Information Sciences 329 (2016) 329–345

Contents lists available at ScienceDirect

Information Sciences

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

Differential evolution with multi-population based ensemble of mutation strategies

Guohua Wu [a,*], Rammohan Mallipeddi [b], P.N. Suganthan [c], Rui Wang [d], Huangke Chen [a]

[a] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, Hunan, PR China
[b] School of Electronics Engineering, Kyungpook National University, 1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, South Korea
[c] School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore, Singapore
[d] College of Information System and Management, National University of Defense Technology, Changsha 410073, Hunan, PR China

ARTICLE INFO

Article history:
Received 16 April 2015
Revised 7 September 2015
Accepted 12 September 2015
Available online 25 September 2015

Keywords:
Evolutionary algorithm
Differential evolution
Multi-population
Ensemble of mutation strategies
Numerical optimization

ABSTRACT

Differential evolution (DE) is among the most efficient evolutionary algorithms (EAs) for global optimization and now widely applied to solve diverse real-world applications. As the most appropriate configuration of DE to efficiently solve different optimization problems can be significantly different, an appropriate combination of multiple strategies into one DE variant attracts increasing attention recently. In this study, we propose a multi-population based approach to realize an ensemble of multiple strategies, thereby resulting in a new DE variant named multi-population ensemble DE (MPEDE) which simultaneously consists of three mutation strategies, i.e., "current-to-pbest/1" and "current-to-rand/1" and "rand/1". There are three equally sized smaller indicator subpopulations and one much larger reward subpopulation. Each constituent mutation strategy has one indicator subpopulation. After every certain number of generations, the current best performing mutation strategy will be determined according to the ratios between fitness improvements and consumed function evaluations. Then the reward subpopulation will be allocated to the determined best performing mutation strategy dynamically. As a result, better mutation strategies obtain more computational resources in an adaptive manner during the evolution. The control parameters of each mutation strategy are adapted independently as well. Extensive experiments on the suit of CEC 2005 benchmark functions and comprehensive comparisons with several other efficient DE variants show the competitive performance of the proposed MPEDE (Matlab codes of MPEDE are available from http://guohuawunudt.gotoip2.com/publications.html).

1. Introduction

Differential evolution (DE), first proposed by Storn and Price [47], is one of the most efficient evolutionary algorithms (EAs) currently in use. DE is a population-based stochastic search technique, in which mutation, crossover, and selection operators are utilized at each generation to move the population toward the global optimum [56]. Numerous studies have been done on DE with respect to the novel mutation strategy design [5,23,51,52], hybridization [37,43,72] and population diversity control

* Corresponding author. Tel.: +86 15580845945; fax: +86 15580845945.
E-mail address: guohuawu@nudt.edu.cn (G. Wu).

http://dx.doi.org/10.1016/j.ins.2015.09.009
0020-0255/

[10,61,73]. DE has been now widely applied to solve various optimization problems from various fields, such as power systems optimization [7,60], time series prediction [15], and feature selection [2].

The performance of DE highly depends on the configuration of mutation strategy and control parameters, such as population size *NP*, scaling factor *F* and crossover rate C_r [33]. Generally, the most appropriate mutation and parameter settings required by DE to solve different optimization problems are different [56]. This is because some mutation strategies are effective for the global search [39] and some others are useful for rotated problems [12], and that some control parameter settings can promote the convergence [40] and some other settings are more efficient for solving separable functions [41]. In addition, evidences show that even for one specific optimization problem, the required best strategies and parameters may vary during the evolutionary process [39]. Moreover, as can be observed in literature, several claims and counter-claims were reported concerning the setting of the control parameters [14]. Nevertheless, traditional trial-and-error approaches for determining the best strategy and parameters are usually inefficient and time-consuming, especially when solving a variety of optimization problems.

