	7.39E–01	1.12E+00	1.97E–01	9.84E+00	2.94E+00
F7	0.00E+00	1.48E–03	0.00E+00	0.00E+00	2.87E–03	1.40E–03	0.00E+00
F8	0.00E+00	1.53E+01	4.69E+01	1.51E+01	0.00E+00	9.95E–02	0.00E+00
F9	2.34E+01	7.40E+01	1.51E+02	3.60E+01	9.24E+01	2.96E+01	4.55E+01
F10	7.63E–03	1.93E+02	1.94E+03	1.44E+02	1.79E+02	6.98E–04	3.37E–01
F11	1.40E+03	3.27E+03	6.20E+03	3.63E+03	5.38E+03	1.76E+03	3.29E+03
F12	2.89E–01	1.28E–01	1.94E+00	1.35E+00	1.61E+00	2.92E–01	6.78E–01
F13	1.93E–01	4.39E–01	2.72E–01	2.04E–01	2.63E–01	2.34E–01	3.53E–01
F14	2.10E–01	3.41E–01	2.60E–01	2.39E–01	2.47E–01	2.18E–01	2.56E–01
F15	2.23E+00	1.09E+01	1.37E+01	6.29E+00	9.70E+00	3.56E+00	9.73E+00
F16	8.95E+00	1.14E+01	1.22E+01	1.15E+01	1.20E+01	9.51E+00	1.08E+01
F17	3.73E+02	5.30E+03	3.56E+05	6.18E+04	1.68E+03	9.92E+02	7.52E+02
F18	1.64E+01	6.67E+01	4.42E+02	2.81E+02	6.54E+01	7.60E+01	1.23E+01
F19	3.37E+00	4.77E+00	5.09E+00	5.15E+00	5.00E+00	4.43E+00	3.90E+00
F20	8.48E+00	3.70E+01	5.88E+01	6.64E+01	5.13E+01	2.91E+01	1.08E+01
F21	1.52E+02	1.23E+03	8.50E+03	5.50E+03	3.96E+02	3.58E+02	1.42E+02
F22	1.13E+02	3.85E+02	1.38E+02	5.62E+01	1.24E+02	1.32E+02	1.19E+02
F23	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02
F24	2.24E+02	2.27E+02	2.26E+02	2.24E+02	2.24E+02	2.24E+02	2.23E+02
F25	2.03E+02	2.04E+02	2.05E+02	2.04E+02	2.05E+02	2.04E+02	2.03E+02
F26	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.00E+02
F27	3.56E+02	3.79E+02	3.15E+02	3.02E+02	3.39E+02	3.67E+02	3.97E+02
F28	8.12E+02	8.57E+02	8.17E+02	8.08E+02	8.40E+02	8.03E+02	8.22E+02

Table 3 continued

Func.	RAM-JAPDE	CoDE	EPSDE	ZEPDE	SaDE	SA-SHADE	EFADE
F29	7.02E+02	8.80E+02	2.41E+03	1.98E+03	9.21E+02	7.25E+02	7.22E+02
F30	8.82E+02	1.57E+03	1.73E+03	1.66E+03	9.57E+02	1.62E+03	8.32E+02
Wil. <i>p</i> value	–	2.83E–07	4.08E–06	5.36E–04	1.11E–05	6.64E–04	6.70E–03
F.A.R.	1.95	4.85	5.73	4.42	4.58	3.13	3.33
S.R.E.	11.27	16.61	23.49	18.97	16.89	15.23	15.60

In order to be able to study the convergence speed of RAM-JAPDE and compare it with other algorithms, some convergence graphs are presented in Fig. 4. In this figure, the performance of the algorithms on 12 functions are presented. These functions can be divided into 4 groups:

- Unimodal functions: F1, F2 and F3.
- Multimodal functions: F4, F6 and F9.
- Hybrid functions: F17, F18 and F21.
- Composite functions: F24, F29 and F30.

It can be observed that on unimodal functions RAM-JAPDE is not the most exploitative algorithm due to the exploratory behaviour that RAM-JAPDE has at the beginning of the search. In F1, it can be observed that although RAM-JAPDE did not converge fast at the beginning, it behaved better than other algorithms when it started to change its behaviour to an exploitative one. The same result can be observed in multimodal functions, specially in F4 and F6.

