Expert Systems With Applications 123 (2019) 54–69

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Association rule mining based parameter adaptive strategy for differential evolution algorithms

Chuan Wang*, Yancheng Liu, Qinjin Zhang, Haohao Guo, Xiaoling Liang, Yang Chen, Minyi Xu, Yi Wei

Marine Engineering College, Dalian Maritime University, Dalian 116026, People's Republic of China

ARTICLE INFO

Article history:
Received 14 June 2018
Revised 27 November 2018
Accepted 9 January 2019
Available online 11 January 2019

Keywords:
Differential evolution
Association Rule Mining
Parameter adaption

ABSTRACT

It is a very challenging and important task to adaptively adjust the scale factor *F* and the crossover rate *Cr* for Differential Evolutionary (DE) algorithms. Most recent adaptive techniques were designed to generate parameters randomly based on successful trial values during the previous evolving process, lacking explicit guidelines to generate appropriate values. This paper proposes a novel parameter adaption strategy, which could incorporate promising *F* and *Cr* pairs extracted by using Association Rule Mining (ARM) into DE algorithms. First, all successful *F* and *Cr* values generated by their original methods are recorded during the whole evolution, resulting in an increasing dataset. Second, we discretize the dataset and extract the most frequent itemset of parameters by using a modified version of the widely used Apriori algorithm. Third, a greedy operator is developed to generate new parameters in the next generation by comparing the presented ARM-based and original-method-based fitness values. The presented technique provides an additional pair of *F* and *Cr* values to be evaluated, without replacing existing strategies for the control parameters. The main contribution of this paper is that we propose a novel way, which utilizes information generated during the evolutionary process, to enhance exploration capabilities by adjusting control parameters. Experimental results demonstrate that the proposed ARM-based parameter adaptive strategy is able to enhance performances of some state-of-the-art DE variants. Further, this methodology might be helpful for other control parameters of Evolutionary Algorithms (EA).

1. Introduction

Since 1995, Differential Evolution (DE) algorithm and its variants have become very popular in evolutionary computing community (Storn & Price, 1997). DE family also has been successfully and widely used to solve many real-world engineering problems on diverse domains (Arce, Zamora, Sossa, & Barron, 2018; Baig, Aslam, Shum, & Zhang, 2017; Buba & Lee, 2018; Cui et al., 2018; Das & Suganthan, 2011; Das, Mullick, & Suganthan, 2016; Hu et al., 2018; Mlakar, Fister, Brest, & Potocnik, 2017; Muangkote et al., 2017; Neri & Tirronen, 2010; Tsakiridis, Theocharis, & Zalidis, 2017). The performance of DE algorithm mainly depends on the evolutionary operators and the control parameters which are usually the scaling factor *F* and the crossover rate *Cr* (Cui, Li, Lin, Chen, & Lu, 2016). Selecting the most appropriate parameter values is a task, which requires adjusting processes by trial-and-error. Therefore, it is also a time-consuming task. Meanwhile, it also depends on given optimization problems. This approach is not suitable when preknowledge is needed, nor the problem would be optimized in an automated environment (Mallipeddi, Suganthan, Pan, & Tasgetiren, 2011). Thus, it is very important to adaptively and automatically tune *F* and *Cr* by using potential helpful information during the whole evolution process.

Since the last decade, a good number of articles which have focused on adapting two main control parameters, usually the scale factor *F* and the crossover rate *Cr*, have emerged. A DE variant with an adaptive parameter control was proposed in (Elsayed, Sarker, & Ray, 2013). The authors designed two candidates sets, which store *F* and *Cr* values selected from the top10-40% individuals. First, a counter is used and initialized to zero for each of the combinations of *F* and *Cr*. A trial vector is generated by randomly selecting a pair of the *F* and *Cr* from the sets. If the trial vector has better fitness value, the counter automatically is incremented by 1. After some generations the top 50% of valuable combinations of *F* and *Cr* are selected, their counters are reset to zero and the rest are discarded. Later, this work has been improved (Sarker, Elsayed, & Ray, 2014). Meanwhile, an improved version of Self-adaptive Differ-

* Corresponding author.
E-mail addresses: labrador@dlmu.edu.cn (C. Wang), liuyc@dlmu.edu.cn (Y. Liu), zqj20@dlmu.edu.cn (Q. Zhang), ghh1984@dlmu.edu.cn (H. Guo), liangxl@dlmu.edu.cn (X. Liang), shangruihan@dlmu.edu.cn (Y. Chen), xuminyi@dlmu.edu.cn (M. Xu), weiyi@dlmu.edu.cn (Y. Wei).

https://doi.org/10.1016/j.eswa.2019.01.035
0957-4174/

ential Evolution (JADE) named Success History based DE (SHADE) was proposed in (Tanabe et al., 2013). Instead of generating the *F* and *Cr* values by using random number generators that obey different probability distributions, the researchers have used successful memory records which store *F* and *Cr* values in the last past generations. Further, new *F* and *Cr* pairs are generated by directly sampling the parameter space close to the successful memory records. An individual dependent adaptation scheme was proposed by using a two-stage process (Yu et al., 2014). In the first stage, the authors evaluate the population to determine whether the population is explorative or exploitative. Based on the population state, *F* and *Cr* are adapted. In the second stage, the fitness value and the distance of each individual from the best one are utilized to estimate and change the population settings for each individual. Six adaptive schemes based on sinusoidal functions were developed to nonlinearly renew *F* and *Cr* values along with their changing directions (Draa, Bouzoubia, & Boukhalfa, 2015). In these designs, *F* and *Cr* values could be obtained by using scaled/shifted sine expressions or classic constants, such as $F = 0.5$ and $Cr = 0.9$. That is to say, the authors present a hybrid parameter adaption method which mixes adaptive and fixed schemes. Distributed DE sub-populations and three updating schemes were introduced in (De Falco, Della Cioppa, Maisto, Scafuri, & Tarantino, 2014). Meanwhile, there is a migration policy among the sub-populations. While adapting *F* and *Cr*, the algorithm also introduced a new parameter *T*. Some scholars have proposed a scheme to adapt *F* and *Cr* values in DE by using a Gaussian adaptation method, which is a stochastic process using a Gaussian distribution to a region in some promising search domains (De Falco et al., 2014). Recently, some researchers recommended that the most suitable control parameter pairs and corresponding trial vectors generation could be obtained by observing fitness values at individual level (Tang, Dong, & Liu, 2015). The authors provided *F* and *Cr* expressions that use maximum, minimum and the difference between minimum and second minimum fitness values in the current generation for each individual. More interesting parameter adaptation mechanisms could be seen in various DE variants, such as MPDE (Yu & Zhang, 2011), SaMDE (Pedrosa Silva, Lopes, & Guimaraes, 2011), FiADE (Ghosh et al., 2011), MDE-*p*BX (Islam, Das, Ghosh, Roy, & Suganthan, 2012), TLBSaDE (Biswas, Kundu, Das, & Vasilakos, 2013), MDE (Zou et al., 2013) and ADE (Bujok, Tvrdik, & Polakova, 2014). In summary, for adapting *F* and *Cr*, the above methods have a few drawbacks that could be categorized into two types: (1) Most parameter adaptions have exquisite and complicated structures that are difficult to be developed by researchers. (2) Numerous of *F* and *Cr* adaptions also introduce new control parameters. Based on the observations, we will introduce a novel Association Rule Mining (ARM) based parameter adaptive strategy for DE algorithms to overcome the above shortcomings.

ARM was first developed to mine large collections of basket data type transactions. Due to its effectiveness, ARM has been widely applied in different domains to mine frequent patterns in various datasets. To make the scope of this paper more focused, we assume that readers have known some basic concepts of ARM. More definitions and details could be seen in numerous published articles (Cohen et al., 2001; Djenouri & Comuzzi, 2017; Doostan & Chowdhury, 2017; Harikumar & Dilipkumar, 2016; Patill et al., 2016; Sheng, Hou, Jiang, & Chen, 2018; Tung, Lu, Han, & Feng, 2003; Yang, Tang, Shintemirov, & Wu, 2009; Zaki, 2000). Some literatures that combined ARM and Evolution Algorithm (EA) have been published. For instance, there are some articles that use EAs to optimize ARM. MODENAR (Alatas, Akin, & Karci, 2008), genetic-fuzzy data mining (Hong, Chen, Lee, & Wu, 2008), MOPNAR (Martin, Rosete, Alcala-Fdez, & Herrera, 2014) and Particle Swarm Optimization with Apriori (PSO-Apriori) (Djenouri & Comuzzi, 2017) are good examples. Also, recent reviews on multi-objective EAs for data mining could be seen in (Mukhopadhyay, Maulik, Bandyopadhyay, & Coello Coello, 2014a; Mukhopadhyay, Maulik, Bandyopadhyay, & Coello Coello, 2014b). However, very few literatures have used knowledge-based methods, such as Machine Learning (ML) or ARM, to improve performances of EAs by adapting parameters. That drives us to apply ARM method for the parameter adaption of DE algorithms even EAs. Therefore, in this paper, we apply the ARM-based parameter adaptive strategy which could extract and incorporate the most successful *F* and *Cr* couples into existing parameters adaptive methods to enhance the exploration capability. As a result, *F* and *Cr* values might be adapted automatically, resulting in better performances of DEs. This novel methodology has three advantages: (1) The proposed strategy could extract association rules automatically from successful *F* and *Cr* records along with evolution and incorporate the association rules to generate new *F* and *Cr* values. That means an easy application of *F* and *Cr* adaption for DE algorithms. (2) The process of adaption for *F* and *Cr* introduces no extra control parameters. (3) The proposed strategy could choose a better *F* and *Cr* couple from ARM-based and original-method-based values to enhance exploration capabilities of DE algorithms. That means, ARM-based parameter adaptive strategy does not replace existing strategies for the control parameters, but rather provides an additional pair of *F* and *Cr* values to be evaluated. The main contribution of this paper is that, we propose a different methodology to adapt *F* and *Cr* for DE algorithms by using the ARM-based strategy which could mine potentially outstanding *F* and *Cr* values from successful records so far. Moreover, we should discuss the main difference between the proposed ARM-based *F* and *Cr* adaptive strategy and other adaptive or self-adaptive methods for DE variants. In other methods, *F* and *Cr* are usually adapted based on previous successful memories separately. These methods do not consider the impact of the combination of *F* and *Cr*, resulting in a less efficient parameters adaption. In our proposed ARM-based method, successful *F* and *Cr* are stored in pair. Then, we use Apriori algorithm to mine the most promising combinations of *F* and *Cr* to generate new trial vectors. More details could be seen in Section 3.

The remainder of this paper is structured as follows: In the next section, related work on *F* and *Cr* adaption of DEs and Apriori algorithm are introduced. In Section 3, the ARM-based parameter adaptive strategy is proposed. In Section 4, comprehensive experimental results on IEEE Congress on Evolutionary Computation CEC2005 benchmark functions are shown and discussed. In Section 5, we apply the proposed ARM-based method for six state-of-the-art DE variants on CEC2011 real-world problems and compare the performances. Finally, the contributions of this paper are summarized and future work is outlined in Section 6.

2. Related work

In this part, some background materials are introduced. First, SaDE is chosen as an example to exhibit adaptive adjustment for *F* and *Cr*. Next, we will give a review on adaptive techniques of *F* and *Cr* in some other published papers. Last, in order to provide explicit recommend to adaptively generate *F* and *Cr* on any optimization problem, the widely used Apriori algorithm will be introduced.

2.1. Parameter-adaptive DE algorithm

There are many DE algorithms which designed versatile *F* and *Cr* adaption strategies to improve exploration and exploitation capability. In this paper, SaDE (Qin, Huang, & Suganthan, 2009) is selected as a paradigm due to its highly cited times. Later, we will embed the proposed ARM-based parameters adaptive strategy into SaDE and other state-of-the-art DE variants. Before that, the main

Algorithm 1 The main structure of SaDE algorithm.

```
Uniformly randomly initialize each individual within the searching range;
Calculate the fitness values of the initial population;
Set generation number G = 1; set population size as NP; set function evaluations as Fes = NP; set maximum function evaluations as Max_Fes; set
  learning period LP = NP; initial median value of Cr (Crm_l) and strategy probability p_{l,G} (l = 1, ..., L, L is the number of available strategies, L = 4);
  Clear Success and Failure Memory;
while Fes ≤ Max_Fes
    if G > LP
        Update p_{l,G} based on lth strategy performance in last LP generations;
        Update Success and Failure Memory;
    end
    for i = 1: NP
        % Mutation strategy adaption
        Select one strategy l for X_{i,G} by using roulette wheel selection according to p_{l,G};
        % Mutation
        F_i = normrnd(0.5, 0.3);
        Generate mutation vector V_{i,G} according to corresponding strategy l and F_i;
        % Crossover
        Calculate lth crossover rate for V_{i,G} using Crm_l = median(Cr_memory_l);
        Cr_i = normrnd(Crm_l, 0.1);
        Generate trial vector U_{i,G} according to corresponding strategy l and Cr_i;
        Calculate the fitness value of U_{i,G};
        Fes = Fes + 1;
        % Selection
        if f(U_{i,G}) < f(X_{i,G})
            X_{i,G+1} = U_{i,G};
            Add Cr_i into Cr_memory_l;
    end
    Add successful and failure information of lth mutation strategy into Success and Failure Memory, respectively;
  end
  G = G + 1;
end
```

description of SaDE is given as below. More details could be seen in the original paper (Qin et al., 2009).

- Mutation

After initialization, for each target vector $\mathbf{X}_{i,G}$, lth mutation strategy is selected by using roulette wheel selection from the candidate strategy pool that contains four mutation strategies, such as DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin and DE/current-to-rand/1, where $l = 1, 2, 3, 4$. All these mutation strategies are listed as below. One mutation strategy is selected according to the probability, $p_{l,G}$, learning from its success rate in generating improved solutions within a certain number of previous generations.

DE/rand/1/bin:

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right) \tag{1}$$

DE/rand-to-best/2/bin:

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}\right) + F \cdot (\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}) + F \cdot \left(\mathbf{X}_{r_3,G} - \mathbf{X}_{r_4,G}\right) \tag{2}$$

DE/rand/2/bin:

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right) + F \cdot \left(\mathbf{X}_{r_4,G} - \mathbf{X}_{r_5,G}\right) \tag{3}$$

DE/current-to-rand/1:

$$\mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{r_1,G} - \mathbf{X}_{i,G}) + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right) \tag{4}$$

where the indices r_1, r_2, r_3, r_4, r_5 are mutually exclusive randomly generated integers, that are also different from i, within the range from 1 to NP. The indices are randomly generated once for each mutant vector. $\mathbf{X}_{\text{best},G}$ is the best individual with the best fitness value so far. In mutation operator, the parameter F is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by normrnd (0.5, 0.3) in Algorithm 1. A set of F values are randomly sampled from such normal distribution and applied to each target vector in the current population.