As a result, methods for automated tuning or ensemble of mutation strategies and parameters naturally attract increasing attention [6,19,33,39,67]. Consequently, many enhanced DE variants such as SaDE (with adapted mutation strategies and parameters) [39], jDE (with self-adapted parameters) [6], CoDE (composition of multiple strategies and parameter settings) [56], JADE (with "current-to-pbest/1" mutation strategy and adaptive parameters) [67], DE–DPS [42] (with dynamic selection of the best performing combinations of parameters), self-CCDE and self-CSDE [17] (with cluster-based strategy and self-adaptive parameter control), ADE [63] (with two-level adaptive parameter control scheme), CoBiDE (with covariance matrix learning and bimodal distribution parameter setting) [57] and EPSDE (with ensemble of mutation strategies and parameters) [33], have been proposed.

In this study, we propose a novel DE variant (named MPEDE for short), in which a multi-population based approach is utilized to realize a dynamic ensemble of multiple mutation strategies. In addition, parameters such as scaling factor *F* and crossover rate C_r, associated with each mutation strategy are adapted based on the approach proposed in [67]. In MPEDE, mutation strategies "current-to-pbest/1", "current-to-rand/1" and "rand/1" are taken as constituent mutation strategies. There are two types of subpopulations in MPEDE namely, three indicator subpopulations and one reward subpopulation. Initially, each mutation strategy obtains an indicator subpopulation and the reward subpopulation is randomly assigned to one of the three mutation strategies. Then during the evolutionary process, after every certain number of generations, the mutation strategy that performed the best during the previous generations is determined with respect to the ratios between the fitness improvements and consumed function evaluations. Subsequently, the reward subpopulation is assigned to the determined best performed mutation strategy as a reward. With the algorithm proceeding, the best mutation strategy determination and reward population assignment operations are executed periodically. By using these steps, we ensure that the recently best performing strategy will be given more computational resources. MPEDE is tested on the suit of CEC 2005 benchmark functions with 30 and 50 variables, respectively. The competitive performance of MPEDE is exhibited by extensive comparisons with several state-of-the-art DE variants.

Recently, population partitioning techniques for enhancing the performance of EAs and swarms, such as particle swarm optimization (PSO) and DE, attracted increasing attention [4,28,35,64,68,70]. Our work is different from previous studies in several aspects. First and foremost, the application of multi-population techniques in previous literature is aimed to maintain population diversities of EAs while our study is aimed to realize the ensemble of multiple mutation strategies as well as automated computational resource allocation among mutation strategies of DE. Second, all previous work partition the original population into multiple smaller ones that have the same sizes. By contrast, the sizes of subpopulations in this paper are not equal. Third, major former studies utilize the same mutation strategy in different subpopulations while in MPEDE three mutation strategies are employed and the best mutation strategy will dynamically be rewarded with larger population resources during the run of MPEDE. We believe that the proposed multi-population framework will be a new paradigm for effective ensemble of multiple strategies for DE.

The rest of the paper is structured as follows: Section 2 gives a brief introduction to canonical DE, including its typical mutation operators, crossover, and selection operators. Section 3 reviews the related works in literature. Section 4 introduces details of the implementation of MPEDE. Section 5 reports the experimental results. Section 6 applies MPEDE to solve a real-world problem. Section 7 concludes this paper.

2. Differential evolution

Differential evolution (DE) being a parallel direct search method utilizes *NP*, *D*-dimensional decision vectors called population that encodes the candidate solutions, i.e. $\mathbf{X}_{i,G} = \{x^1_{i,G}, \ldots, x^D_{i,G}\}, i = 1, \ldots, NP$. The initial value of the *j*th decision variable of the *i*th individual at generation $G = 0$ is generated within the search space constrained by the prescribed minimum and maximum decision variable's bounds $\mathbf{X}_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$ and $\mathbf{X}_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$ by:

$$x^j_{i,0} = x^j_{\min} + \text{rand}(0,1).(x^j_{\max} - x^j_{\min}) \quad j = 1, 2, \ldots, D \tag{1}$$

where rand(0,1) represents a uniformly distributed random variable within the range [0,1]. In other words, the initial population is obtained by uniform random sampling of the search space.