A different observation can be made in hybrid functions. It seems that the algorithms can be divided into three groups. The first group is actually a single algorithm, JADE, that gets stuck in local optima faster than other algorithms due to the usage of DE/ctpb/l with a small *p*. The second group is formed by jDE and ZEPDE. Since they use DE/rand/l, they have an exploratory behaviour. Then, when ZEPDE turns to adapt mutation strategies, we can observe how it could improve its performance. The third group is formed by all the other algorithms where differences between them depend on how much they explore/exploit at the beginning. It can be observed from this group that, at the beginning of the search, RAM-JAPDE was the most exploratory algorithm, but when the search process advanced, it reached better solutions than its counterparts.

A similar situation happened in composite functions, with the difference that only two groups existed. The first group is formed by jDE and ZEPDE. And the second group is formed by all the other algorithms. In these functions, the difference between the algorithms is smaller.

5.6 Dimensionality study

In this section, we will study the performance of RAM-JAPDE when dealing with problems of different dimensions. In order to perform the study, we have compared RAM-JAPDE against CoDE, EPSDE, ZEPDE, SaDE, SA-SHADE, EFADE, JADE, jDE and SHADE on the CEC2014 benchmark functions but with lower dimensionality (dimension 10) and higher dimensionalities (dimension 50 and 100).

Firstly, a summary of the results obtained by the mentioned algorithms, for the problems in dimension 10, is presented (Table 5). It can be observed that when the dimensionality of the problem is reduced, RAM-JAPDE outperformed all its counterparts.

Secondly, a summary of the results obtained by the different algorithms in problems of dimension 50 (Table 6) and dimension 100 (Table 7) are presented. It can be observed that RAM-JAPDE outperformed all its counterparts, except CoDE in dimension 50 and SHADE in dimension 50 and 100, in which they are statistically similar. These results indicate that RAM-JAPDE can perform well in higher dimensions.

The superiority of RAM-JAPDE to its counterparts is owing to the strength of the two adaptation methods: RAM and JAPDE are used together. In RAM, the selection probabilities of mutation strategies are adjusted specifically for individuals of different fitness ranking. In JAPDE, we achieve a joint parameter adaptation by considering a complete set of pairs of *F* and *CR* values. Such advantages for both parameter and mutation adaptation are not available with other adaptive DE algorithms

5.7 Discussion of the adaptive probability distribution matrix (*M*)

Different parameters have been defined within RAM-JAPDE in the previous sections that affect the probability matrix *M*. In Fig. 5, the probabilities of creating a pair of *f-cr* values inside the different groups, using *meanF* and *meanCR* equal to 0.5, are shown. It can be observed that though we are using mean values of 0.5, *f-cr* combinations from different groups can be created as well, with smaller probabilities. Moreover, combinations, in which *F* is equal to 1, will have

Table 4 Mean error comparison between RAM-JAPDE, ODE, JADE, SHADE, jDE, IDDE, MC-SHADE, SA-SHADE, η_{CODE} and SinDE in dimension 30. Additionally, the results of Wilcoxon's statistical test ($\alpha = 0.05$), the average ranking according to Friedman statistical test (F.A.R.) and the summation of the relative errors with respect to the worst (S.R.E.) are given at the bottom of the table. p values in boldface means that there are statistical differences in favour of RAM-JAPDE