- Crossover

After the mutation phase, crossover operation is applied to each pair of the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$ to generate a trial vector $\mathbf{U}_{i,G}$. In the basic version, the binomial crossover is employed as follows:

$$u_{i,G}^{j} = \begin{cases} v_{i,G}^{j}, & \text{if } \ rand_j \le Cr \text{ or } j = j_{rand} \\ x_{i,G}^{j}, & \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, D \tag{5}$$

where, j_{rand} is a randomly chosen integer within the range [1, D]. $rand_j$ is uniformly distributed random numbers independently generated within [0, 1]. $j = j_{rand}$ is developed to ensure that the trial vector will be different from its corresponding target vector by at least one dimension. Note that, target vector $\mathbf{X}_{i,G}$, which is mutated by using DE/current-to-rand/1, will not enter crossover phase. Cr obeys a normal distribution with mean value Crm and standard deviation $std = 0.1$, denoted by normrnd (Crm, 0.1). To adapt Cr, memories that store those Cr values with respect to the lth strategy that has generated trial vectors successfully entering the next generation within the previous LP generations. At each generation after LP generations, the median value stored in memories will be calculated to overwrite Crm_l. Then, Cr values can be generated according to normrnd (Crm_l, 0.1) when applying the lth strategy.

- Selection

Before selection operation, the objective function values of all trial vectors are evaluated. After that, a selection operation is performed. $f(\mathbf{U}_{i,G})$ and $f(\mathbf{X}_{i,G})$ are objective function values of trial vector and its corresponding target vector at generation G, respectively. The selection operation can be expressed as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } \ f(\mathbf{U}_{i,G}) < f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \tag{6}$$

The beyond three basic operations are repeated iteration by iteration till the specified termination criteria are met. The algorithmic description is shown as Algorithm 1. In order to make Algorithm 1 more readable, high-level pseudo code is given. More details on the adaption of selected probability could be seen in the

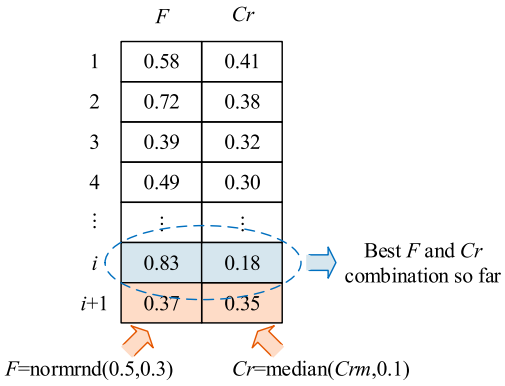

Fig. 1. A simplified example of *F* and *Cr* generation of SaDE.

original paper (Qin et al., 2009). At the generation *G*, after evaluating all the generated trial vectors, the numbers of trial vectors generated by the *l*th strategy that can successfully enter the next generation and be discarded in the next generation, are stored as *Success* and *Failure Memory*. After the initial *LP* generations, the probabilities of choosing different strategies will be updated at each subsequent generation based on the *Success* and *Failure Memory*.

From the above pseudo code, *F* and *Cr* are random numbers that obey normal distribution. There is a drawback for such an adaption: such parameter-adaptive method may not generate the most appropriate combinations within a few generations. We are trying to illustrate this by using a simplified example shown in Fig. 1. We assume that *F* and *Cr* values will be generated iteration by iteration. At *i*th iteration, the *F* and *Cr* couple circled with blue dash line represents the best combination so far. It is the most promising parameter setting that *F* is bigger than 0.8 and *Cr* is smaller than 0.2 currently. In the next iteration, *F* and *Cr* should be generated near such values. However, according to $F = \text{normrnd}(0.5, 0.3)$ and $Cr = \text{normrnd}(Crm_l, 0.1)$ in SaDE, the new parameter values may be far from the best one. That is, it may spend many *Fes* to generate a promising trial vector with such *F* and *Cr*, resulting in a less efficient evolution process. Many other adaptive DE variants have the same drawback. Consider that, we will introduce a novel parameters adaptive strategy based on ARM to speed up *F* and *Cr* tuning procedure. Section 3 presents more details.

2.2. Previous work on adaption of F and Cr

There are three control parameters *F*, *Cr*, and *NP* for classic DE algorithm. Numerical values of the three parameters should be chosen according to different problems and evolving stages. In this paper, we leave *NP*, which highly replies on the complexity of a given problem, as a fixed and pre-defined parameter. As claimed in SaDE (Qin et al., 2009), "the population size does not need to be fine-tuned and just a few typical values can be tried according to the pre-estimated complexity of the given problem".

For the other two parameters, *F* and *Cr* are related to convergence speed and different characteristics, respectively. Storn and Price (1997) recommended that *F* is taken between 0.4 and 1 and *Cr* is set to 0.1 or 0.9. Note that, different *F* and *Cr* couples are responsible for different searching behaviors. For example, $F = 1$ and $Cr = 0.9$ can provide good exploration capability in searching space, while $F = 0.5$ and $Cr = 0.1$ could result in satisfying exploitation capability. A parameter study (Gamperle, D Muller, & Koumoutsakos, 2002) for DE suggested that *F* equals 0.6 and *Cr* ranges from 0.3 to 0.9. In another research literature (Ronkkonen, Kukkonen, & Price, 2005), *F* was suggested to be ranged within 0.4 and 0.95 with an initial value of 0.5. Meanwhile, *Cr* lies in [0, 0.2] for separable functions, while in [0.9, 1] when the functions are parameter-dependent and multi-modal. It was also recommended that DE with the settings $F \geq 0.6$ and $Cr \geq 0.6$ performs well on convergence in most cases (Zielinski, Weitkemper, Laur, & Kammeyer, 2006). Although these advice are beneficial, they may cause confusions for researchers and engineers when they deal with real engineering problems with unknown characteristics. Consequently, it seems that fixed parameters settings lack of universality, and could not adapt for different functions and evolving stages (Fan & Yan, 2016).

Some researchers have developed some adaptive techniques to avoid tuning *F* and *Cr* manually. A self-adaptive Pareto DE algorithm was proposed to generate *F* and *Cr* by using random number generator which obeys Gaussian distribution (Abbass, 2002). A fuzzy adaptive DE was presented to adjust *F* and *Cr* by using fuzzy logic controllers (Liu & Lampinen, 2005). A mutation strategy with an optional external archive was used in JADE (Zhang & Sanderson, 2009), and *F* and *Cr* are tuned according to their previous successful records. A self-adaptive DE algorithm (Omran, Salman, & Engelbrecht, 2005), wherein *F* is adaptive and *Cr* is a random number with a normal distribution, was introduced. In SaDE (Qin et al., 2009), *F* is generated from a normal distribution $N(0.5, 0.3)$ and *Cr* is a median value of previous successful memory. In FiADE (Ghosh, Das, Chowdhury, & Gini, 2011), the adaption of *F* and *Cr* is based on the fitness values of individuals. In jDE (Brest, Greiner, Boskovic, Mernik, & Zumer, 2006), both *F* and *Cr* are adapted at individual level. New *F* is generated randomly within the range of [0.1, 1] with a probability τ_1, and new *Cr* is generated randomly within the range of [0, 1] with a probability τ_2. Although experimental results of these algorithms have demonstrated their better performances on some test functions, most adaptive techniques automatically generate new *F* and *Cr* by using random number generator or calculated values based on previous successful archives. That means, the potentially effective associations of *F* and *Cr* hiding in previous successful combinations have been rarely noticed. If we can mine and utilize the *F* and *Cr* couples that have better fitness values during iterations, many tedious trials for pairs of *F* and *Cr* may be avoided, resulting in a more effective searching process. Next, we will introduce a widely used Apriori algorithm in ARM. How to use Apriori algorithm in parameters adaption for DE could be seen in Section 3 in details.

2.3. Apriori algorithm

One of the most widely applied ARM algorithms is Apriori algorithm proposed in the early 1990s. This algorithm could generate high dimensional frequent itemset (length *k*) from low dimensional frequent itemset (length *k*-1) over database. Finding frequent itemsets is a very important difficult task due to its combinatorial explosion. Once frequent itemsets are obtained, it is straightforward to generate association rules with larger confidence (Wu et al., 2008).

The pseudo code for the algorithm is given below for a database. Min_sup means a threshold of support value. L_k is the candidate set for level *k*. At each step, the algorithm is assumed to generate the candidate sets from the large itemsets of the preceding level, heeding the downward closure lemma. Here is an example shown as Fig. 2. We will use Apriori to determine the frequent *k*-itemset of this database. To do this, we will say that an itemset is frequent if it appears in at least 3 transactions of the database: $\text{Min\_sup} = 3/7$. The first step of Apriori is to count up the number of occurrences of each member item separately. By scanning the database for the first time, we obtain the 1-itemset in Fig. 2. All the itemsets of size 1 have a support of at least 3/7, so they are all

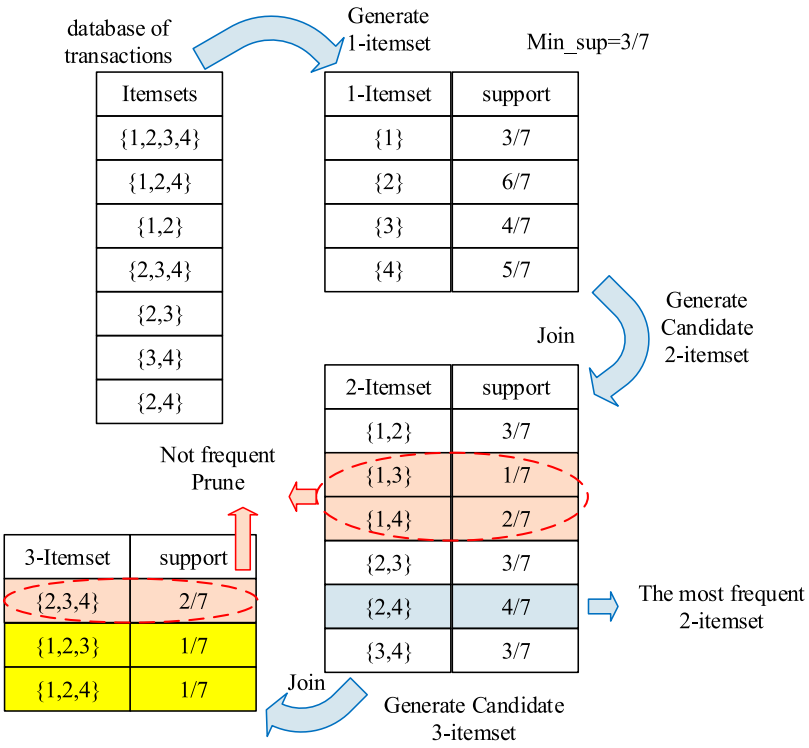

Fig. 2. An example of how Apriori algorithm works.

frequent. The next step is to generate a list of all pairs of the frequent items. The pairs {1, 2}, {2, 3}, {2, 4}, and {3, 4} all meet or exceed the minimum support of 3/7, so they are frequent. The pairs {1, 3} and {1, 4} are not. Now, because {1, 3} and {1, 4} are not frequent, any larger set which contains {1, 3} or {1, 4} cannot be frequent. In this way, we can prune sets: {1, 3} or {1, 4}. We will now look for frequent triples in the database, but we can already exclude all the triples that contain one of these two pairs. In the example, there are no frequent triplets. The itemset {2, 3, 4} is below the minimal threshold, and the other two triplets {1, 2, 3} and {1, 2, 4} are excluded because they are supersets of pairs that are already below the threshold. We have thus determined the frequent sets of items in the database, and illustrated how some items are not counted because one of their subsets was already known to be below the threshold.

In this paper, we set $k=2$. If we have the successful F and Cr couple records (transactions database), the most promising F and Cr intervals (most frequent itemsets) could be found by using Apriori algorithm. We use this methodology to adapt F and Cr for DE algorithms. More details will be seen in Section 3.

The pseudo code of Apriori algorithm for frequent k-itemset is given below. In order to make the Apriori algorithm more readable, we try to use high-level pseudo code instead of low-level pseudo code. More details could be seen in (Wu et al., 2008).

Note that, there are few limitations of Apriori algorithm. This algorithm needs many scans of the database, resulting in spending a very long time to obtain frequent itemsets when the database is large. In this paper, Apriori algorithm will be used to find association rules based on successful records that would be a small size dataset during iterations. Besides, we only need F and Cr association rules, i.e., frequent 2-itemset. Thus, the extra CPU time caused by Apriori algorithm is acceptable. Section 4.5 gives a short analysis. Another drawback of Apriori algorithm is that many trivial rules are derived and it will be hard to extract the most interesting rules. In this paper, we will only choose the generated association rule with the biggest support value.

3. ARM-based parameter adaptive strategy

In order to achieve satisfying performances for adaptive DEs, a trial-and-error scheme is usually used to tune control parameter values. This mechanism would cost many function evaluations (*Fes*). Meanwhile, in most state-of-the-art adaptive DE variants, previous successful parameter values, which may contain potentially useful information and knowledge, were often underutilized. Based on these observations, we incorporate ARM into adaptive DEs to generate explicit promising parameter pairs to speed up the evolution. The main procedures of the proposed mechanism will be listed as follows. SaDE is taken as an example to represent adaptive DE algorithms. Meanwhile, for easy understanding, some variables and descriptions are summarized in the following Table 1.

Step 1. Record every successful F and Cr into original dataset D_o. At every iteration, if a better trial vector $\mathbf{U}_{i,G}$ is generated, the corresponding F and Cr combination for $\mathbf{U}_{i,G}$ will be added into a two-column dataset, which is named as D_o. On the contrary, if the trial vector $\mathbf{U}_{i,G}$ is worse, the corresponding parameters pair is discarded. F is defined as a scale factor which usually lies between 0 and 1 (Das & Suganthan, 2011). Cr is denoted as crossover rate in the range [0, 1) (Das, Mandal, & Mukherjee, 2014). Thus, D_o is a real number two-column matrix, whose length will increase once a better trial vector is obtained. D_o is added with new F and Cr value if a better fitness value is found for each vector, as shown in line 42 and 45 of Algorithm 4. Fig. 3 shows the diagram of saving successful F and Cr into original dataset D_o.

Step 2. Discretize D_o. Before ARM, D_o should be pre-processed into a binary matrix, which is named as D_b. In Section 2.2, we have reviewed that different combinations of F and Cr may result in distinctive searching behaviors. For instance, $F=1$ and $Cr=0.9$ is mainly to maintain population diversity, while $F=0.8$ and $Cr=0.2$ is for encouraging the exploita-

Table 1
Variables and illustrations in the ARM-based parameter adaptive strategy.

Variables names	Abbreviation	Descriptions
Original dataset	D_o	A real number two-column matrix, which is used to store every F and Cr combination when the corresponding better trial vector $\mathbf{U}_i$ is generated.
Binary dataset	D_b	A binary ten-column matrix, which is a discretized matrix of D_o according to different intervals.
F and Cr most frequent itemset	L_2	An itemset contains two numbers, which represent different F and Cr distributing intervals.

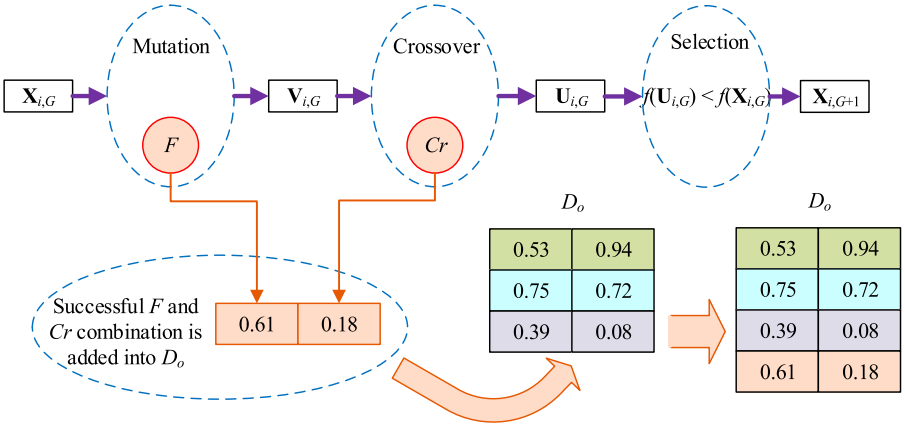

Fig. 3. Diagram of saving a successful F and Cr combination into D_o.