After initialization, the population evolves over generations through operations such as mutation, crossover and selection. In every generation, corresponding to each individual in the current population, trial vectors are produced through mutation and crossover operations. Each trial vector competes to replace the corresponding parent in the population during the selection process.

At generation G, during the mutation process, corresponding to each individual $\mathbf{X}_{i,G}$ in the current population, mutant vector $\mathbf{V}_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ is produced employing one of the following mutation strategies.

$$\text{“DE/best/1” [46]}: \quad \mathbf{V}_{i,G} = \mathbf{X}_{\text{best},G} + F.(\mathbf{X}_{r_1{}^i,G} - \mathbf{X}_{r^i_2,G}) \tag{2}$$

$$\text{“DE/best/2” [46]}: \quad \mathbf{V}_{i,G} = \mathbf{X}_{\text{best},G} + F.(\mathbf{X}_{r_1{}^i,G} - \mathbf{X}_{r^i_2,G}) + F.(\mathbf{X}_{r_3{}^i,G} - \mathbf{X}_{r^i_4,G}) \tag{3}$$

$$\text{“DE/rand/1” [46]}: \quad \mathbf{V}_{i,G} = \mathbf{X}_{r_1{}^i,G} + F.(\mathbf{X}_{r_2{}^i,G} - \mathbf{X}_{r^i_3,G}) \tag{4}$$

$$\text{“DE/rand/2” [39]}: \quad \mathbf{V}_{i,G} = \mathbf{X}_{r_1{}^i,G} + F.(\mathbf{X}_{r_2{}^i,G} - \mathbf{X}_{r^i_3,G}) + F.(\mathbf{X}_{r_4{}^i,G} - \mathbf{X}_{r^i_5,G}) \tag{5}$$

$$\text{“DE/current-to-rand/1” [24]}: \quad \mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K.(\mathbf{X}_{r^i{}_1,G} - \mathbf{X}_{i,G}) + F.(\mathbf{X}_{r^i{}_2,G} - \mathbf{X}_{r^i{}_3,G}) \tag{6}$$

In Eqs. (2)–(6), corresponding to each i, the indices $r^i_1, r^i_2, r^i_3, r^i_4, r^i_5$ should be mutually exclusive and are generated randomly once for each mutant vector within the range of [1, NP]. $\mathbf{X}_{\text{best},G}$ is the best individual vector with the best fitness value in the population at generation G. F is positive parameter referred to as the scale parameter for scaling the difference vector. K and F are randomly chosen within the range [0,1].

From the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$, trial vector $\mathbf{U}_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \ldots, u^D_{i,G}\}$ is generated by a crossover operation. In DE, the crossover operation can be realized by using one of the two methods referred to as binomial (or uniform) and exponential (or two-point modulo) [71]. Frequently DE employs binomial crossover defined as [47]:

$$u^j_{i,G} = \begin{cases} v^j_{i,G} & \text{if } (\text{rand}_j[0,1] \le CR) \text{ or } (j = j_{\text{rand}}), \quad j = 1, 2, \ldots, D \\ x^j_{i,G} & \text{otherwise} \end{cases} \tag{7}$$

In (7), the crossover probability, $CR \in [0,1]$, is a user-specified constant which controls the number of decision variable values that are copied from the mutant vector to the trail vector. j_{rand} is a randomly chosen integer in the range [1, D]. In the above equation, the condition $j = j_{\text{rand}}$ is employed to ensure trial vector $\mathbf{U}_{i,G}$ to differ from its corresponding target vector $\mathbf{X}_{i,G}$ in at least one decision variable. Binomial crossover is uniform and does not exhibit a representational bias since each decision variable of the mutant vector, regardless of its location, has the same probability CR of inheriting its value from a given to the trail vector $\mathbf{U}_{i,G}$.