Func.	RAM-JAPDE	ODE	JADE	SHADE	jDE	IDDE	MC-SHADE	η_{CODE}	SinDE
F1	5.75E+03	1.11E+05	7.55E+02	2.87E+02	6.18E+05	1.22E+05	2.92E+03	5.41E-04	8.56E+05
F2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F3	0.00E+00	0.00E+00	5.41E-04	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F4	0.00E+00	3.50E-01	0.00E+00	0.00E+00	1.84E+00	4.00E-01	7.82E-02	3.04E+00	7.73E-01
F5	2.03E+01	2.09E+01	2.03E+01	2.02E+01	2.06E+01	2.05E+01	2.02E+01	2.00E+01	2.03E+01
F6	6.78E-01	6.00E-02	9.98E+00	9.98E+00	1.67E-01	6.44E-01	1.23E+00	1.17E+00	9.02E-02
F7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.90E-04	1.67E-03	0.00E+00
F8	0.00E+00	1.53E+02	0.00E+00	0.00E+00	2.40E-02	9.51E+00	0.00E+00	0.00E+00	6.57E+00
F9	2.34E+01	1.83E+02	4.59E+01	2.41E+01	1.07E+02	3.99E+01	1.95E+01	1.56E+01	2.48E+01
F10	7.63E-03	6.00E+02	6.94E-03	2.78E-03	2.11E+02	3.76E+01	1.10E-02	9.84E-03	9.99E+01
F11	1.40E+03	6.97E+03	2.64E+03	1.63E+03	4.59E+03	2.07E+03	1.57E+03	1.56E+03	1.70E+03
F12	2.89E-01	2.56E+00	3.64E-01	2.15E-01	9.26E-01	4.35E-01	2.10E-01	1.34E-01	1.78E-01
F13	1.93E-01	3.74E-01	3.11E+00	2.31E-01	3.22E-01	2.06E-01	2.06E-01	1.80E-01	1.73E-01
F14	2.10E-01	2.67E-01	2.41E+00	2.50E-01	2.73E-01	2.25E-01	2.19E-01	2.38E-01	2.26E-01
F15	2.23E+00	1.55E+01	4.18E+00	2.81E+00	1.06E+01	3.77E+00	3.00E+00	2.55E+00	3.50E+00
F16	8.95E+00	1.26E+01	9.32E+00	9.46E+00	1.13E+01	9.50E+00	9.42E+00	9.26E+00	8.80E+00
F17	3.73E+02	1.51E+03	1.23E+03	9.65E+02	2.48E+03	7.84E+03	1.24E+03	8.31E+02	5.18E+03
F18	1.64E+01	5.72E+01	7.79E+01	5.77E+01	4.33E+01	1.66E+01	7.81E+01	3.48E+01	1.74E+01
F19	3.37E+00	4.74E+00	4.42E+00	4.67E+00	4.95E+00	2.98E+00	4.50E+00	4.58E+00	2.55E+00
F20	8.48E+00	3.83E+01	2.86E+03	1.48E+01	2.15E+01	7.34E+00	1.99E+01	1.02E+01	1.09E+01
F21	1.52E+02	8.19E+02	1.58E+04	2.45E+02	5.54E+02	3.34E+02	3.15E+02	2.22E+02	3.59E+02
F22	1.13E+02	8.42E+01	1.64E+02	1.30E+02	4.97E+01	3.98E+01	1.37E+02	5.96E+01	9.91E+01
F23	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02	3.15E+02
F24	2.24E+02	2.16E+02	2.26E+02	2.27E+02	2.23E+02	2.00E+02	2.25E+02	2.28E+02	2.24E+02
F25	2.03E+02	2.03E+02	2.04E+02	2.03E+02	2.03E+02	2.03E+02	2.05E+02	2.03E+02	2.03E+02
F26	1.00E+02	1.00E+02	1.00E+02	1.04E+02	1.00E+02	1.00E+02	1.02E+02	1.00E+02	1.00E+02
F27	3.56E+02	3.67E+02	3.50E+02	3.35E+02	3.45E+02	3.19E+02	3.40E+02	3.52E+02	3.07E+02
F28	8.12E+02	7.63E+02	7.85E+02	8.17E+02	7.91E+02	7.87E+02	8.00E+02	8.29E+02	8.02E+02
F29	7.02E+02	3.43E+02	7.29E+02	7.21E+02	9.10E+02	8.88E+02	7.35E+02	6.76E+02	9.15E+02
F30	8.82E+02	7.66E+02	1.53E+03	1.77E+03	1.08E+03	1.14E+03	1.56E+03	1.24E+03	1.32E+03
Wil. p value	–	3.19E-02	4.20E-03	5.10E-03	3.60E-03	4.62E-02	2.80E-02	1.24E-01	2.63E-02
F.A.R.	3.32	5.90	6.00	4.88	6.12	4.63	5.17	4.23	4.75
S.R.E.	11.28	17.58	19.32	13.82	15.73	12.64	13.28	13.71	13.49