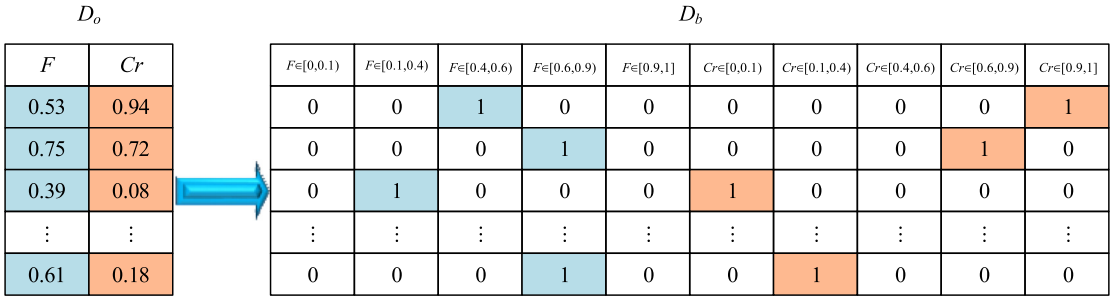

Fig. 4. Discretization diagram of converting D_o to D_b.

tion capability (Wang, Cai, & Zhang, 2011). Thus, in this paper, the two parameters could be categorized into five different intervals ((0, 0.1), [0.1, 0.4), [0.4, 0.6), [0.6, 0.9), [0.9, 1]) to represent distinctive behaviors. Consequently, F and Cr are discretized into five different intervals ((0, 0.1), [0.1, 0.4), [0.4, 0.6), [0.6, 0.9), [0.9, 1]) according to their values. Thus, D_b is a ten-column binary matrix, whose first 5 columns and last 5 columns represent F and Cr, respectively. Fig. 4 gives an example to display the discretization process.

Step 3. Find the most frequent 2-itemset by using Apriori algorithm. Based on the original Apriori algorithm for frequent k-itemset, we develop a modified Apriori-2 algorithm which could obtain the most successful combination of F and Cr so far. Here, k is set to 2 since we only need association rule of F and Cr. In this paper, the most successful combination of F and Cr represents the most frequent 2-itemset of D_b by using the Apriori-2 algorithm. In other words, in D_b, the rows, which repeat the most times, represent the most frequent F and Cr intervals. That is, the most successful combination of F and Cr. Fig. 5 gives an example. There is a D_b with 5 rows. By using the Apriori-2 algorithm, the most successful F and Cr pair ($F \in [0.6,0.9)$, $Cr \in [0.6,0.9)$) is outputted.

Meanwhile, the Apriori-2 algorithm will only be implemented at every Learning Period (LP) generations. In this paper, LP is set to population size (NP). Based on our previous experimental results, the performance of ARM-based strategy is not sensitive to how frequently we execute the Apriori-2 algorithm. For frequent 2-itemset, the Apriori-2 algorithm will generate neither repeated itemset nor the itemset which contains any infrequent sub itemset. Therefore, delete and prune operations are omitted. Details of the Apriori-2 algorithm (Algorithm 3) is described as follows.

Step 4. Generate new values based on the above association rule. From Apriori-2 algorithm, we could obtain two-column numbers, which represent different F and Cr distributing intervals. For example, if $L_2 = [3, 9]$ is returned, it means the most frequent association rule of F and Cr distribution domain is that $F \in [0.4, 0.6)$ and $Cr \in [0.6, 0.9)$. Meanwhile, recommended upper and lower limits, which are named as F_u, F_l, Cr_u and Cr_l, respectively, are outputted. Based on these explicit association rule of F and Cr intervals, new F and Cr could be generated as follows:

$$F_{ARM} = F_l + rand \cdot (F_u - F_l) \tag{7}$$

$$Cr_{ARM} = Cr_l + rand \cdot (Cr_u - Cr_l) \tag{8}$$

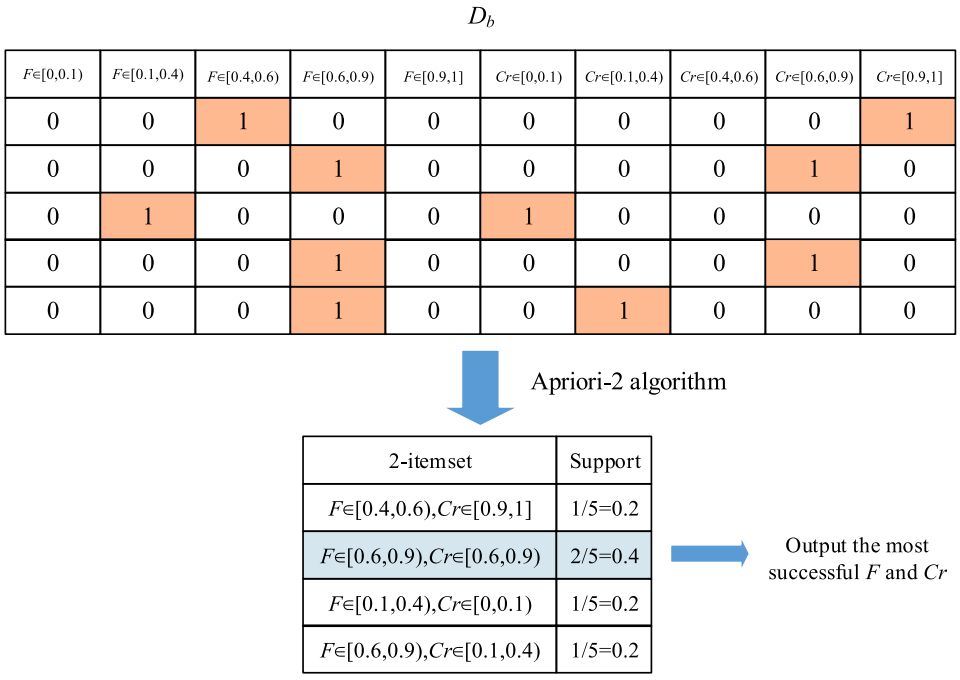

Fig. 5. Diagram of the process for generating the most successful *F* and *Cr*.

where, F_{ARM} and Cr_{ARM} mean *F* and *Cr* values generated by using the ARM-based method, *rand* is uniformly distributed random numbers independently generated within [0, 1].

Step 5. Select better candidate parameters by using a greedy operator. In order to keep good searching capabilities of state-of-the-art DE variants, this paper proposes a greedy operator by mixing ARM-based parameters adaptive strategy and original methods together. At each iteration, two trial vectors $\mathbf{U}_i$ and $\mathbf{U}_{i\_ARM}$, representing each trial vectors generated by original and ARM-based parameters adaptive strategy, respectively, are both evaluated. Then, the trial vector with better fitness value will be chosen to try to replace $\mathbf{X}_i$ in selection operation. This greedy selection operator could be described as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f(\mathbf{U}_{i,G}) < f(\mathbf{U}_{i\_ARM,G}) \,\&\, f(\mathbf{U}_{i,G}) < f(\mathbf{X}_{i,G}) \\ \mathbf{U}_{i\_ARM,G}, & \text{else if } f(\mathbf{U}_{i\_ARM,G}) < f(\mathbf{U}_{i,G}) \,\&\, f(\mathbf{U}_{i\_ARM,G}) < f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \tag{9}$$

In this modified selection operator, we evaluate both two trail vectors with different *F* and *Cr* values. This design would twice the number of function evaluations, reducing the number of generations to half. Similar situation occurs in CoDE, which compared three trial vectors with three control parameter settings. Experimental results of both CoDE and ARM-based DEs suggested that overall performances with such a design were better than competitors. It may attest to the fact that the proposed ARM-based strategy enhances the exploration capabilities of DE algorithms, which were probably stuck in local optima. Section 4.2 gives experimental results and statistical analysis.

Moreover, it is necessary to explain further why we design this greedy operator. Generally speaking, we are trying to present a parameter-adjusting method for adaptive DE variants without deteriorating performances of the original algorithms. First, the chosen state-of-the-art algorithms have been widely used and highly cited. It is unwise to abandon these well-designed and proven original parameter adaptions. Second, this presented prior-knowledge-based method assumes that previous successful combinations of parameters will be effective in the future. Nevertheless, different evolutionary stages with different fitness landscapes may require different combinations of parameters. Successful combinations of parameters in history might not result in better solutions in the future. Therefore, the authors believe that it is necessary to retain the original parameter-adjusting methods to adapt to unknown fitness landscapes. Third, experimental results demonstrate that it is effective to improve performances of DE algorithms by embedding the proposed ARM-based strategy to some extent. For example, SaDE-ARM outperforms SaDE on 9 test functions for $D = 30$. SaDE-ARM is significantly better than SaDE on 8 test functions for $D = 50$. More details could be seen in Section 4.2. Overall, the authors believe that it may be reasonable to develop such a greedy operator.

In summary, we incorporate the Apriori-2 algorithm into SaDE to show how to adapt *F* and *Cr* using mined association rules. The pseudo code of the modified algorithm, which is named as SaDE-ARM, is listed below.

Note that, this proposed ARM-based parameter adaptive methodology could be applied for any DE variant with adaptive or self-adaptive mechanism, which generates new *F* and *Cr* by trial or similar scheme. That is to say, this new technique is suitable to the algorithms that could renew control parameters in a relatively large range randomly to some extent. Thus, some DE algorithms with fixed parameters settings, such as conventional DE (Storn & Price, 1997) and CoDE (Wang et al., 2011), are not recommended to implement this method. Besides, it does not introduce new control parameter by setting every *NP* generations to mine association rules of *F* and *Cr* values. Based on our previous experimental results, the performance of ARM-based strategy is not sensitive to how frequently we execute Apriori-2 algorithm. Thus, this operation is similar to the dealing of *LP* in SaDE.

Also, there is another issue we should discuss. That is, the main difference between the proposed ARM-based *F* and *Cr* adaptive strategy and other *F* and *Cr* adaptive or self-adaptive methods for DE variants. In other methods, *F* and *Cr* are usually adapted based on previous successful memories separately. For example, in JADE,

Algorithm 2 The pseudo code of Apriori algorithm.

Input: Binary dataset D_b, minimum threshold support Min_sup
Output: Frequent k-itemsets $L_k, k = 1, 2, \ldots$

```
L_1 = ∅; k = 1;
Calculate 1-itemsets L_1 having support ≥ Min_sup from D_b
k = 2;
while L_{k-1} ≠ ∅
    C_k = ∅;
    % Join step
    Generate candidate I_tmp of k-itemset by joining each item in L_{k-1};
    for each item in I_tmp
        if length(item) == k & support(item) ≥ Min_sup
            Add item into C_k;
    end
 end
 % Prune step
 Delete repeated k-itemset in C_k;
 Prune each k-itemset in C_k which contains any infrequent itemset of
length k-1;
 Add C_k into L_k;
 k = k + 1;
end
return L_k
```

Algorithm 3 The pseudo code of Apriori-2 algorithm for F and Cr.

Input: Binary dataset D_b discretized from D_o, minimum threshold support Min_sup
Output: Most Frequent 2-itemset L_2 for F and Cr

```
L_1 = ∅;
Calculate 1-itemsets L_1 having support ≥ Min_sup from D_b
L_2 = ∅;
Generate non-repeated candidate I_tmp of 2-itemset by joining each item
in 1-itemset L_1;
for each item in I_tmp
    if support(item) ≥ Min_sup
        Add item into C_2;
    end
end
 return the itemset in L_2 with biggest support value.
```

F is renewed by using a random number generator obeys Cauchy distribution whose location parameter is calculated based on previous successful F sets. While, Cr is renewed by using a random number generator obeys Normal distribution whose mean value is calculated based on previous successful Cr sets. F and Cr are generated separately. This method has not been considered the impact of the combination of F and Cr, resulting in a less efficient parameters adaption. In our proposed ARM-based method, successful F and Cr are stored in a pair into D_o. Then, we use Apriori algorithm to mine the most promising combinations of F and Cr to generate new trial vectors. Simply speaking, we are trying to get the set of discrete (F, Cr) with the maximum successes every NP generation. The basis for this approach is that historically better combinations of parameters may be more helpful in generating new parameters in the future. We need a full-developed algorithm to find the most frequent combinations of parameters from the successful records. The authors believe that the Apriori algorithm is the most appropriate method to solve this issue. That is the reason why we chose the Apriori algorithm rather than other methods.

4. Experimental results on benchmark functions

In this section, the authors are going to conduct comprehensive experiments to test the performance of the proposed ARM-based parameters-adaptive method. We select benchmark functions presented in CEC2005 as the test suite. More details about definitions of benchmark problems can be found via http://www3.ntu.edu.sg/home/epnsugan/. The computational configurations are listed as following:

Table 2
Parameter settings of involved DE variants.

Algorithm	Setup
SaDE (Qin et al., 2009)	F = Norm(0.5, 0.3), Cr = Norm(Cr_{mk}, 0.1), $LP = 50$
JADE (Zhang & Sanderson, 2009)	$c = 0.1$, $p = 0.05$, $\mu_F = 0.5$, $\mu_{CR} = 0.5$
jDE (Brest et al., 2006)	$F_l = 0.1$, $F_u = 0.9$, $\tau_1 = \tau_2 = 0.1$
AS-JADE (Gong et al., 2011)	$c = 0.1$, $p = 0.05$, $\mu_F = 0.5$, $\mu_{CR} = 0.5$, $\alpha = 0.3$, $\beta = 0.8$
MDE_pBX (Islam et al., 2012)	$q = 0.15$, F_i = Cauchy(F_m, 0.1), Cr_i = Gaussian(Cr_m, 0.1)
rank-jDE (Gong & Cai, 2013)	$F_l = 0.1$, $F_u = 0.9$, $\tau_1 = \tau_2 = 0.1$

- OS: Win 7.
- CPU: Intel i5 @2.60 GHz.
- RAM: 8 G.
- Platform: Matlab 2012b.

4.1. Compared algorithms and simulation setup

In order to fairly compare the results between the ARM-based DE algorithm and its corresponding original DE algorithm, we use the following six state-of-the-art algorithms shown in Table 2 with their recommended setup in their original literatures. All peer methods could adapt their F and Cr values during evolution.

All benchmark functions are tested in 30 and 50 dimensions. The population size has been kept equal to 100 irrespective of the problem dimension. The maximum of function evaluations (Max_Fes) is set to 10000D. Calculated the results of different algorithms on each function are averaged over 50 independent runs. In addition, minimum support that is named as Min_sup is a user-defined parameter. According to our previous experimental results, if Min_sup is set too large such as 0.4 on some test functions, there will be no 2-itemset (F and Cr pair) mined and the proposed ARM-based method will never be used. Thus, Min_sup is set to 0.05 to guarantee at least one F and Cr combination to be chosen. Nevertheless, if the test function is too difficult to find any better fitness value, original adaptive DE variants will fail, so do modified ones with the ARM-based strategy. All of the DE variants start evolving from the same initial population in each run so that any difference in their performances may be attributed to their internal search operators only.