During mutation and crossover operations, if any decision variable's value of newly generated trial vectors exceeds the upper or lower bound, then they are set to the corresponding bound or reinitialized randomly and uniformly within the pre-specified range. The objective function values of all trial vectors are then evaluated and the selection operation is performed. In other words, the objective function value of each trial vector $f(\mathbf{U}_{i,G})$ is compared to its corresponding target vector $f(\mathbf{X}_{i,G})$ in the current population. In a minimization problem, if the trial vector has a less or equal objective function value compared to the corresponding target vector, then the trial vector will replace the target vector and enter the population for the next generation. If not, the target vector remains in the population for the next generation. The selection operation can be expressed as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & \text{if } f(\mathbf{U}_{i,G}) \le f(\mathbf{X}_{i,G}) \\ \mathbf{X}_{i,G}, & \text{otherwise} \end{cases} \tag{8}$$

These 3 steps – mutation, crossover, and selection – are repeated generation after generation until a termination criterion (reaching the preset maximum number of function evaluations) is satisfied.

3. Related works

Differential evolution (DE) algorithm is a floating-point encoded evolutionary algorithm over continuous spaces [47]. Recently, DE has attracted much attention due to its simplicity. However, the performance of conventional DE algorithm depends on the chosen mutation/crossover strategies and the associated control parameters. In addition, the performance of DE becomes more sensitive to strategies and their associated parameter values as the complexity of the problem increases [16]. In other words, the inappropriate selection of strategies and parameters may lead to a premature convergence, stagnation, or a waste of computational resources [13,16,27,38,65]. Initially it was thought that [47,48] selection of strategies and parameters is straightforward. But, due to the complex decision variables' interaction with DE's performance on hard optimization problems [6], choosing appropriate mutation and control strategies and control parameters require some expertise. Since DE was proposed, various empirical guidelines have been suggested for choosing strategies and their associated control parameter settings depending on the characteristics of the problem.

The standard DE algorithm proposed by Price and Storn employs DE/rand/1/bin. In [16, 47] it was stated that 2 difference vector strategies, such as DE/rand/2/bin and DE/best/2/bin, are better single difference vector strategies such as DE/rand/1/bin and DE/best/1/bin due to their ability to improve the diversity by producing more trial vectors [27]. In addition, the mutation strategies relying on the best individual in the current population such as DE/best/1/bin and DE/rand-to-best/1/bin are faster for easier optimization problems, but become unreliable when solving highly multi-modal problems. To balance the exploration

and exploitation abilities of DE, the DE/target-to-best/1/bin scheme employing a topological neighborhood concept of each population member was proposed [12]. Classical rand-based strategy is slower, but more robust compared to strategies that rely on the best-so-far vector. "current-to-rand/1" mutation strategy demonstrates rotation-invariance compared to other strategies [24]. In [67], Zhang and Sanderson proposed the "current-to-pbest/1" mutation operator with optional archive, which balances the exploitation and exploration capabilities. Exploitation and exploration are two key points in designing efficient evolutionary and swarm intelligence algorithms [59]. In [18], ranking-based mutation operators were designed and incorporated into classical mutation strategies, which results in significant performance improvement.