Fig. 4 Mean converge speeds of RAM-JAPDE, CoDE, EPSDE, JADE, jDE, SaDE, SHADE, ZEPDE on 12 functions from CEC'14 benchmark set. All results are mean values of 50 runs

Table 5 Comparison between RAM-JAPDE and nine adaptive Differential Evolution algorithms in dimension 10 using three measures: 1) Wilcoxon's statistical test ($\alpha = 0.05$) of RAM-JAPDE vs. other algorithms, 2) the average ranking according to Friedman statistical test (F.A.R.) and 3) the summation of the relative errors with respect to the worst (S.R.E.). *p* values in boldface means that there are statistically significant differences in favour of RAM-JAPDE

	Wil. <i>p</i> value	F.A.R.	S.R.E.
RAM-PADE	–	2.72	11.43
CoDE	3.21E–04	7.45	20.8
EPSDE	1.19E–07	7.47	19.71
ZEPDE	1.89E–06	5.55	14.76
SA-SHADE	3.40E–02	4.28	13.61
EFADE	2.49E–02	4.12	13.05
SaDE	1.19E–07	5.68	15.39
JADE	3.20E–07	7.08	20.24
jDE	4.57E–06	6.77	17.28
SHADE	1.87E–02	3.88	13.90

Table 6 Comparison between RAM-JAPDE and nine adaptive Differential Evolution algorithms in dimension 50 using three measures: 1) Wilcoxon's statistical test ($\alpha = 0.05$) of RAM-JAPDE vs. other algorithms, 2) the average ranking according to Friedman statistical test (F.A.R.) and 3) the summation of the relative errors with respect to the worst (S.R.E.). *p* values in boldface means that there are statistically significant differences in favour of RAM-JAPDE

	Wil. <i>p</i> value	F.A.R.	S.R.E.
RAM-PADE	–	3.52	12.02
CoDE	5.65E–01	4.95	16.81
EPSDE	2.11E–05	8.45	24.34
ZEPDE	1.13E–02	6.02	17.41
SA-SHADE	4.51E–02	4.35	13.48
EFADE	2.40E–03	5.35	14.36
SaDE	3.20E–07	6.42	15.67
JADE	8.20E–07	6.92	16.99
jDE	1.79E–02	5.65	16.09
SHADE	6.79E–01	3.38	11.90

higher probabilities than those closer to 0 in order to allow more exploration during the search. Additionally, it can be observed that it is wider along F candidates than along CR candidates. This happens due to the usage of a Cauchy distribution when F values are created.

Through the different generations, *M* changes and adapts to the different functions. Some examples are given in Fig. 6. Different observations can be made from this figure. First, a lot of high values inside *M* mean that many combinations will have a high probability of being selected. Second, there is a big change from generation 300 to generation 1200, in which bad combinations had a really low probability of being used again. Third, from generation 1200 to generation 2400

Table 7 Comparison between RAM-JAPDE and nine adaptive Differential Evolution methods in dimension 100 using three measures: 1) Wilcoxon's statistical test ($\alpha = 0.05$) between RAM-JAPDE vs. other algorithms, 2) the average ranking according to Friedman statistical test (F.A.R.) and 3) the summation of the relative errors with respect to the worst (S.R.E.). *p* values in boldface means that there are statistically significant differences in favour of RAM-JAPDE

	Wil. <i>p</i> value	F.A.R.	S.R.E.
RAM-PADE	–	3.07	14.17
CoDE	2.61E–07	7.82	21.42
EPSDE	3.18E–05	6.95	21.18
ZEPDE	4.80E–02	3.82	14.68
SA-SHADE	2.03E–02	5.18	17.38
EFADE	2.16E–02	5.48	16.75
SaDE	2.05E–05	5.53	17.51
JADE	6.56E–07	7.37	19.42
jDE	1.70E–03	6.65	21.00
SHADE	6.46E–01	3.13	14.25

Fig. 5 Probability of obtaining a *f-cr* combination from each group using *meanF* and *meanCR* values equal to 0.5

there are almost no change with respect to the probabilities, meaning that good combinations of parameters are found. Finally, it can be seen that good combinations of parameters normally have extreme values of CR (close to 0 or 1). Additionally, F values close to 1 obtained higher probabilities when selecting CR equal to 0, while F values around 0.5 have higher probabilities when selecting CR values close to 1.