4.2. Effect on peer DE algorithms

In this section, we incorporate the ARM-based parameter adaptive strategy into some advanced DE algorithms. The error values of all DE variants are reported in Tables 3 and 4 on CEC05 functions for $D = 30$ and $D = 50$, respectively. All results are averaged over 50 independent runs. In order to compare the significance between the two algorithms, the Wilcoxon rank sum test at 0.05 level is used (Derrac, Garcia, Molina, & Herrera, 2011; Garcia, Fernandez, Luengo, & Herrera, 2010). The Wilcoxon rank sum test results regarding Algorithm 1 vs. Algorithm 2 are shown as '+', '–', '≈', when Algorithm 1 is significantly better than, significantly worse than and significantly equal to Algorithm 2, respectively. In this paper, we use Wilcoxon rank sum test regarding DE algorithm with ARM-based strategy vs. original algorithm. Best results are shown in boldface.

In order to demonstrate the effectiveness of the proposed ARM-based F and Cr adaptive strategy, we insert the proposed strategy into some famous DE variants, resulting in satisfying performances. For fair comparison, all parameters are kept the same as used in their original literatures as shown in Table 3. DE algorithms with the ARM-based strategy beat the corresponding algorithms on 6

Table 3
Compared results of peer DE algorithms and the corresponding method with ARM-based operator for $D = 30$.

	SaDE		SaDE-ARM	JADE		JADE-ARM	jDE		jDE-ARM
f_1	6.43E-30 ± 2.79E-30	+	**2.80E-33 ± 2.09E-33**	1.58E-54 ± 9.36E-54	+	**2.27E-57 ± 9.28E-57**	3.52E-29 ± 4.32E-29	+	**8.93E-31 ± 1.43E-31**
f_2	1.35E-17 ± 9.01E-17	+	**3.36E-21 ± 2.07E-21**	4.46E-28 ± 1.59E-28	+	**2.19E-28 ± 1.17E-28**	1.05E-05 ± 1.45E-05	+	**1.44E-06 ± 2.38E-06**
f_3	**4.51E + 04 ± 3.24E + 04**	≈	4.93E + 04 ± 4.68E + 04	8.18E + 03 ± 5.47E + 03	≈	**7.56E + 03 ± 6.79E + 03**	1.76E + 05 ± 1.02E + 05	+	**8.11E + 04 ± 3.28E + 04**
f_4	6.35E-01 ± 1.94E + 00	+	**3.39E-02 ± 8.33E-01**	7.18E-16 ± 2.56E-15	+	**5.59E-16 ± 1.57E-15**	**2.94E-01 ± 5.62E-01**	–	3.04E-01 ± 6.80E-01
f_5	1.57E + 03 ± 4.00E + 02	+	**1.19E + 03 ± 4.18E + 02**	9.76E-02 ± 2.77E-01	+	**4.55E-02 ± 1.89E-01**	**1.10E + 03 ± 4.34E + 02**	≈	1.13E + 03 ± 5.07E + 02
f_6	**2.01E + 00 ± 1.93E + 00**	≈	2.10E + 00 ± 2.11E + 00	8.33E + 00 ± 2.27E + 01	+	**7.95E-01 ± 3.53E + 00**	**2.49E + 01 ± 2.44E + 01**	≈	2.54E + 01 ± 1.08E + 00
f_7	**1.57E-02 ± 1.51E-02**	–	1.88E-02 ± 2.95E-02	9.44E-03 ± 8.80E-03	+	**6.12E-03 ± 7.21E-03**	1.38E-02 ± 9.49E-03	+	**9.94E-03 ± 8.34E-03**
f_8	2.09E + 01 ± 7.11E-02	≈	2.09E + 01 ± 7.51E-02	2.09E + 01 ± 1.43E-01	≈	2.09E + 01 ± 1.43E-01	2.09E + 01 ± 4.93E-02	≈	2.09E + 01 ± 4.98E-02
f_9	2.37E-15 ± 1.36E-14	+	**3.55E-16 ± 3.54E-15**	5.72E-22 ± 8.43E-22	+	**3.92E-23 ± 1.26E-23**	3.52E-29 ± 4.62E-29	+	**3.82E-30 ± 1.76E-30**
f_{10}	5.18E + 01 ± 6.27E + 00	+	**4.64E + 01 ± 4.59E + 00**	2.56E + 01 ± 4.99E + 00	+	**2.39E + 01 ± 4.35E + 00**	5.86E + 01 ± 1.07E + 01	+	**4.82E + 01 ± 9.19E + 00**
f_{11}	2.95E + 01 ± 5.25E + 00	≈	**2.77E + 01 ± 4.32E + 00**	**2.68E + 01 ± 1.05E + 00**	≈	2.73E + 01 ± 1.74E + 00	2.81E + 01 ± 1.82E + 00	≈	**2.77E + 01 ± 2.52E + 00**
f_{12}	2.20E + 03 ± 1.81E + 03	+	**1.63E + 03 ± 1.82E + 03**	7.23E + 03 ± 3.96E + 03	+	**4.23E + 03 ± 3.59E + 03**	1.08E + 04 ± 7.72E + 03	+	**1.89E + 03 ± 1.84E + 03**
f_{13}	**2.40E + 00 ± 1.97E-01**	≈	2.41E + 00 ± 2.01E-01	**1.44E + 00 ± 1.03E-01**	≈	1.49E + 00 ± 1.18E-01	1.60E + 00 ± 1.55E-01	≈	**1.56E + 00 ± 2.73E-01**
f_{14}	1.29E + 01 ± 1.87E-01	≈	1.29E + 01 ± 1.90E-01	1.24E + 01 ± 3.37E-01	≈	**1.21E + 01 ± 3.34E-01**	**1.31E + 01 ± 2.08E-01**	≈	1.38E + 01 ± 2.48E-01
f_{15}	3.17E + 02 ± 5.28E + 01	≈	**3.01E + 02 ± 4.80E + 01**	3.65E + 02 ± 8.77E + 01	≈	**3.64E + 02 ± 9.98E + 01**	**3.43E + 02 ± 1.18E + 02**	≈	3.63E + 02 ± 5.45E + 01
f_{16}	7.64E + 01 ± 3.10E + 01	+	**6.91E + 01 ± 2.99E + 01**	**7.84E + 01 ± 8.66E + 01**	≈	8.81E + 01 ± 1.07E + 02	7.58E + 01 ± 8.85E + 00	+	**6.01E + 01 ± 9.02E + 00**
f_{17}	1.34E + 02 ± 5.50E + 01	≈	**1.33E + 02 ± 5.76E + 01**	1.23E + 02 ± 9.61E + 01	≈	**1.07E + 02 ± 8.68E + 01**	1.23E + 02 ± 1.47E + 01	+	**1.05E + 02 ± 3.23E + 01**
f_{18}	8.77E + 02 ± 4.44E + 01	≈	**8.75E + 02 ± 4.02E + 01**	**8.85E + 02 ± 3.80E + 01**	≈	9.03E + 02 ± 2.55E + 01	**9.07E + 02 ± 1.34E + 00**	≈	9.08E + 02 ± 2.01E + 00
f_{19}	**8.66E + 02 ± 5.32E + 01**	≈	8.77E + 02 ± 5.26E + 01	**8.98E + 02 ± 3.85E + 01**	≈	9.09E + 02 ± 2.29E + 01	**9.06E + 02 ± 1.71E + 00**	≈	9.07E + 02 ± 1.80E + 00
f_{20}	**7.58E + 02 ± 4.36E + 01**	–	8.95E + 02 ± 4.77E + 01	8.89E + 02 ± 3.94E + 01	≈	**8.87E + 02 ± 3.68E + 01**	9.09E + 02 ± 1.58E + 00	≈	**9.08E + 02 ± 1.78E + 00**
f_{21}	5.05E + 02 ± 4.42E + 01	≈	5.05E + 02 ± 4.42E + 01	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00
f_{22}	9.19E + 02 ± 1.32E + 01	+	**9.09E + 02 ± 1.26E + 02**	8.99E + 02 ± 1.37E + 01	+	**8.88E + 02 ± 1.41E + 01**	9.05E + 02 ± 1.02E + 01	+	**8.87E + 02 ± 1.16E + 01**
f_{23}	5.34E + 02 ± 1.19E-03	≈	5.34E + 02 ± 3.30E-03	5.34E + 02 ± 1.29E-04	≈	5.34E + 02 ± 2.88E-03	5.34E + 02 ± 2.17E-04	≈	5.34E + 02 ± 1.21E-03
f_{24}	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00
f_{25}	2.10E + 02 ± 3.34E-01	≈	**2.09E + 02 ± 2.91E-01**	2.09E + 02 ± 1.22E-01	≈	2.09E + 02 ± 1.02E-01	2.10E + 02 ± 3.29E-01	≈	**2.09E + 02 ± 2.88E-01**
w/t/l	9/14/2		–	10/15/0		–	10/14/1		–
	AS-JADE		AS-JADE-ARM	MDE_*p*BX		MDE_*p*BX-ARM	rank-jDE		rank-jDE-ARM
f_1	2.62E-20 ± 5.34E-20	+	**6.37E-21 ± 1.90E-21**	1.29E-62 ± 1.85E-61	+	**4.17E-64 ± 1.99E-64**	**8.95E-59 ± 4.17E-59**	≈	9.07E-59 ± 2.99E-59
f_2	4.78E-27 ± 1.15E-26	+	**9.23E-28 ± 1.88E-28**	**1.81E-26 ± 2.27E-26**	≈	1.99E-26 ± 2.63E-26	1.44E-11 ± 2.05E-11	+	**2.40E-12 ± 1.35E-12**
f_3	4.79E + 04 ± 2.90E + 04	≈	**4.38E + 04 ± 2.32E + 04**	**2.77E + 03 ± 1.99E + 03**	–	3.50E + 03 ± 2.58E + 03	8.11E + 04 ± 3.83E + 04	≈	**8.09E + 04 ± 5.46E + 04**
f_4	6.97E-16 ± 2.88E-15	+	**4.72E-18 ± 1.34E-18**	**6.86E-08 ± 6.08E-08**	≈	6.94E-08 ± 4.13E-08	7.98E-04 ± 1.56E-03	+	**4.35E-04 ± 1.79E-03**
f_5	**2.14E + 02 ± 1.94E + 02**	–	4.10E + 02 ± 2.64E + 02	2.57E + 02 ± 1.19E + 02	+	**1.13E + 02 ± 1.54E + 02**	**1.12E + 03 ± 5.48E + 02**	≈	1.36E + 03 ± 5.35E + 02
f_6	**6.27E + 01 ± 5.05E + 01**	≈	6.37E + 01 ± 2.24E + 01	3.70E-01 ± 1.98E + 00	≈	**3.56E-01 ± 8.45E-01**	5.74E-01 ± 2.31E + 00	+	**3.44E-01 ± 2.13E + 00**
f_7	2.15E + 02 ± 8.50E + 01	≈	**2.22E + 02 ± 7.06E + 01**	6.72E-03 ± 9.07E-03	+	**2.51E-03 ± 5.66E-03**	9.67E-03 ± 4.57E-03	+	**7.49E-03 ± 2.73E-03**
f_8	2.09E + 01 ± 5.18E-02	≈	2.09E + 01 ± 5.95E-02	2.00E + 01 ± 6.85E-07	≈	2.00E + 01 ± 9.16E-07	2.09E + 01 ± 4.94E-02	≈	2.09E + 01 ± 4.99E-02
f_9	1.96E + 01 ± 3.22E + 00	≈	**1.93E + 01 ± 3.01E + 00**	1.49E-09 ± 3.46E-10	+	**4.06E-10 ± 9.14E-09**	**3.56E-30 ± 3.55E-30**	≈	3.62E-30 ± 3.81E-30
f_{10}	1.49E + 02 ± 4.14E + 01	+	**1.08E + 02 ± 3.45E + 01**	1.90E + 01 ± 8.58E-01	≈	**1.84E + 01 ± 7.37E-01**	4.81E + 01 ± 9.41E + 00	+	**4.31E + 01 ± 8.49E + 00**
f_{11}	**2.45E + 01 ± 1.35E + 01**	≈	3.13E + 01 ± 2.00E + 01	1.90E + 01 ± 6.15E + 00	+	**1.32E + 01 ± 3.22E + 00**	2.76E + 01 ± 2.55E + 00	+	**2.19E + 01 ± 1.13E + 00**
f_{12}	**5.63E + 03 ± 6.58E + 03**	≈	6.08E + 03 ± 6.26E + 03	**1.93E + 03 ± 8.83E + 02**	≈	2.05E + 03 ± 8.45E + 02	**1.56E + 03 ± 2.02E + 03**	≈	1.70E + 03 ± 2.23E + 03
f_{13}	1.19E + 01 ± 2.05E + 00	+	**1.03E + 01 ± 2.02E + 00**	1.51E + 00 ± 5.60E-02	≈	**1.46E + 00 ± 9.85E-02**	1.65E + 00 ± 1.75E-01	≈	**1.55E + 00 ± 1.35E-01**
f_{14}	1.36E + 01 ± 3.10E-01	+	**1.12E + 01 ± 2.99E-01**	**1.29E + 01 ± 3.20E-01**	≈	1.34E + 01 ± 3.09E-01	**1.40E + 01 ± 2.03E-01**	≈	1.51E + 01 ± 1.62E-01
f_{15}	3.45E + 02 ± 8.42E + 01	≈	**3.43E + 02 ± 8.66E + 01**	2.53E + 02 ± 9.42E + 01	≈	**2.47E + 02 ± 8.76E + 01**	3.88E + 02 ± 5.33E + 01	≈	**3.79E + 02 ± 4.18E + 01**
f_{16}	1.54E + 02 ± 1.31E + 02	+	**1.17E + 02 ± 1.18E + 02**	5.27E + 01 ± 3.52E + 00	≈	**5.17E + 01 ± 3.40E + 00**	6.19E + 01 ± 9.20E + 00	+	**4.73E + 01 ± 5.99E + 00**
f_{17}	2.25E + 02 ± 1.01E + 02	≈	**2.22E + 02 ± 1.03E + 02**	8.28E + 01 ± 3.47E + 01	≈	**8.25E + 01 ± 3.45E + 01**	**1.03E + 02 ± 3.60E + 01**	≈	1.17E + 02 ± 3.73E + 01
f_{18}	**8.68E + 02 ± 5.69E + 01**	≈	8.73E + 02 ± 5.13E + 01	**7.35E + 02 ± 1.07E-01**	≈	8.35E + 02 ± 1.18E-01	9.08E + 02 ± 4.64E + 00	≈	**9.00E + 02 ± 4.34E + 00**
f_{19}	**8.69E + 02 ± 5.44E + 01**	≈	8.80E + 02 ± 4.07E + 01	**8.25E + 02 ± 1.90E-01**	≈	8.28E + 02 ± 1.36E-01	**9.08E + 02 ± 1.97E + 00**	≈	9.09E + 02 ± 2.00E + 00
f_{20}	**8.74E + 02 ± 5.32E + 01**	≈	8.90E + 02 ± 4.94E + 01	6.42E + 02 ± 2.17E-01	≈	**6.38E + 02 ± 1.86E-01**	9.08E + 02 ± 1.78E + 00	≈	**8.99E + 02 ± 1.69E + 00**
f_{21}	5.37E + 02 ± 7.10E + 01	≈	**5.28E + 02 ± 8.16E + 01**	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00
f_{22}	9.32E + 02 ± 1.57E + 01	≈	**9.22E + 02 ± 1.26E + 01**	5.21E + 02 ± 4.55E-01	+	**5.00E + 02 ± 3.90E-01**	**8.89E + 02 ± 1.22E + 01**	≈	8.97E + 02 ± 1.33E + 01
f_{23}	5.84E + 02 ± 2.92E + 02	+	**5.57E + 02 ± 2.81E + 02**	5.16E + 02 ± 2.64E-04	≈	**5.08E + 02 ± 2.68E-04**	5.34E + 02 ± 1.21E-03	≈	5.34E + 02 ± 1.21E-03
f_{24}	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00
f_{25}	**2.95E + 02 ± 2.36E + 02**	≈	3.00E + 02 ± 2.47E + 02	**2.62E + 02 ± 2.58E + 00**	≈	2.71E + 02 ± 3.90E + 00	**2.09E + 02 ± 2.88E-01**	≈	2.10E + 02 ± 2.97E-01
w/t/l	8/16/1		–	6/18/1		–	7/18/0		–

Table 4
Compared results of peer DE algorithms and the corresponding method with ARM-based operator for $D = 50$.