Extensive studies have been done on appropriate setting of the control parameters of DE, such as the population size NP, crossover rate CR and scaling factor F. For example, separable and uni-modal functions require smaller population sizes to speed up the convergence, whereas decision variable-linked multi-modal functions require larger populations to avoid premature convergence. Initially, a population size $NP = 10D$ was considered to be a good choice for the DE to find the global optimum [47]. However, to balance convergence speed and reliability, different ranges of NP values, such as $5D$–$10D$ [47], $3D$–$8D$ [16], and $2D$–$40D$ [41] have also been suggested. Crossover rate CR ($0 \leq CR \leq 1$) controls the number of components in each individual to be mutated in the current population [66]. A large value of CR speeds up convergence [16, 46, 47]. In [47], it was said that $CR = 0.1$ was a good initial choice whereas $CR = 0.9$ or 1.0 could be tried to increase the convergence speed. In [16], a good choice for CR was said to lie in between 0.3 and 0.9. The scaling factor, F is generally chosen from [0.5,1] [46]. A larger F increases the probability of escaping from a local optimum [16,41]. In [16,47], it was said that $F = 0.6$ or 0.5 would be a good initial choice. However, in [41] it was mentioned that $F = 0.9$ would be a good initial choice. Therefore, according to [41] typical values of F lie in between 0.4 and 0.95.

From the above, it can be observed that various conclusions have been drawn regarding the manual parameter tuning of DE, which lack sufficient justification. Therefore, to avoid the tuning of trial-and-error, various techniques have been developed. Some parameter adaptation strategies were proposed, such as linear reduction [13], random sampling [8], fuzzy logic control [30], simulated annealing based parameter control [22] and population diversity based parameter control [65]. Omran et al. [36] introduced a self-adaptation scheme (SDE) in which the CR was generated randomly for each individual using a normal distribution $N(0.5, 0.15)$, whereas the scale factor F was adapted analogously to the adaptation of the crossover rate CR in [1]. Brest et al. [6] proposed a self-adaptation scheme (jDE), in which control parameters F and CR were encoded into the individuals and are adjusted in the run of DE.

Recently, to alleviate the problem of strategy and parameter tuning, the idea of incorporating ensemble strategies and parameters into evolutionary algorithms has been explored. Qin et al. [39] proposed a self-adaptive DE algorithm (SaDE) in which the mutation strategies and the respective control parameter are self-adapted based on their previous experiences of generating promising solutions. The scale factor, F was randomly generated with a mean and standard deviation of 0.5 and 0.3, respectively. Gong et al. [20] presented two DE variants with two adaptive strategy selection techniques, namely the Probability Matching and Adaptive Pursuit to choose appropriate mutation strategies with certain probabilities. The selection probabilities of mutation strategies are determined by their respective previous search performance that is evaluated by a credit assignment technique. In [31,33], the authors proposed a DE algorithm with an ensemble of mutation strategies and parameter values (EPSDE) which consists of a pool of mutation and crossover strategies and their associated parameter values. Initially, the population members randomly pick the strategies and parameter values from the respective pools and produce an offspring population. Depending on the success of the offspring, the corresponding combination of strategies and their associated parameter values is retained or reinitialized. In EPSDE, the retaining of the combination that produces better offspring and reinitializing the combination that are incapable of producing competitive offspring favors the selection of the combination that produce better solutions in the due course of the evolution. Inspired by the success, the idea ensemble learning has been further explored to solve constrained optimization problems [32]. Wang and co-workers presented a DE variant named ICDE to deal with constrained optimization problems. ICDE employs multiple mutation strategies and the binomial crossover to generate the offspring population [25]. Very recently, Gong et al. [21] proposed a cheap surrogate model for the ensemble of multiple search operators in evolutionary optimization. In their approach, a set of candidate offspring solutions are generated by using the multiple offspring reproduction operators and the best one according to the surrogate model is chosen as the offspring solution. Multi-objective optimization involves more than one objective function to be optimized simultaneously, which plays an important role in multiple criteria decision making [53–55]. Zhao et al. proposed an ensemble of different neighborhood sizes with online self-adaptation to enhance the multiobjective evolutionary algorithm based on decomposition [69].