5.8 Comparison of the used F and CR values by RAM-JAPDE and SHADE

In this subsection, a comparison between RAM-JAPDE and SHADE, with regards to the used F and CR values, is per-

Fig. 6 Evolution of M in four different functions (F1, F5, F18 and F25). A value equal to 0.016 (dark red) means that the F-CR pair has a probability equal or greater than 0.03

Fig. 7 Used F and CR values by RAM-JAPDE and SHADE in four different functions (F5, F15, F18, and F30). Blue points show unsuccessful parameters, while red points show successful parameters

formed (Fig. 7). Usually, two different situations are found with regards to SHADE and CR values.

Firstly, F5 and F15 are considered (See Fig. 7a, b). In these two functions, SHADE will tend to use CR equal to 0, while RAM-JAPDE will mostly use small values of CR in F5 and small and high values in F15. SHADE obtained a better performance in F5, since CR equal to 0 is the best option for this problem. However, in F15, RAM-JAPDE outperformed SHADE thanks to its ability to adapt to different search stages. It can be seen from Fig. 7b that in later stages, RAM-JAPDE started using high values (as well as small values), which seems beneficial for its performance, while SHADE got stuck on CR values equal to 0, and consequently obtaining poorer performance.

Secondly, F18 and F30 are considered (Fig. 7c, d). In these two functions, it can be seen how high values of CR are beneficial for both algorithms. The difference between both algorithms is that SHADE took a greedy approach and moved towards high CR values, while RAM-JAPDE used high and small values (which shows how both are beneficial for RAM-JAPDE). Over time, it can be observed how SHADE get penalized for that greedy decision, and it stopped its improvement (See Fig. 4h, l, respectively). On the other

hand, thanks to the approach used in RAM-JAPDE, which instead of only using high CR values, it also uses small CR values. RAM-JAPDE is able to keep improving for a longer time.

On the other hand, if F values are considered, it can be observed how a larger range of successful values exist. RAM-JAPDE keeps a high variability of the used F through the generations. On the contrary, SHADE tends to focus on some specific range of F values. This focus changes through the generations.

5.9 Comparison of RAM-JAPDE with L-SHADE

In this subsection, RAM-JAPDE is compared with L-SHADE, the winner of the CEC2014 competition. The parameters for the algorithms are listed below:

- L-SHADE: $N^{init} = 18 \times n$, $N^{arc} = 1.4 \times PS$, $p = 0.11$ and $H = 6$, as specified in their paper.
- RAM-JAPDE: $PS = 100$, $A = 100$, $gr = 10$, $LP_{RAM} = LP_{PA} = 30$ and $E_{RAM} = E_{PA} = 0.05$.

Table 8 Wilcoxon's statistical test ($\alpha = 0.05$) between RAM-JAPDE and L-SHADE in dimensions 10, 30, 50 and 100

Dimension	<i>p</i> value	Stat. Diff.?	Which alg.?
10	8.46E-01	No	None
30	6.03E-03	Yes	L-SHADE
50	9.52E-04	Yes	L-SHADE
100	7.85E-02	No	None

Table 9 Wilcoxon's statistical test ($\alpha = 0.05$) between L-RAM-JAPDE and L-SHADE in dimensions 10, 30, 50 and 100

Dimension	<i>p</i> value	Stat. Diff.?	Which alg.?
10	5.29E-01	No	None
30	1.84E-01	No	None
50	3.93E-01	No	None
100	7.85E-02	No	None

The results in (Table 8) show that L-SHADE (Tanabe and Fukunaga 2014) is statistically better than RAM-JAPDE in dimensions 30 and 50, while both are statistically similar in dimensions 10 and 100. This difference is caused by the linear reduction of the population size implemented in L-SHADE. Besides, RAM-JAPDE uses a constant population size independent of the dimension of the problem, while L-SHADE uses varying population sizes dependent on the dimension thereby better adapting itself to changes of dimensionality.