	SaDE		SaDE-ARM	JADE		JADE-ARM	jDE		jDE-ARM
f_1	1.72E-11 ± 2.35E-12	+	**4.54E-13 ± 6.93E-13**	7.51E-14 ± 3.68E-14	+	**2.27E-15 ± 4.08E-15**	3.44E-09 ± 4.15E-09	+	**2.27E-11 ± 4.15E-11**
f_2	2.35E-10 ± 3.39E-10	+	**7.61E-13 ± 3.31E-14**	5.31E-04 ± 2.03E-04	≈	**5.10E-04 ± 3.17E-04**	9.09E-02 ± 7.89E-02	+	**6.53E-02 ± 2.38E-02**
f_3	**1.58E + 05 ± 6.27E + 04**	≈	1.77E + 05 ± 5.24E + 04	1.54E + 04 ± 5.49E + 03	≈	**1.47E + 04 ± 4.58E + 03**	5.83E + 05 ± 3.94E + 05	+	**2.85E + 05 ± 3.89E + 05**
f_4	1.07E + 03 ± 1.16E + 03	+	**8.57E + 02 ± 7.04E + 02**	1.93E + 00 ± 5.05E + 00	+	**9.11E-01 ± 2.64E + 00**	**7.90E + 02 ± 7.39E + 02**	≈	8.08E + 02 ± 9.07E + 02
f_5	4.41E + 03 ± 9.26E + 02	≈	**4.10E + 03 ± 9.27E + 02**	**1.68E + 03 ± 4.23E + 02**	≈	1.78E + 03 ± 5.73E + 02	**3.33E + 03 ± 6.23E + 02**	≈	3.63E + 03 ± 4.96E + 02
f_6	5.48E + 00 ± 2.03E + 00	+	**2.55E + 00 ± 1.70E + 00**	2.17E + 00 ± 6.76E + 00	+	**1.05E + 00 ± 5.24E + 00**	3.70E + 01 ± 9.62E + 00	+	**1.55E + 01 ± 8.46E + 00**
f_7	**8.70E + 03 ± 1.59E + 02**	≈	8.86E + 03 ± 1.74E + 02	**3.45E + 03 ± 3.94E + 03**	≈	4.41E + 03 ± 2.47E + 03	**4.15E + 03 ± 7.29E + 03**	≈	4.34E + 03 ± 8.85E + 03
f_8	2.11E + 01 ± 4.23E-02	≈	2.11E + 01 ± 3.53E-02	2.11E + 01 ± 3.74E-02	≈	2.11E + 01 ± 4.16E-02	2.11E + 01 ± 4.05E-02	≈	2.11E + 01 ± 4.79E-02
f_9	1.18E + 02 ± 1.66E + 01	+	**9.71E + 01 ± 5.57E + 00**	5.31E-04 ± 1.76E-06	+	**1.08E-04 ± 5.28E-06**	1.16E + 02 ± 1.09E + 01	+	**6.19E + 01 ± 6.14E + 00**
f_{10}	1.32E + 02 ± 1.45E + 01	≈	**1.19E + 02 ± 1.22E + 00**	6.15E + 01 ± 8.97E + 00	+	**4.98E + 01 ± 7.02E + 00**	1.00E + 02 ± 1.31E + 01	≈	**9.88E + 01 ± 1.25E + 01**
f_{11}	5.39E + 01 ± 3.27E + 00	≈	**5.21E + 01 ± 3.45E + 00**	**5.36E + 02 ± 2.68E + 01**	≈	5.37E + 02 ± 2.52E + 01	5.22E + 01 ± 2.14E + 00	≈	**5.14E + 01 ± 2.32E + 00**
f_{12}	1.06E + 04 ± 9.78E + 03	+	**8.82E + 03 ± 8.90E + 03**	1.77E + 04 ± 2.33E + 04	+	**1.39E + 04 ± 1.90E + 04**	3.98E + 04 ± 2.17E + 04	+	**6.69E + 03 ± 9.57E + 03**
f_{13}	**4.89E + 00 ± 5.30E-01**	≈	4.97E + 00 ± 5.80E-01	**2.50E + 00 ± 3.96E-01**	≈	2.74E + 00 ± 1.42E-01	2.91E + 00 ± 2.13E-01	≈	**2.90E + 00 ± 2.03E-01**
f_{14}	**2.36E + 01 ± 2.28E-01**	≈	2.44E + 01 ± 2.20E-01	**2.16E + 01 ± 4.72E-01**	≈	2.17E + 01 ± 4.15E-01	2.26E + 01 ± 3.93E-01	≈	2.26E + 01 ± 3.99E-01
f_{15}	3.57E + 02 ± 7.14E + 01	≈	**3.38E + 02 ± 7.01E + 01**	**3.05E + 02 ± 9.33E + 01**	≈	3.15E + 02 ± 9.36E + 01	**3.43E + 02 ± 9.02E + 01**	≈	3.44E + 02 ± 9.25E + 01
f_{16}	8.60E + 01 ± 9.82E + 00	+	**7.57E + 01 ± 9.57E + 00**	6.34E + 01 ± 3.79E + 01	+	**5.55E + 01 ± 3.73E + 01**	8.56E + 01 ± 7.92E + 00	+	**6.58E + 01 ± 6.38E + 00**
f_{17}	**1.68E + 02 ± 1.60E + 01**	≈	1.77E + 02 ± 2.05E + 01	1.13E + 02 ± 4.94E + 01	+	**1.05E + 02 ± 3.78E + 01**	1.72E + 02 ± 1.73E + 01	≈	**1.62E + 02 ± 1.42E + 01**
f_{18}	8.87E + 02 ± 6.06E + 01	≈	**8.64E + 02 ± 5.92E + 01**	**9.31E + 02 ± 3.70E + 01**	≈	9.33E + 02 ± 3.83E + 01	**9.09E + 02 ± 3.34E + 00**	−	9.45E + 02 ± 4.02E + 00
f_{19}	**9.53E + 02 ± 3.48E + 01**	≈	9.58E + 02 ± 3.58E + 01	9.37E + 02 ± 1.05E + 01	≈	**9.34E + 02 ± 1.07E + 01**	**9.23E + 02 ± 1.17E + 00**	≈	9.33E + 02 ± 1.74E + 00
f_{20}	9.43E + 02 ± 3.90E + 01	≈	9.47E + 02 ± 2.43E + 01	**9.28E + 02 ± 1.16E + 01**	≈	9.32E + 02 ± 1.20E + 01	**9.28E + 02 ± 3.85E + 00**	≈	9.30E + 02 ± 3.87E + 00
f_{21}	5.49E + 02 ± 1.75E + 02	+	**5.29E + 02 ± 2.04E + 02**	5.18E + 02 ± 7.63E + 01	≈	5.18E + 02 ± 7.63E + 01	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00
f_{22}	**9.67E + 02 ± 7.80E + 00**	≈	9.74E + 02 ± 8.46E + 00	9.42E + 02 ± 1.29E + 01	≈	**9.40E + 02 ± 5.97E + 01**	9.47E + 02 ± 1.29E + 01	≈	**9.38E + 02 ± 1.24E + 01**
f_{23}	5.75E + 02 ± 1.26E + 02	≈	**5.68E + 02 ± 1.22E + 02**	5.47E + 02 ± 6.97E + 01	≈	**5.45E + 02 ± 6.80E + 01**	**5.39E + 02 ± 4.05E-03**	−	5.74E + 02 ± 1.09E + 01
f_{24}	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00
f_{25}	**2.16E + 02 ± 6.17E-01**	≈	2.20E + 02 ± 5.91E-01	2.14E + 02 ± 5.58E-01	≈	2.14E + 02 ± 5.95E-01	2.14E + 02 ± 4.19E-01	≈	2.14E + 02 ± 4.25E-01
w/t/l	8/17/0		−	8/17/0		−	7/16/2		−
	AS-JADE		AS-JADE-ARM	MDE_*p*BX		MDE_*p*BX-ARM	rank-jDE		rank-jDE-ARM
f_1	6.79E-10 ± 9.64E-11	+	**2.27E-11 ± 3.21E-11**	**4.03E-32 ± 8.35E-20**	≈	4.27E-32 ± 8.29E-20	4.54E-13 ± 6.61E-13	≈	**3.15E-13 ± 6.56E-13**
f_2	1.42E-10 ± 3.87E-10	+	**7.69E-11 ± 3.88E-11**	4.46E-06 ± 8.16E-11	+	**1.11E-07 ± 2.91E-10**	3.64E-05 ± 3.97E-05	+	**9.99E-04 ± 2.88E-05**
f_3	2.55E + 05 ± 8.77E + 04	+	**1.82E + 05 ± 7.49E + 04**	**3.38E + 04 ± 2.43E + 04**	≈	3.55E + 04 ± 3.86E + 04	**3.15E + 05 ± 1.42E + 05**	≈	3.17E + 05 ± 1.89E + 05
f_4	7.89E-03 ± 1.65E-02	+	**7.54E-03 ± 1.41E-02**	1.75E + 02 ± 2.57E-06	+	**1.14E + 02 ± 3.03E-06**	2.87E + 02 ± 4.12E + 02	+	**2.31E + 02 ± 4.07E + 02**
f_5	2.59E + 03 ± 5.01E + 02	+	**2.14E + 03 ± 3.94E + 02**	2.58E + 03 ± 1.44E + 02	+	**1.36E + 03 ± 1.32E + 02**	3.67E + 03 ± 5.13E + 02	+	**2.29E + 03 ± 3.03E + 02**
f_6	2.71E + 03 ± 2.18E + 03	+	**6.13E + 02 ± 7.68E + 02**	7.57E-01 ± 1.12E + 00	+	**4.58E-01 ± 1.01E + 00**	7.56E + 00 ± 1.53E + 01	+	**4.54E + 00 ± 1.25E + 01**
f_7	**6.14E + 02 ± 2.66E + 02**	−	8.65E + 02 ± 3.12E + 02	**6.35E + 03 ± 2.12E + 00**	≈	6.48E + 03 ± 3.52E + 00	4.82E + 03 ± 2.18E + 03	+	**4.14E + 03 ± 2.08E + 03**
f_8	2.11E + 01 ± 3.78E-02	≈	2.11E + 01 ± 4.54E-02	**2.02E + 01 ± 2.70E-02**	−	2.11E + 01 ± 3.94E-02	2.12E + 01 ± 4.26E-02	≈	**2.11E + 01 ± 4.92E-02**
f_9	1.66E + 02 ± 1.39E + 01	≈	**1.64E + 02 ± 1.49E + 01**	1.47E + 02 ± 1.43E + 01	≈	**1.42E + 02 ± 1.55E + 01**	**1.16E + 02 ± 1.95E + 01**	−	1.87E + 02 ± 9.09E + 00
f_{10}	3.46E + 02 ± 5.24E + 01	+	**2.15E + 02 ± 4.83E + 01**	3.18E + 01 ± 1.82E + 01	≈	**1.98E + 01 ± 1.33E + 01**	7.66E + 01 ± 1.52E + 01	≈	**7.58E + 01 ± 1.40E + 01**
f_{11}	**7.02E + 01 ± 1.22E + 00**	≈	7.07E + 01 ± 1.47E + 00	4.28E + 01 ± 1.50E + 00	+	**2.29E + 01 ± 1.07E + 00**	5.31E + 01 ± 2.97E + 00	+	**3.50E + 01 ± 2.27E + 00**
f_{12}	4.24E + 04 ± 3.36E + 04	+	**3.06E + 04 ± 3.04E + 04**	1.79E + 04 ± 8.02E + 03	+	**1.24E + 04 ± 5.58E + 03**	6.02E + 03 ± 3.87E + 03	+	**3.72E + 03 ± 3.53E + 03**
f_{13}	**2.22E + 01 ± 4.91E + 00**	≈	2.26E + 01 ± 5.81E + 00	**2.23E + 01 ± 1.52E + 00**	≈	2.28E + 02 ± 1.62E + 00	**2.81E + 00 ± 3.65E-01**	≈	3.00E + 00 ± 4.89E-01
f_{14}	2.24E + 01 ± 2.28E-01	≈	**2.23E + 01 ± 2.92E-01**	**2.20E + 01 ± 2.01E-01**	≈	2.24E + 01 ± 3.33E-01	2.26E + 01 ± 4.54E-02	≈	2.26E + 01 ± 3.74E-02
f_{15}	**3.09E + 02 ± 8.95E + 01**	−	3.36E + 02 ± 9.52E + 01	3.70E + 02 ± 6.95E + 01	+	**2.50E + 02 ± 3.16E + 01**	3.12E + 02 ± 9.13E + 01	+	**2.11E + 02 ± 6.32E + 01**
f_{16}	**2.64E + 02 ± 3.64E + 01**	≈	2.65E + 02 ± 3.41E + 01	**1.58E + 02 ± 3.88E + 01**	≈	1.60E + 02 ± 4.05E + 01	6.83E + 01 ± 1.74E + 01	≈	**6.69E + 01 ± 3.45E + 01**
f_{17}	**2.54E + 02 ± 6.29E + 01**	−	2.81E + 02 ± 7.74E + 01	**1.02E + 02 ± 1.24E + 01**	≈	1.10E + 02 ± 1.40E + 01	**1.43E + 02 ± 2.94E + 01**	≈	1.53E + 02 ± 4.20E + 01
f_{18}	9.21E + 02 ± 4.60E + 01	≈	**9.15E + 02 ± 3.69E + 01**	8.40E + 02 ± 8.17E + 01	+	**6.59E + 02 ± 5.72E + 01**	9.33E + 02 ± 4.38E + 00	≈	**9.21E + 02 ± 5.37E + 00**
f_{19}	9.27E + 02 ± 2.87E + 01	≈	9.27E + 02 ± 2.93E + 01	9.46E + 02 ± 2.43E + 01	≈	**9.37E + 02 ± 1.78E + 01**	**9.32E + 02 ± 4.51E + 00**	≈	9.39E + 02 ± 5.16E + 00
f_{20}	9.28E + 02 ± 4.07E + 01	≈	**9.20E + 02 ± 3.47E + 01**	**9.04E + 02 ± 2.76E + 01**	≈	9.18E + 02 ± 2.82E + 01	**9.25E + 02 ± 8.03E + 00**	−	9.35E + 02 ± 8.92E + 00
f_{21}	5.28E + 02 ± 9.09E + 01	≈	**5.23E + 02 ± 9.35E + 01**	**5.00E + 02 ± 0.00E + 00**	≈	5.04E + 02 ± 4.56E-03	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00
f_{22}	9.79E + 02 ± 1.91E + 01	≈	**9.76E + 02 ± 1.82E + 01**	9.18E + 02 ± 9.01E + 00	+	**8.86E + 02 ± 7.82E + 00**	9.43E + 02 ± 6.12E + 01	≈	**9.35E + 02 ± 6.32E + 01**
f_{23}	6.44E + 02 ± 1.74E + 02	+	**5.80E + 02 ± 1.60E + 02**	5.00E + 00 ± 0.00E + 00	≈	5.00E + 00 ± 0.00E + 00	5.38E + 02 ± 8.29E-03	≈	**5.20E + 02 ± 8.05E-03**
f_{24}	2.10E + 02 ± 6.83E + 00	+	**2.02E + 02 ± 6.36E + 00**	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00
f_{25}	8.85E + 02 ± 2.97E + 02	+	**7.44E + 02 ± 3.68E + 02**	**9.56E + 02 ± 6.76E + 00**	≈	9.66E + 02 ± 6.83E + 00	2.14E + 02 ± 8.29E-01	≈	2.14E + 02 ± 9.11E-01
w/t/l	11/11/3		−	9/15/1		−	8/15/2		−

Table 5
Compared results and Friedman test of JADE-ARM with different population sizes for $D = 30$.