The concepts of "multi-swarm" and "multi-population island models" have been introduced to improve the performance of particle swarm optimization (PSO) and DE in several studies [4,28,29,35,64,68,70]. These previous studies mainly partition the initial population or swarm into multiple equal smaller subpopulations or sub-swarms. As the algorithm proceeds, information exchange among subpopulations (or sub-swarms) and regrouping operators will be triggered with a certain frequency with the aim to maintain the diversity of the whole population and balance the exploitation and exploration capabilities. Recently, a multi-population DE (mDE-bES) was introduced to boost the population diversity while preserving simplicity to solve large-scale global optimization problems [3]. In mDE-bES, subpopulations have the same sizes and each is with a different mutation and update strategy. After every certain number of generations, individuals between the subgroups are exchanged to facilitate information exchange. Shang et al. [45] proposed a multi-population based cooperative coevolutionary algorithm (MPCCA) to solve the multi-objective capacitated arc routing problem. In MPCCA, population is partitioned into multiple subpopulations with respect to their different direction vectors. These subpopulations evolve separately and search different objective sub-regions simultaneously.

The adjacent subpopulations are able to share their information. The differences between our research and other related studies are explained in Section 1.

4. Multi-population based ensemble DE (MPEDE)

In this work, to realize an effective ensemble of multiple mutation strategies into one DE variant, we dynamically partition the whole population into multiple indicator subpopulations (with equal and relatively smaller sizes) and one relatively large sized reward subpopulation at each generation. Each indicator subpopulation is assigned to a mutation strategy. Meanwhile, the reward subpopulation is assigned to the mutation strategy that performed the best during the recent previous generations. Subpopulations represent computational resources. Thus, the best performing mutation strategy is able to gain more computational resources.

4.1. Multi-population based mutation strategy ensemble approach

Different mutation strategies are required for a DE variant to solve various optimization problems efficiently. In addition, even for a specific optimization problem, the most suitable mutation strategies may be different at different stages of the evolutionary process. As a result, the choice of candidate mutation strategy plays a key role in designing an efficient ensemble DE variant. Our principle is to choose well investigated mutation strategies each of which has respective advantages. Three popular mutation strategies are selected in this study, including "current-to-pbest/1" with an archive, "current-to-rand/1" and "rand/1". The "rand/1" is robust and the most commonly used mutation strategy. In addition, evidences show that "current-to-pbest/1" with an archive is very competitive in solving complex optimization problems, especially those with unimodal landscapes [67] or after one or more population members discover the global basin in a multimodal landscape. In contrast, the "current-to-rand/1" mutation strategy, which is applied without the aid of crossover operation, is particularly useful in solving rotated problems, as it is rotation-invariant [12]. The employed constituent mutation strategies are listed as below. It should be noted that the "rand/1" and "current-to-pbest/1" mutation strategies are used with the combination of binominal crossover while the "current-to-rand/1" strategy is used without crossover.

Mutation strategy 1: "current-to-pbest/1"

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot (\mathbf{X}_{\text{pbest},G} - \mathbf{X}_{i,G} + \mathbf{X}_{r_1^i,G} - \tilde{\mathbf{X}}_{r_2^i,G}) \tag{9}$$

Mutation strategy 2: "current-to-rand/1"

$$\mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{r_1^i,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G}) \tag{10}$$

Mutation strategy 3: "rand/1"

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \cdot (\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G}) \tag{11}$$

To realize the effective ensemble of the three constituent mutation strategies, we divided the population into four subpopulations, say pop_1, pop_2, pop_3 and pop_4 randomly every generation. pop_1, pop_2 and pop_3 have the same size and are categorized as indicator subpopulations while pop_4 is categorized as the reward subpopulation. In general, the size of an indicator subpopulation is much smaller than the size of the reward subpopulation. Let pop denote the overall population. Obviously, we have

$$pop = \bigcup\nolimits_{j=1,\ldots 4} pop_j \tag{12}$$

Let NP be the size of pop, NP_j be the size of pop_j, and λ_j be the portion between pop_j and pop. Clearly, we have

$$NP_j = \lambda_j \cdot NP, \quad j = 1, 2, \ldots, 4 \tag{13}$$

$$\sum_{j=1,\ldots 4} \lambda_j = 1 \tag{14}$$

In this study, we let $\lambda_1 = \lambda_2 = \lambda_3$.