In order to further study the capability of RAM-JAPDE, we attempted to apply the same linear reduction of the population size as implemented in L-SHADE in RAM-JAPDE. The new version of the algorithm is stated as L-RAM-JAPDE. L-RAM-JAPDE uses the same parameters as RAM-JAPDE, except for the population size and the archive size that are the same as in L-SHADE, and LP is set to 3000 fitness evaluations. The results in (Table 9) show that L-RAM-JAPDE is statistically similar to L-SHADE in dimensions 10, 30, 50 and 100.

Observing the results of L-RAM-JAPDE also leaves us with the sense that reducing population during the search does not show strong potential to further enhance the performance of RAM-JAPDE. This can be attributed to the increasingly smaller populations caused by the population size reduction scheme. This would result in too many generations in a learning period and thereby delay the response of parameter adaptation given the accelerated convergence. On the other hand, smaller population makes less important to have multiple probability distributions in the adaptation of mutation strategies.

6 Conclusion

Selecting the best combination of control parameters and the best mutation strategies in DE can be an arduous task. This paper proposes a new parameter adaption method called Joint Adaptation of Parameters in DE (JAPDE), which jointly adapts the generation of the mutation factor and crossover rate during the running of DE. The key trick is to maintain and dynamically update a matrix of selection probabilities for a set of pairs of parameter creating functions, which covers the whole range of both parameters. Additionally, it adds an exploration plane that will enforce the search towards more exploration. Further we combine JAPDE with the Rank-Based Adaptation (RAM) method developed in our recent study (Leon and Xiong 2018), leading to the new RAM-JAPDE algorithm. RAM is able to adapt the selection between two strong mutation strategies (DE/pBest/1 and DE/current-to-pBest/1) using different probability distributions for individuals belonging to distinct rank intervals. Moreover, RAM-JAPDE linearly decreases the parameter *p* inside both mutation strategies to allow for more exploration at the start of the search yet more exploitation at the end.

RAM-JAPDE has been evaluated in CEC2014 benchmark suit and compared with well-known DE algorithms. From the experiments, it was found that RAM-JAPDE was the best algorithm among those that also perform adaptation of mutation strategies, as well as those that adapt the mutation factor and crossover rate. Moreover, RAM-JAPDE was tested on different problem dimensions, and it obtained the same superiority to its counterparts. Additionally, the convergence speed of RAM-JAPDE was evaluated and it can be concluded that RAM-JAPDE enables a better exploration/exploitation trade-off than its counterparts.

A practical limitation of the proposed RAM-JAPDE algorithm is that it cannot be expected as highly effective when the population size is small. One reason is that a small population entails more generations in a learning period to evaluate all combinations of *F* and *CR* values. This would likely prevent parameter adaptation to be conducted in a timely manner, considering the fast convergence of DE with a small population. On the other hand, a small number of individuals seems to give less motivation to maintain multiple probability distributions in mutation strategy selection, which would mitigate the strength of RAM that is used in our algorithm.

Acknowledgements Open access funding provided by Mälardalen University.