	$NP = 50$	$NP = 100$	$NP = 150$	$NP = 200$	p-value
f_1	2.13E-45 ± 2.79E-45	2.27E-57 ± 9.28E-57	1.58E-57 ± 8.36E-57	**5.66E-58 ± 4.17E-58**	1.56E-07
f_2	5.35E-28 ± 301E-28	2.19E-28 ± 1.17E-28	1.46E-29 ± 3.59E-29	**3.19E-30 ± 8.17E-30**	0.0019
f_3	8.51E + 03 ± 6.24E + 03	7.56E + 03 ± 6.79E + 03	3.18E + 03 ± 2.47E + 03	**1.56E + 02 ± 7.79E + 02**	0.0004
f_4	1.35E-04 ± 3.94E-04	5.59E-16 ± 1.57E-15	9.18E-21 ± 2.56E-20	**5.79E-25 ± 2.57E-24**	1.94E-10
f_5	7.57E-01 ± 1.00E + 00	4.55E-02 ± 1.89E-01	1.76E-06 ± 5.77E-06	**3.55E-07 ± 1.89E-07**	6.18E-09
f_6	1.11E + 00 ± 2.13E + 00	**7.95E-01 ± 3.53E + 00**	8.93E-01 ± 4.27E + 00	9.95E-01 ± 4.53E + 00	0.0055
f_7	1.49E-02 ± 1.01E-02	**6.12E-03 ± 7.21E-03**	8.44E-03 ± 7.80E-03	7.12E-03 ± 7.21E-03	0.0078
f_8	2.09E + 01 ± 1.11E-02	2.09E + 01 ± 1.43E-01	2.09E + 01 ± 3.07E-01	2.09E + 01 ± 4.93E-01	0.6815
f_9	3.37E-15 ± 7.36E-14	3.92E-23 ± 1.26E-23	8.72E-25 ± 5.43E-25	**2.39E-27 ± 2.16E-27**	2.77E-06
f_{10}	6.18E + 01 ± 6.27E + 00	**2.39E + 01 ± 4.35E + 00**	2.65E + 01 ± 5.01E + 00	2.49E + 01 ± 4.53E + 00	0.0026
f_{11}	2.95E + 01 ± 5.25E + 00	2.73E + 01 ± 1.74E + 00	**2.68E + 01 ± 1.05E + 00**	2.82E + 01 ± 1.64E + 00	0.0578
f_{12}	5.20E + 03 ± 3.81E + 03	4.23E + 03 ± 3.59E + 03	2.93E + 03 ± 2.96E + 03	**2.23E + 03 ± 1.59E + 03**	0.0020
f_{13}	**1.20E + 00 ± 1.97E-01**	1.49E + 00 ± 1.18E-01	2.44E + 00 ± 2.03E-01	2.49E + 00 ± 2.18E-01	0.0169
f_{14}	1.31E + 01 ± 1.77E-01	**1.21E + 01 ± 3.34E-01**	1.24E + 01 ± 3.44E-01	1.24E + 01 ± 3.42E-01	0.0509
f_{15}	**3.16E + 02 ± 4.28E + 01**	3.64E + 02 ± 9.98E + 01	3.67E + 02 ± 7.77E + 01	3.84E + 02 ± 8.98E + 01	0.0081
f_{16}	1.64E + 02 ± 1.10E + 02	**8.81E + 01 ± 1.07E + 02**	1.84E + 02 ± 9.66E + 02	1.31E + 02 ± 5.07E + 02	0.0003
f_{17}	2.34E + 02 ± 8.50E + 01	**1.07E + 02 ± 8.68E + 01**	3.23E + 02 ± 8.61E + 01	4.07E + 02 ± 9.68E + 01	0.0348
f_{18}	9.07E + 02 ± 3.44E + 01	**9.03E + 02 ± 2.55E + 01**	9.05E + 02 ± 2.80E + 01	9.05E + 02 ± 2.75E + 01	0.3855
f_{19}	9.16E + 02 ± 5.32E + 01	9.09E + 02 ± 2.29E + 01	9.09E + 02 ± 2.85E + 01	9.09E + 02 ± 2.22E + 01	0.2219
f_{20}	8.90E + 02 ± 3.76E + 01	**8.87E + 02 ± 3.68E + 01**	8.89E + 02 ± 3.64E + 01	8.88E + 02 ± 3.69E + 01	0.3452
f_{21}	5.13E + 02 ± 4.42E + 01	5.00E + 02 ± 0.00E + 00	5.00E + 02 ± 9.71E-14	5.00E + 02 ± 8.12E-14	0.0447
f_{22}	9.19E + 02 ± 1.32E + 01	**8.88E + 02 ± 1.41E + 01**	8.99E + 02 ± 1.37E + 01	8.90E + 02 ± 1.41E + 01	0.4778
f_{23}	5.34E + 02 ± 1.19E-03	5.34E + 02 ± 2.88E-03	5.34E + 02 ± 1.29E-04	5.34E + 02 ± 2.88E-03	0.7591
f_{24}	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	2.00E + 02 ± 0.00E + 00	0.7743
f_{25}	2.10E + 02 ± 3.34E-01	2.09E + 02 ± 1.02E-01	2.09E + 02 ± 1.22E-01	2.09E + 02 ± 1.02E-01	0.4162
Average ranking	7.08	**4.52**	5.42	4.94	

to 10 benchmark functions. Meanwhile, few methods with ARM-based strategy deteriorate the performances on very few test functions. For example, SaDE-ARM outperforms SaDE on 9 test functions (f_1, f_2, f_4, f_5, f_9, f_{10}, f_{12}, f_{16}, f_{22}), and SaDE-ARM is defeated on 2 test functions (f_7, f_{20}). On the other problems (14 out of 25 cases), there is no significant difference between SaDE-ARM and SaDE. Furthermore, for unimodal functions (f_1-f_5), SaDE-ARM beats SaDE in most cases (f_1, f_2, f_4, f_5). For multimodal functions (f_6-f_{14}), SaDE-ARM is significantly better that SaDE on 3 test functions (f_9, f_{10}, f_{12}). Meanwhile, SaDE-ARM is significantly worse that SaDE on f_7. For composition functions (f_{15}-f_{25}), there are no significant differences between the two algorithms in most cases. Other algorithms also show similar performances. Thus, we may conclude that, the proposed ARM-based strategy could keep searching capabilities of original DE variants in most cases, and improve performances on some optimization problems. Table 4 indicates similar demonstration with Table 3 for $D = 50$. In general, the proposed ARM-based parameter adaptive strategy may improve performances of the original algorithms to some extent.

Moreover, it may be necessary to illustrate two issues. One is the frequency in which ARM-based trial vectors are selected over the generated trial vectors using the initial approach. The other one is when these selections occur. Generally speaking, both of them depend on the optimization problems. For some simple unimodal functions, we observed that the ARM-based trail vectors were chosen with higher frequency. While, for some complicated multi-modal functions, it seemed that the ARM-based trail vectors were chosen with lower frequency. In addition, these selections occurred mainly at early stage in fast stagnant situation, and throughout the evolution in continuous exploration situation, respectively. Thus, it seems that timings when these selections occur vary with different optimization problems.

4.3. Influence of population size

In the previous section, we investigated performances of some advanced DE algorithms with ARM-based F and Cr adaptive strategy on CEC05 test functions. As we claimed before, this methodology could be used for adapting any control parameter in DE, such as probability of each candidate mutation strategy and population size. In order to make the scope of this paper focused, we try to study the influence of population size on the proposed strategy in this part, leaving adaption of population size as a significant work in the future. Here, we choose JADE-ARM as the representative with different population sizes (50, 100, 150, 200) (Gong & Cai, 2013; Yi, Zhou, Gao, Li, & Mou, 2016; Zheng, Zhang, Tang, & Zheng, 2017). Over 50 independent runs are conducted on CEC05 test functions for $D = 30$. Like the previous experiment, Max_Fes is set to 10000D. The experimental results are displayed in Table 5. The boldfaces indicate the best results that are achieved among the population size settings. Moreover, in order to show the overall rankings of JADE-ARM with different population sizes, Friedman test (Derrac et al., 2011; Garcia et al., 2010) is conducted at $\alpha = 0.05$ in Matlab environment. The p-values and average rankings for all functions are shown in Table 5.

From Table 5 we can see, for unimodal functions JADE-ARM with $NP = 200$ performs better than other settings on all unimodal functions (f_1, f_2, f_3, f_4, f_5). While, for multimodal and hybrid composition functions JADE-ARM with $NP = 100$ could achieve the most promising results among the settings in 5 out of 20 cases (f_6, f_7, f_{10}, f_{16}, f_{17}). Also, JADE-ARM obtains the best average ranking. Nevertheless, from the p-values of Friedman tests on all test functions, we would generally say that there are no significant differences among all the four population size settings. It seems that the proposed ARM-based parameter adaptive strategy is not sensitive to population size. Therefore, in this paper, it is acceptable that population size is set to 100.

4.4. Influence of different discretized ranges for F and Cr

DE algorithms with different couples of F and Cr may result in distinctive searching behaviors. In this paper, the two parameters are categorized into five different intervals ((0, 0.1), [0.1, 0.4), [0.4, 0.6), [0.6, 0.9), [0.9, 1]). Why do we use such discretized ranges of F and Cr for the proposed ARM-based strategy? It is necessary to investigate the influence of different discretized ranges for F and Cr. In this part, we choose jDE-ARM as the paradigm to execute com-

Table 6
Parameter settings of jDE-ARMs.

Algorithm	*F*	*Cr*
jDE-ARM-1	(0, 0.1), [0.1, 0.2), [0.2, 0.3), [0.3, 0.4), [0.4, 0.5), [0.5, 0.6), [0.6, 0.7), [0.7, 0.8), [0.8, 0.9), [0.9, 1]	(0, 0.1), [0.1, 0.2), [0.2, 0.3), [0.3, 0.4), [0.4, 0.5), [0.5, 0.6), [0.6, 0.7), [0.7, 0.8), [0.8, 0.9), [0.9, 1]
jDE-ARM-2	(0, 0.5), [0.5, 1]	(0, 0.5), [0.5, 1]
jDE-ARM-3	(0, 0.4), [0.4, 0.6), [0.6, 1]	(0, 0.4), [0.4, 0.6), [0.6, 1]
jDE-ARM-4	(0, 0.1), [0.1, 0.9), [0.9, 1]	(0, 0.1), [0.1, 0.9), [0.9, 1]
jDE-ARM	(0, 0.1), [0.1, 0.4), [0.4, 0.6), [0.6, 0.9), [0.9, 1]	(0, 0.1), [0.1, 0.4), [0.4, 0.6), [0.6, 0.9), [0.9, 1]

Table 7
Compared results of jDE-ARM with different *F* and *Cr* discretized ranges for $D = 30$.