At the beginning, subpopulations pop_1, pop_2 and pop_3 are assigned to the corresponding mutation strategies and pop_4 is randomly assigned to one constituent mutation. As the algorithm proceeds, after every ng (a predefined parameter) number of generations, we determine the mutation strategy that performed the best during the previous ng generations. The determined best performed mutation strategy then will be rewarded by more computational resources in the following ng generations by assigning the reward subpopulation pop_4. When determining the current best performing constituent mutation strategy, the metric for evaluating the performance of the jth mutation strategy is $\Delta f_j / \Delta Fes_j$, where Δf_j is the accumulated fitness improvement brought by the jth mutation strategy during the previous ng generations, and ΔFes_j is the function evaluations consumed by the jth mutation strategy during the former ng generations.

The best performing mutation strategy determination and reward subpopulation assignment operations described above are executed periodically with ng being the period. With this idea, we ensure that the best mutation strategy consumes the most computational resources.

The indicator subpopulation is to provide each mutation strategy with basic and sufficient computational resources to facilitate the effective evaluation of the performance of each mutation strategy. The reward subpopulation gives more resources to the best performed mutation strategy which then dominates the optimization process. It should be noted that, in the implementation of MPEDE, each subpopulation is randomly sampled from the overall population at each generation, which actually enables mutation strategies to share optimization experience with each other and realizes the information exchange among subpopulations.

4.2. Parameter adaptation

Several effective parameter adaptation approaches have been proposed in previous studies [6,39,67]. As mutation strategies and parameters are correlated in influencing the performance of a DE variant and different mutation strategies may require different parameter settings, we let each mutation strategy have its independent parameters. We have tried to use different techniques in [6,39,67] to adapt the parameters of MPEDE and found the technique in [67] is the most suitable. As a result, the technique in [67] is eventually applied and extended here to adapt the parameters of multiple mutation strategies in MPEDE.

Let $CR_{i,j}$ be the crossover probability of individual X_i that uses jth mutation strategy to produce a trial solution. At each generation g, $CR_{i,j}$ is generated according to the following normal distribution,

$$CR_{i,j} = \text{randn}_{i,j}(\mu CR_j, 0.1), \tag{15}$$

where, μCR_j is the mean value and 0.1 is the standard deviation value. $CR_{i,j}$ will be truncated to [0,1] if necessary. Let $S_{CR,j}$ be the collection of any $CR_{i,j}$ that helps the jth mutation strategy to generate improved solutions at generation g. The initial value of μCR_j is set to 0.5. After each generation, μCR_j is updated as

$$\mu CR_j = (1-c)\cdot \mu CR_j + c \cdot \text{mean}_A(S_{CR,j}), \tag{16}$$

where, c is a positive constant between 0 and 1 and mean_A is a function calculating the arithmetic mean value of elements in $S_{CR,j}$.

Similarly, the scaling factor $F_{i,j}$ of individual X_i that uses the jth mutation strategy is updated according to Cauchy distribution as below at each generation g

$$F_{i,j} = \text{randc}_{i,j}(\mu F_j, 0.1), \tag{17}$$

where, μF_j is the location parameter and 0.1 is the scale parameter of the used Cauchy distribution. Also, $F_{i,j}$ will be truncated to [0,1] if necessary after the update.

Let $S_{F,j}$ be the collection of any $F_{i,j}$ that helps the jth mutation strategy to generate improved solutions at generation g. μF_j is initialized to 0.5 and updated as below at each generation,

$$\mu F_j = (1-c)\cdot \mu F_j + c \cdot \text{mean}_L(S_{F,j}), \tag{18}$$

where, mean_L is the Lehmer mean as below

$$\text{mean}_L = \frac{\sum_{F\in S_F} F^2}{\sum_{F\in S_F} F} \tag{19}$$

According to the mutation strategy ensemble and parameter adaptation introduced above, we come to the framework of MPEDE as given in Algorithm 1.