Compliance with ethical standards

Conflict of interest M. Leon declares that he has no conflict of interest. N. Xiong declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bakare GA, Krost G, Venayagamoorthy GK, Aliyu UO (2007) Comparative application of differential evolution and particle swarm techniques to reactive power and voltage control. In: International conference on intelligent systems applications to power systems (2007) ISAP 2007. Toki Messe, Niigata, pp 1–6
- Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans Evol Comput* 10:646–657
- Dawar D, Ludwig SA (2018) Effect of strategy adaptation on differential evolution in presence and absence of parameter adaptation: an investigation. *J Artif Intell Soft Comput Res* 8(3):211–235. <https://doi.org/10.1515/jaiscr-2018-0014>
- Deng L, Sun H, Zhang L, Qiao L (2019) η_{CODE} : a differential evolution with η_{Cauchy} operator for global numerical optimization. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2019.2926422>
- Derrac J, Garcia S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a method for comparing evolutionary and swarm intelligence Algorithms. *Swarm Evol Comput* 1:3–18
- Draa A, Bouzoubia S, Boukhalfa I (2015) A sinusoidal differential evolution algorithm for numerical optimisation. *Appl Soft Comput J* 27:99–126. <https://doi.org/10.1016/j.asoc.2014.11.003>
- Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. *IEEE Trans Evol Comput* 3(2):124–141
- Fan Q, Yan X (2016) Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. *IEEE Trans Cybern* 46(1):219–232
- Islam SM, Das S, Ghoshand S, Roy S, Suganthan PN (2012) An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Trans Syst Man Cybern Part B Cybern* 42(2):482–500
- Leon M, Xiong N (2014) Investigation of mutation strategies in differential evolution for solving global optimization problems. In: *Artificial Intelligence and Soft Computing*. Springer, pp 372–383
- Leon M, Xiong N (2017) Alopex based mutation strategy in differential evolution. In: 2017 IEEE congress on evolutionary computation (CEC), pp 1978–1984
- Leon M, Xiong N (2018) Enhancing adaptive differential evolution algorithms with rank-based mutation adaptation. In: 2018 IEEE congress on evolutionary computation, CEC 2018—proceedings, pp 1–7. <https://doi.org/10.1109/CEC.2018.8477879>
- Leon M, Evstedt M, Xiong N (2015) Adaptive differential evolution supports automatic model calibration in furnace optimized control system. In: International joint conference on computational intelligence, pp 42–55
- Leon M, Zenlanter Y, Xiong N, Herrera F (2016) Design optimal harmonic filters in power systems using Greedy adaptive differential evolution. In: IEEE 21st international conference on emerging technologies and factory automation (ETFA), pp 1–7
- Leon Miguel, Xiong Ning (2019) A novel memetic framework for enhancing differential evolution algorithms via combination with Alopex local search. *Int J Comput Intell Syst*
- Liang JJ, Qu B, Suganthan PN (2013a) Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Technical report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore
- Liang JJ, Qu BY, Suganthan PN, Hernandez-Diaz AG (2013b) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical report, Ehngzhou University and Nanyang Technological University
- Mallipeddi R, Suganthan PNP, Pan KQ, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl Soft Comput* 11(2):1679–1696
- Mohamed AW, PonnuthuraiNagaratnam Suganthan (2018) Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation. *Soft Comput* 22(10):3215–3235. <https://doi.org/10.1007/s00500-017-2777-2>
- Perez-Gonzalez A, Begovich-Mendoza O, Ruiz-Leon J (2018) Modeling of a greenhouse prototype using PSO and differential evolution algorithms based on a real-time LabView application. *Appl Soft Comput* 62:86–100
- Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans Evol Comput* 13(2):398–417
- Rahnamayan S, Tizhoosh Hamid R, Salama Magdy MA (2006) Opposition-based differential evolution algorithms. In: 2006 IEEE congress on evolutionary computation, number January, pp 7363–7370, ISBN 0-7803-9487-9. <https://doi.org/10.1109/CEC.2006.1688554>
- Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech rep. tr-95-012, Comput. Sci. Inst., Berkeley, CA, USA
- Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim* 11(4):341–359
- Sun G, Peng J, Zhao R (2018) Differential evolution with individual-dependent and dynamic parameter adjustment. *Soft Comput* 22(17):5747–5773. <https://doi.org/10.1007/s00500-017-2626-3>
- Suresh D, Lal S (2017) Modified differential evolution algorithm for contrast and brightness enhancement of satellite image. *Appl Soft Comput* 61:622–641
- Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for Differential Evolution. In: 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, pp 71–78
- Tanabe R, Fukunaga AS (2014) Improving the search performance of SHADE using linear population size reduction. In: IEEE congress on evolutionary computation (CEC), pp 1658–1665
- Teo J (2006) Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput Fusion Found Methodol Appl* 10:673–686
- Viktorin A, Pluhacek M, Senkerik R (2016) Success-history based adaptive differential evolution algorithm with multi-chaotic framework for parent selection performance on CEC2014 benchmark set. In: IEEE congress on evolutionary computation (CEC), pp 4797–4803
- Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans Evol Comput* 15(1):55–66

- Xiong N, Molina D, Ortiz ML, Herrera F (2015) A walk into meta-heuristics for engineering optimization: principles, methods and recent trends, ISSN 18756883
- Yildiz YE, Altun O, Topal AO (2015) The effects of crossover and mutation rates on chemotaxis differential evolution optimization algorithm. *J Nat Tech Sci* 1:89–101
- Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. *IEEE Trans Evolut Comput* 13:945–958

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.