	jDE-ARM-1		jDE-ARM-2		jDE-ARM-3		jDE-ARM-4		jDE-ARM
f_1	5.43E-27 ± 8.79E-27	+	7.84E-25 ± 3.66E-25	+	7.19E-26 ± 4.53E-26	+	**8.77E-31 ± 1.28E-31**	≈	8.93E-31 ± 1.43E-31
f_2	**6.27E-07 ± 5.86E-07**	–	1.68E-04 ± 9.45E-04	+	2.86E-05 ± 4.77E-05	+	4.89E-04 ± 1.22E-04	+	1.44E-06 ± 2.38E-06
f_3	1.08E + 05 ± 6.01E + 05	+	5.77E + 05 ± 4.23E + 05	+	3.09E + 06 ± 4.75E + 06	+	6.68E + 05 ± 8.20E + 05	+	**8.11E + 04 ± 3.28E + 04**
f_4	8.53E + 00 ± 4.94E + 00	+	1.85E + 00 ± 3.35E + 00	+	4.18E-01 ± 7.56E-01	≈	4.06E + 01 ± 9.85E + 01	+	**3.04E-01 ± 6.80E-01**
f_5	1.27E + 03 ± 4.00E + 02	≈	1.94E + 04 ± 9.08E + 04	+	5.51E + 04 ± 8.06E + 04	+	2.55E + 04 ± 6.89E + 04	+	**1.13E + 03 ± 5.07E + 02**
f_6	7.71E + 01 ± 3.93E + 01	+	8.19E + 01 ± 1.88E + 01	+	9.03E + 01 ± 1.27E + 01	+	5.97E + 02 ± 3.55E + 02	+	**2.54E + 01 ± 1.08E + 00**
f_7	4.96E-02 ± 8.02E-02	+	7.53E-02 ± 1.95E-02	+	4.66E-01 ± 7.70E-01	+	2.66E-02 ± 7.21E-02	+	**9.94E-03 ± 8.34E-03**
f_8	2.09E + 01 ± 6.11E-02	≈	2.09E + 01 ± 6.51E-02	≈	2.09E + 01 ± 5.43E-01	≈	2.09E + 01 ± 4.43E-01	≈	2.09E + 01 ± 4.98E-02
f_9	5.37E-25 ± 6.36E-24	+	**3.65E-30 ± 1.54E-30**	≈	5.72E-22 ± 8.43E-22	+	3.92E-30 ± 1.96E-30	≈	3.82E-30 ± 1.76E-30
f_{10}	2.18E + 02 ± 2.27E + 02	+	1.64E + 02 ± 1.59E + 02	+	8.56E + 01 ± 4.99E + 01	+	7.39E + 01 ± 4.35E + 01	+	**4.82E + 01 ± 9.19E + 00**
f_{11}	2.91E + 01 ± 1.25E + 00	≈	2.77E + 01 ± 1.34E + 00	≈	**2.69E + 01 ± 1.15E + 00**	≈	2.70E + 01 ± 1.72E + 00	≈	2.77E + 01 ± 2.52E + 00
f_{12}	6.20E + 03 ± 3.81E + 03	+	5.63E + 03 ± 4.82E + 03	+	7.23E + 03 ± 3.96E + 03	+	4.23E + 03 ± 3.59E + 03	+	**1.89E + 03 ± 1.84E + 03**
f_{13}	**1.50E + 00 ± 1.97E-01**	≈	1.51E + 00 ± 2.01E-01	≈	1.54E + 00 ± 1.03E-01	≈	1.49E + 00 ± 1.18E-01	≈	1.56E + 00 ± 2.73E-01
f_{14}	2.39E + 01 ± 2.97E-01	≈	6.29E + 01 ± 9.90E-01	+	7.24E + 01 ± 5.37E-01	+	3.21E + 01 ± 5.34E-01	+	**1.38E + 01 ± 2.48E-01**
f_{15}	3.57E + 02 ± 5.08E + 01	≈	**1.01E + 02 ± 4.80E + 01**	–	1.65E + 02 ± 3.77E + 01	–	3.69E + 02 ± 8.98E + 01	≈	3.63E + 02 ± 5.45E + 01
f_{16}	8.64E + 01 ± 2.10E + 01	+	9.91E + 01 ± 2.99E + 01	+	9.84E + 01 ± 3.66E + 01	+	8.81E + 01 ± 5.07E + 01	+	**6.01E + 01 ± 9.02E + 00**
f_{17}	3.34E + 02 ± 7.50E + 01	+	4.33E + 02 ± 8.76E + 01	+	3.23E + 02 ± 7.61E + 01	+	4.07E + 02 ± 6.68E + 01	+	**1.05E + 02 ± 3.23E + 01**
f_{18}	9.04E + 02 ± 4.44E + 01	≈	9.04E + 02 ± 4.02E + 01	≈	9.06E + 02 ± 3.80E + 01	≈	**9.03E + 02 ± 2.55E + 01**	≈	9.08E + 02 ± 2.01E + 00
f_{19}	9.08E + 02 ± 4.32E + 01	≈	9.09E + 02 ± 5.26E + 01	≈	9.08E + 02 ± 4.85E + 01	≈	9.09E + 02 ± 4.29E + 01	≈	**9.07E + 02 ± 1.80E + 00**
f_{20}	9.08E + 02 ± 7.36E + 01	≈	9.08E + 02 ± 8.77E + 01	≈	9.08E + 02 ± 7.94E + 01	≈	9.08E + 02 ± 8.68E + 01	≈	9.08E + 02 ± 1.78E + 01
f_{21}	5.05E + 02 ± 1.22E + 01	≈	5.05E + 02 ± 2.42E + 01	≈	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00	≈	5.00E + 02 ± 0.00E + 00
f_{22}	9.59E + 02 ± 1.32E + 01	+	9.69E + 02 ± 1.26E + 02	+	9.99E + 02 ± 1.37E + 01	+	9.88E + 02 ± 1.41E + 01	+	**8.87E + 02 ± 1.16E + 01**
f_{23}	5.34E + 02 ± 1.29E-03	≈	5.34E + 02 ± 3.20E-03	≈	5.34E + 02 ± 1.39E-04	≈	5.34E + 02 ± 2.08E-03	≈	5.34E + 02 ± 1.21E-03
f_{24}	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00	≈	2.00E + 02 ± 0.00E + 00
f_{25}	2.10E + 02 ± 2.34E-01	≈	2.09E + 02 ± 2.61E-01	≈	2.09E + 02 ± 2.03E-01	≈	2.09E + 02 ± 2.14E-01	≈	2.09E + 02 ± 2.88E-01
w/t/l	11/13/1		13/11/1		13/11/1		12/13/0		–

pared experiments. Different parameters settings that may obtain better results and their corresponding names are listed in Table 6.

Over 50 independent runs are conducted on CEC05 test functions for $D = 30$. Like the previous experiment, Max_FEs is set to 10000*D*. Population size is set to 100. The experimental results are displayed in Table 7. The boldface indicates the best results that are achieved among different settings. Moreover, In order to compare the significance between two algorithms, the Wilcoxon rank sum test at 0.05 level is used (Derrac et al., 2011; Garcia et al., 2010).

Compared with other *F* and *Cr* discretized ranges, jDE-ARM with the proposed five intervals could beat other compositions on almost half of total benchmark functions. Meanwhile, there are no significant differences among different *F* and *Cr* settings on other half of test functions. On very few test functions, jDE-ARM with the proposed discretization performs worse than other three algorithms. In general, the proposed discretized intervals might be acceptable for ARM-based *F* and *Cr* adaptive strategy in this paper.

4.5. Analysis of extra time cost

For the proposed ARM-based parameter adaptive strategy, it may be a major drawback that Apriori-2 algorithm requires many scans of dataset, resulting in extra CPU time cost. While, in this paper, the authors believe that the additional time cost might be negligible. There are two reasons: (1) The Apriori algorithm is very time consuming because it requires repeated scans of the database during the generation of frequent itemsets. The database is usually very large. In this paper, as the population evolves, the new successful combinations of parameters will be less and less due to convergence and stagnation, resulting in a relatively small dataset. Moreover, according to our previous experiments, dataset that stores successful *F* and *Cr* values increase fast in early stage, while almost keeps its size in later stage. Thus, the Apriori algorithm only increases negligible CPU time. (2) It is widely known that most of time cost is associated with the calculation of the fitness value, and not with the calculations/selections of the control parameters. That is to say, the presented ARM-based strategy does not increase calculation time in general.

5. Experimental results on real-world problems

In this part, we try to evaluate the performances of ARM-based strategy on over 15 real-world optimization problems, which come from CEC2011 problems. These problems can be used to evaluate the performance of different stochastic optimization algorithms, such as some state-of-the-art DE variants with our proposed ARM-based parameter adaptive strategy. All the problem definitions and evaluation criteria of CEC2011 could be found by visiting http://www3.ntu.edu.sg/home/epnsugan/. In order to fairly compare the results between ARM-based and its corresponding original DE variant, we use the same state-of-the-art algorithms shown in Section 4.2 with their recommended setup in the original literatures. The population size has been kept equal to 100. Max_Fes is set to 50,000. Calculated the results of different algorithms on each function are averaged over 25 independent runs. In order to compare the significance between two algorithms, the Wilcoxon rank sum test at 0.05 level is used. The average errors

Table 8
Compared results of DE algorithms and the corresponding methods with ARM-based strategy on CEC11 problems.

	SaDE		SaDE-ARM	JADE		JADE-ARM	jDE		jDE-ARM
T01FM	1.02E + 00 ± 2.06E + 00	≈	1.10E + 00 ± 3.06E + 00	1.77E + 00 ± 3.54E + 00	≈	1.56E + 00 ± 3.11E + 00	1.47E + 00 ± 2.32E + 00	+	**9.87E-01 ± 2.43E + 00**
T02L-J	−1.45E + 01 ± 1.11E + 00	≈	−1.36E + 01 ± 1.07E + 00	−2.46E + 01 ± 1.29E + 00	≈	−1.99E + 01 ± 3.54E + 00	−1.04E + 01 ± 3.47E + 00	+	**−2.56E + 01 ± 2.38E + 00**
T03BCB	−1.15E-05 ± 0.00E + 00	≈	−1.15E-05 ± 0.00E + 00	−1.15E-05 ± 0.00E + 00	≈	−1.15E-05 ± 0.00E + 00	−1.15E-05 ± 0.00E + 00	≈	−1.15E-05 ± 0.00E + 00
T04STR	1.48E + 01 ± 8.34E + 00	≈	1.47E + 01 ± 8.33E + 00	1.39E + 01 ± 8.56E + 00	+	**1.21E + 01 ± 6.57E + 00**	1.47E + 01 ± 8.33E + 00	–	2.02E + 01 ± 9.73E + 00
T05Si(B)	−3.01E + 01 ± 1.35E + 00	≈	−3.06E + 01 ± 1.44E + 00	−3.12E + 01 ± 9.88E-01	+	−3.46E + 01 ± 1.40E + 00	−3.09E + 01 ± 9.79E-01	+	**−4.10E + 01 ± 1.00E + 00**
T06Si©	−2.31E + 01 ± 1.63E + 00	+	−3.10E + 01 ± 2.11E + 00	−2.76E + 01 ± 1.27E + 00	+	**−3.19E + 01 ± 2.25E + 00**	−2.30E + 01 ± 2.94E + 00	+	−2.94E + 01 ± 2.38E + 00
T07SPRP	1.44E + 00 ± 1.42E-01	–	1.88E + 00 ± 3.95E-01	1.27E + 00 ± 1.08E-01	+	6.32E-01 ± 7.21E-01	8.11E-01 ± 2.49E-01	+	5.94E-01 ± 3.34E-01
T08TNEP	2.20E + 02 ± 0.00E + 00	≈	2.20E + 02 ± 0.00E + 00	2.20E + 02 ± 0.00E + 00	≈	2.20E + 02 ± 0.00E + 00	2.20E + 02 ± 0.00E + 00	≈	2.20E + 02 ± 0.00E + 00
T09LSTP	2.37E + 04 ± 1.16E + 04	+	5.68E + 03 ± 8.16E + 03	4.72E + 03 ± 8.34E + 03	+	1.92E + 03 ± 7.63E + 03	4.57E + 03 ± 7.33E + 03	+	1.79E + 03 ± 5.69E + 03
T10CAAD	−2.15E + 01 ± 6.27E-01	≈	−2.14E + 01 ± 3.27E-01	−2.15E + 01 ± 1.09E-01	≈	−2.16E + 01 ± 3.75E-01	−2.15E + 01 ± 4.47E-01	≈	−2.15E + 01 ± 7.03E-01
T11.1DED	5.83E + 04 ± 1.29E + 04	+	2.77E + 04 ± 1.47E + 04	5.32E + 04 ± 1.04E + 04	+	2.63E + 04 ± 2.59E + 04	5.25E + 04 ± 1.82E + 04	+	2.11E + 04 ± 3.25E + 04
T11.3ELD	1.54E + 04 ± 1.81E + 00	≈	1.54E + 04 ± 1.82E + 03	1.54E + 04 ± 3.96E + 00	≈	1.54E + 04 ± 3.59E + 00	1.54E + 04 ± 7.72E-01	≈	1.54E + 04 ± 1.84E + 00
T11.8HS	9.43E + 05 ± 5.58E + 03	≈	9.44E + 05 ± 4.01E + 03	9.45E + 05 ± 6.03E + 03	≈	9.45E + 05 ± 5.67E + 03	9.44E + 05 ± 3.55E + 03	≈	9.43E + 05 ± 4.73E + 03
T12(me)	2.09E + 01 ± 1.87E + 00	+	**1.38E + 01 ± 1.70E + 00**	1.88E + 01 ± 1.37E + 00	≈	1.88E + 01 ± 1.34E + 00	1.61E + 01 ± 2.08E + 00	≈	1.58E + 01 ± 2.49E + 00
T13(Ca)	2.17E + 01 ± 2.28E + 00	+	1.41E + 01 ± 2.80E + 00	1.96E + 01 ± 3.77E + 00	+	1.54E + 01 ± 2.98E + 00	1.63E + 01 ± 3.18E + 00	+	**1.02E + 01 ± 2.45E + 00**
w/t/l	5/9/1		–	7/8/0		–	8/6/1		–
	AS-JADE		AS-JADE-ARM	MDE_*p*BX		MDE_*p*BX-ARM	rank-jDE		rank-jDE-ARM
T01FM	3.79E + 00 ± 5.57E + 00	≈	3.82E + 00 ± 5.66E + 00	3.51E + 00 ± 2.14E + 00	≈	3.44E + 00 ± 2.08E + 00	1.33E + 00 ± 2.22E + 00	≈	1.27E + 00 ± 2.002E + 00
T02L-J	−1.27E + 01 ± 1.95E + 00	≈	−1.40E + 01 ± 2.03E + 00	−1.11E + 01 ± 2.17E + 00	–	−0.37E + 01 ± 3.87E + 00	−1.14E + 01 ± 2.88E + 00	≈	−1.35E + 01 ± 3.64E + 00
T03BCB	−1.15E-05 ± 0.00E + 00	≈	−1.15E-05 ± 0.00E + 00	−1.15E-05 ± 0.00E + 00	≈	−1.15E-05 ± 0.00E + 00	−1.15E-05 ± 0.00E + 00	≈	−1.15E-05 ± 0.00E + 00
T04STR	1.47E + 01 ± 1.69E + 00	≈	1.47E + 01 ± 1.73E + 00	1.47E + 01 ± 5.83E + 00	≈	1.47E + 01 ± 6.71E + 00	1.54E + 01 ± 6.56E + 00	≈	1.60E + 01 ± 5.77E + 00
T05Si(B)	−3.04E + 01 ± 1.94E + 00	–	−3.31E + 01 ± 2.54E + 00	−2.41E + 01 ± 1.09E + 00	+	−3.11E + 01 ± 9.92E-01	−3.13E + 01 ± 1.11E + 00	+	−3.09E + 01 ± 1.08E + 00
T06Si©	−2.18E + 01 ± 1.95E + 00	+	−2.65E + 01 ± 3.15E + 00	−2.06E + 01 ± 2.95E + 00	+	−2.76E + 01 ± 2.45E + 00	−2.30E + 01 ± 3.03E + 00	+	−2.73E + 01 ± 4.11E + 00
T07SPRP	1.55E + 00 ± 1.20E-01	+	1.02E + 00 ± 3.06E-01	1.22E + 00 ± 2.07E-01	≈	1.41E + 00 ± 3.66E-01	9.22E-01 ± 3.49E-01	+	**5.39E-01 ± 2.34E-01**
T08TNEP	2.20E + 02 ± 0.00E + 00	≈	2.20E + 02 ± 0.00E + 00	2.20E + 02 ± 0.00E + 00	≈	2.20E + 02 ± 0.00E + 00	2.20E + 02 ± 0.00E + 00	≈	2.20E + 02 ± 0.00E + 00
T09LSTP	2.88E + 03 ± 6.22E + 03	+	**1.03E + 03 ± 3.01E + 03**	2.49E + 03 ± 4.46E + 03	+	1.06E + 03 ± 4.14E + 03	5.71E + 03 ± 6.03E + 03	+	3.62E + 03 ± 5.89E + 03
T10CAAD	−2.15E + 01 ± 2.66E-01	≈	−2.14E + 01 ± 7.19E-01	−2.15E + 01 ± 5.05E-01	≈	−2.15E + 01 ± 6.77E-01	−2.15E + 01 ± 3.66E-01	≈	−2.16E + 01 ± 4.27E-01
T11.1DED	5.76E + 04 ± 3.54E + 04	+	1.99E + 04 ± 2.00E + 04	3.21E + 05 ± 1.25E + 05	+	8.77E + 04 ± 1.32E + 05	5.25E + 04 ± 1.39E + 04	+	**9.87E + 03 ± 5.77E + 03**
T11.3ELD	1.54E + 04 ± 6.58E-01	≈	1.54E + 04 ± 8.55E-01	1.54E + 04 ± 8.83E-01	≈	1.54E + 04 ± 9.45E-01	1.54E + 04 ± 2.02E + 00	≈	1.54E + 04 ± 2.23E + 00
T11.8HS	9.43E + 05 ± 5.05E + 03	≈	9.44E + 05 ± 6.02E + 03	9.44E + 05 ± 5.60E + 03	≈	9.44E + 05 ± 7.85E + 03	9.44E + 05 ± 3.75E + 03	≈	9.43E + 05 ± 3.35E + 03
T12(me)	1.88E + 01 ± 3.36E + 00	≈	1.88E + 01 ± 2.74E + 00	1.83E + 01 ± 3.20E + 00	≈	1.77E + 01 ± 3.09E + 00	1.67E + 01 ± 2.40E + 00	≈	1.74E + 01 ± 3.51E + 00
T13(Ca)	1.61E + 01 ± 3.42E + 00	+	1.11E + 01 ± 2.66E + 00	1.61E + 01 ± 7.42E + 00	+	1.07E + 01 ± 5.76E + 00	1.67E + 01 ± 4.99E + 00	≈	1.09E + 01 ± 4.67E + 00
	5/9/1		–	5/9/1		–	5/10/0		–

Algorithm 4 The pseudo code of SaDE-ARM algorithm.

```
Uniformly randomly initialize each individual within the searching range;
Calculate the fitness values of the initial population;
Set generation number G = 1; set population size as NP; set function evaluations as Fes = NP; set maximum function evaluations as Max_Fes; set
 learning period LP = NP; initial median value of Cr (Crm_l) and strategy probability p_{l,G} (l = 1, ..., L, L is the number of available strategies, L = 4);
 Clear Success and Failure Memory;
D_o = ∅; D_b = ∅; L_2 = ∅; set minimum support as Min_sup;
while Fes ≤ Max_Fes
    if G > LP
        Update p_{l,G} based on lth strategy performance in last LP generations;
        Update Success and Failure Memory;
end
 if mod(G, NP) == 0 & D_o ≠ ∅
    Discretize D_o into D_b;
    L_2 = Apriori-2 (D_b, 0.05);
    if L_2 ≠f∅
        Output corresponding F_u, F_l, Cr_u and Cr_l;
    end
 end
 for i = 1: NP
    % Mutation strategy adaption
    Select one strategy l for X_{i,G} by using roulette wheel selection according to p_{l,G};
    % Mutation
 F_i = normrnd(0.5, 0.3);
    Generate mutation vector V_{i,G} according to corresponding strategy l and F_i;
    if L_2 ≠f∅
        Generate F_{i_ARM} and Cr_{i_ARM} according to Eq. 7–8;
        Generate V_{i_ARM,G} according to corresponding strategy l and F_{i_ARM};
    end
    % Crossover
    Calculate lth crossover rate for V_{i,G} using Crm_l = median(Cr_memory_l);
    Cr_i = normrnd(Crm_l, 0.1);
    Generate trial vector U_{i,G} according to corresponding strategy l and Cr_i;
    Calculate the fitness value of U_{i,G};
    Fes = Fes + 1;
    if L_2 ≠f∅
        Generate U_{i_ARM,G} according to corresponding strategy l and Cr_{i_ARM};
        Calculate the fitness value of U_{i_ARM,G};
        Fes = Fes + 1;
    end
    % Selection
    if f(U_{i,G}) < f(U_{i_ARM,G}) & f(U_{i,G}) < f(X_{i,G})
        X_{i,G+1} = U_{i,G};
        Add Cr_i into Cr_memory_l;
        Add successful F and Cr into D_o;
    else if f(U_{i_ARM,G}) < f(U_{i,G}) & f(U_{i_ARM,G}) < f(X_{i,G})
        X_{i_ARM,G+1} = U_{i,G};
        Add successful F and Cr into D_o;
    end
    Add successful and failure information of lth mutation strategy into Success and Failure Memory, respectively;
 end
 G = G + 1;
end
```

and variances are reported in Table 8. Best results are shown in boldface.

From the compared results we can see, SaDE-ARM, AS-JADE-ARM and MDE_*p*BX-ARM could beat their corresponding original DE on 5 out of 15 problems, while they are defeated on only one problem. It seems that ARM-based parameter adaptive strategy is more suitable for jDE that other DE variants due to up to 8 winning problems. Meanwhile, we can see similar results for JADE-ARM and MDE_*p*BX-ARM. In general, the proposed ARM-based strategy might improve the performances of various DE algorithms on real-world problems to some extent. At least, it does not deteriorate the original searching capabilities of DE algorithms.

6. Conclusions and future work

In this paper, we propose a novel *F* and *Cr* adaption strategy that uses ARM methodology for DE variants. First, all successful *F* and *Cr* couples are recorded into a dataset along with evolution. Then, a modified Apriori algorithm is used to mine the most promising association pattern every *NP* generations. Further, we design a greedy operator to selection better solution by comparing trial vectors with new ARM-based and original methods. Comprehensive experimental results demonstrate that the proposed *F* and *Cr* adaption strategy could improve the overall performances of some state-of-the-art DE variants to some extent.

The main contribution of this paper is that, to adapt *F* and *Cr* values, we present a knowledge-based methodology which could extract explicit effective *F* and *Cr* couple associations for improving the performance of DE algorithms. The new strategy could automatically mine and generate appropriate *F* and *Cr* values that are suitable for various optimization problems and different evolving stages. Besides, we do not introduce any extra control parameter that would significantly influence the performance of the proposed strategy. Also, the authors believe that such ARM-based methodology could be applied for more control parameters of DE algorithms, such as selecting probability of mutation strategy and *NP*. While, we have to say that, it is a parameter adaption method which is only suitable for DE variants that could generate *F* and *Cr* randomly in relative large range.

In the future, we try to expand the application of this ARM-based strategy to more control parameters of versatile DE methods, even other EAs, such as PSO and GA. Meanwhile, it is attractive to apply such ARM-based strategy for various DE algorithms on real engineering problems, such as parameters identification of PMSM on shipboard and PI parameters optimization of PMSM controllers on shipboard.

Acknowledgement

This work was supported by Natural Science Foundation of China [grant numbers 51709027, 51506019]; Natural Science Foundation of Liaoning Province, China [grant numbers 2014025006]; Education Department General Project of Liaoning Province, China [grant numbers L2014209]; Doctoral Scientific Research Foundation Project of Liaoning Province, China [grant numbers 20170520265]; Yong Elite Scientists Sponsorship Program By CAST [grant numbers 2016QNRC001].

References

Abbass, H. A. (2002). The self-adaptive Pareto differential evolution algorithm. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on.: 1* (pp. 831–836).

Alatas, B., Akin, E., & Karci, A. (2008). Modenar: Multi-objective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing, 8*(1), 646–656.

Arce, F., Zamora, E., Sossa, H., & Barron, R. (2018). Differential evolution training algorithm for dendrite morphological neural networks. *Applied Soft Computing, 68*, 303–313.

Baig, M. Z., Aslam, N., Shum, H. P. H., & Zhang, L. (2017). Differential evolution algorithm as a tool for optimal feature subset selection in motor imagery EEG. *Expert Systems with Applications, 90*, 184–195.

Biswas, S., Kundu, S., Das, S., & Vasilakos, A. V. (2013). Teaching and learning best differential evoltuion with self adaptation for real parameter optimization. In *2013 IEEE Congress on Evolutionary Computation* (pp. 1115–1122).

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 Transactions on Evolutionary Computation, 10*(6), 646–657.

Buba, A. T., & Lee, L. S. (2018). A differential evolution for simultaneous transit network design and frequency setting problem. *Expert Systems with Applications, 106*, 277–289.

Bujok, P., Tvrdik, J., & Polakova, R. (2014). Differential evolution with rotation-invariant mutation and competing-strategies adaptation. In *2014 IEEE Congress on Evolutionary Computation* (pp. 2253–2258).

Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., et al. (2001). Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering, 13*(1), 64–78.

Cui, L., Li, G., Lin, Q., Chen, J., & Lu, N. (2016). Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations. *Computers & Operations Research, 67*, 155–173.

Cui, L. Z., Xu, C., Li, G. H., Ming, Z., Feng, Y. H., & Lu, N. (2018). A high accurate localization algorithm with DV-Hop and differential evolution for wireless sensor network. *Applied Soft Computing, 68*, 39–52.

Das, S., Mandal, A., & Mukherjee, R. (2014). An adaptive differential evolution algorithm for global optimization in dynamic environments. *IEEE Transactions on Cybernetics, 44*(6), 966–978.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution - an updated survey. *Swarm and Evolutionary Computation, 27*, 1–30.

Das, S., & Suganthan, P. N. (2011). Differential evolution: a survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*(1), 4–31.

De Falco, I., Della Cioppa, A., Maisto, D., Scafuri, U., & Tarantino, E. (2014). An adaptive invasion-based model for distributed differential evolution. *Information Sciences, 278*, 653–672.

Derrac, J., Garcia, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1*(1), 3–18.

Djenouri, Y., & Comuzzi, M. (2017). Combining Apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem. *Information Sciences, 420*, 1–15.

Doostan, M., & Chowdhury, B. H. (2017). Power distribution system fault cause analysis by using association rule mining. *Electric Power Systems Research, 152*, 140–147.

Draa, A., Bouzoubia, S., & Boukhalfa, I. (2015). A sinusoidal differential evolution algorithm for numerical optimisation. *Applied Soft Computing, 27*, 99–126.

Elsayed, S. M., Sarker, R. A., & Ray, T. (2013). Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization. In *2013 IEEE Congress on Evolutionary Computation* (pp. 1932–1937).

Fan, Q., & Yan, X. (2016). Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. *IEEE Transactions on Cybernetics, 46*(1), 219–232.

Gamperle, R., D Muller, S., & Koumoutsakos, A. (2002). A parameter study for differential evolution. 10(293–298.

Garcia, S., Fernandez, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180*(10), 2044–2064.

Ghosh, A., Das, S., Chowdhury, A., & Gini, R. (2011). An improved differential evolution algorithm with fitness-based adaptation of the control parameters. *Information Sciences, 181*(18), 3749–3765.

Gong, W., & Cai, Z. (2013). Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics, 43*(6), 2066–2081.

Gong, W., Fialho, A., Cai, Z., & Li, H. (2011). Adaptive strategy selection in differential evolution for numerical optimization: An empirical study. *Information Sciences, 181*(24), 5364–5386.

Harikumar, S., & Dilipkumar, D. U. (2016). Apriori algorithm for association rule mining in high dimensional data. In *Proceedings of the 2016 International Conference on Data Science & Engineering* (pp. 115–120).

Hong, T.-P., Chen, C.-H., Lee, Y.-C., & Wu, Y.-L. (2008). Genetic-fuzzy data mining with divide-and-conquer strategy. *IEEE Transactions on Evolutionary Computation, 12*(2), 252–265.

Hu, J., Wu, M., Chen, X., Du, S., Zhang, P., Cao, W. H., et al. (2018). A multilevel prediction model of carbon efficiency based on the differential evolution algorithm for the iron ore sintering process. *IEEE Transactions on Industrial Electronics, 65*(11), 8778–8787.

Islam, S. M., Das, S., Ghosh, S., Roy, S., & Suganthan, P. N. (2012). An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 42*(2), 482–500.

Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing, 9*(6), 448–462.

Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing, 11*(2), 1679–1696.

Martin, D., Rosete, A., Alcala-Fdez, J., & Herrera, F. (2014). A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules. *IEEE Transactions on Evolutionary Computation, 18*(1), 54–69.

Mlakar, U., Fister, I., Brest, J., & Potocnik, B. (2017). Multi-objective differential evolution for feature selection in facial expression recognition systems. *Expert Systems with Applications, 89*, 129–137.

Muangkote, N., Sunat, K., & Chiewchanwattana, S. (2017). Rr-cr-IJADE: An efficient differential evolution algorithm for multilevel image thresholding. *Expert Systems with Applications, 90*, 272–289.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., & Coello Coello, C. A. (2014a). A survey of multiobjective evolutionary algorithms for data mining: Part I. *IEEE Transactions on Evolutionary Computation, 18*(1), 4–19.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., & Coello Coello, C. A. (2014b). Survey of multiobjective evolutionary algorithms for data mining: Part II. *IEEE Transactions on Evolutionary Computation, 18*(1), 20–35.

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review, 33*(1-2), 61–106.

Omran, M. G. H., Salman, A., & Engelbrecht, A. P. (2005). Self-adaptive differential evolution. In *Computational Intelligence and Security, Pt 1, Proceedings: 3801* (pp. 192–199).

Patill, S. D., Deshmukh, R. R., & Kirange, D. K. (2016). Adaptive Apriori algorithm for frequent itemset mining. In *Proceedings of the 5th International Conference on System Modeling & Advancement in Research Trends* (pp. 7–13).

Pedrosa Silva, R. C., Lopes, R. A., & Guimaraes, F. G. (2011). Self-adaptive mutation in the differential evolution. In *GECCO-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference* (pp. 1939–1946).

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation, 13*(2), 398–417.

Ronkkonen, J., Kukkonen, S., & Price, K. V. (2005). Real-parameter optimization with differential evolution. In *2005 IEEE Congress on Evolutionary Computation, Vols 1-3, Proceedings* (pp. 506–513).

Sarker, R. A., Elsayed, S. M., & Ray, T. (2014). Differential evolution with dynamic parameters selection for optimization problems. *IEEE Transactions on Evolutionary Computation, 18*(5), 689–707.

Sheng, G., Hou, H., Jiang, X., & Chen, Y. (2018). A novel association rule mining method of big data for power transformers state parameters based on probabilistic graph model. *IEEE Transactions on Smart Grid, 9*(2), 695–702.

Storn, R., & Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.

Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation* (pp. 71–78).

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

Tsakiridis, N. L., Theocharis, J. B., & Zalidis, G. C. (2017). DECO3RUM: A differential evolution learning approach for generating compact Mamdani fuzzy rule-based models. *Expert Systems with Applications, 83*, 257–272.

Tung, A. K. H., Lu, H. J., Han, J. W., & Feng, L. (2003). Efficient mining of intertransaction association rules. *IEEE Transactions on Knowledge and Data Engineering, 15*(1), 43–56.

Wang, Y., Cai, Z., & Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation, 15*(1), 55–66.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., et al. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1–37.

Yang, Z., Tang, W. H., Shintemirov, A., & Wu, Q. H. (2009). Association rule mining-based dissolved gas analysis for fault diagnosis of power transformers. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews, 39*(6), 597–610.

Yi, W., Zhou, Y., Gao, L., Li, X., & Mou, J. (2016). An improved adaptive differential evolution algorithm for continuous optimization. *Expert Systems with Applications, 44*, 1–12.

Yu, W.-J., Shen, M., Chen, W.-N., Zhan, Z.-H., Gong, Y.-J., Lin, Y., et al. (2014). Differential evolution with two-level parameter adaptation. *IEEE Transactions on Cybernetics, 44*(7), 1080–1099.

Yu, W.-J., & Zhang, J. (2011). Multi-population differential evolution with adaptive parameter control for global optimization. In *GECCO-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference* (pp. 1093–1098).

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering, 12*(3), 372–390.

Zhang, J. Q., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation, 13*(5), 945–958.

Zheng, L. M., Zhang, S. X., Tang, K. S., & Zheng, S. Y. (2017). Differential evolution powered by collective information. *Information Sciences, 399*, 13–29.

Zielinski, K., Weitkemper, P., Laur, R., & Kammeyer, K.-D. (2006). Parameter study for differential evolution using a power allocation problem including interference cancellation. In *2006 IEEE Congress on Evolutionary Computation: Vols 1-6* (pp. 1857–1864).

Zou, D., Wu, J., Gao, L., & Li, S. (2013). A modified differential evolution algorithm for unconstrained optimization problems. *Neurocomputing, 120*, 469–481.