5. Experimental study

5.1. Experimental settings

To test the performance of MPEDE, a suit of 25 well benchmarked optimization functions proposed in the CEC 2005 special session on real-parameter optimization are utilized [49]. This benchmark suit includes unimodal functions ($F1$–$F5$), basic multimodal functions ($F6$–$F12$), expanded multimodal functions ($F13$–$F14$) and hybrid composition functions ($F15$–$F25$). For more information about the benchmark optimization functions, please refer to [49]. When implementing MPEDE, corresponding parameters are set as: $\lambda_1 = \lambda_2 = \lambda_3 = 0.2$, $NP = 250$ and $ng = 20$. Related parameter sensitivity analyses will be given in Section 5.4.

MPEDE was compared with six other state-of-the-art DE variants including JADE [67], jDE [6], SaDE [39], EPSDE [33], CoDE [56] and SHADE [50]. The reasons we choose these six DE variants as comparative algorithms are explained as follows. First, JADE and jDE are two representative DE variants that are very efficient and frequently cited in literature as baseline algorithms. Second, SaDE, EPSDE and CoDE also incorporate multiple mutation strategies as MPEDE. Hence, it is meaningful to compare MPEDE with them. Third, SHADE is a recently proposed DE variant, which reflects the latest progress of DE. All the mutation strategies and parameter settings of these DE variants are the same as those given in the original references.

To provide a more comprehensive comparison, we run each comparative algorithm 25 times over the benchmark functions with 30 and 50 decision variables. The allowed maximum function evaluations (FEs) for the benchmark functions with 30 and 50 decision variables are set to 300 000 and 500 000, respectively.

Algorithm 1
Pseudo code of MPEDE.

Set $\mu CR_j = 0.5$, $\mu F_j = 0.5$, $\Delta f_j = 0$ and $\Delta Fes_j = 0$ for each $j = 1, \ldots, 4$;
Initialize, NP, ng for each $j = 1, \ldots, 4$;
Initialize the pop randomly distributed in the solution space;
Initial λ_j and set $NP_j = \lambda_j \cdot NP$;
Randomly partition pop into pop_1, pop_2, pop_3 and pop_4 with respect to their sizes.;
Randomly select a subpopulation pop_j ($j = 1, 2, 3$) and combine pop_j with pop_4. Let $pop_j = pop_j \cup pop_4$ and $NP_j = NP_j + NP_4$;
Set $g = 0$;
While $g \leq MaxG$
  $g = g + 1$;
  For $j = 1 \rightarrow 3$
    Calculate μCR_j and μF_j;
    Calculate $CR_{i,j}$ and $F_{i,j}$ for each individual X_i in pop_j;
    Perform the jth mutation strategy and related crossover operators over subpopulation pop_j;
    Set $S_{CR,j} = \emptyset$ and $S_{F,j} = \emptyset$;
  End For
  For $i = 1 \rightarrow NP$
    If $f(X_{i,g}) \leq f(u_{i,g})$
      $X_{i+1,g} = X_{i,g}$;
    Else
      $X_{i+1,g} = U_{i,g}$; $\Delta f_j = \Delta f_j + f(X_{i,g}) - f(u_{i,g})$;
      $CR_{i,j} \rightarrow S_{CR,j}$; $F_{i,j} \rightarrow S_{F,j}$
    If Else
  End For
  $pop = \bigcup_{j=1,\ldots,3} pop_j$;
  If $\mathrm{mod}(g, ng) == 0$
    $k = \arg\left(\max_{1 \leq j \leq 3}\left(\frac{\Delta f_j}{ng \cdot NP_j}\right)\right)$
    $\Delta f_j = 0$;
  End If
  Randomly partition pop into pop_1, pop_2, pop_3 and pop_4;
  Let $pop_k = pop_k \cup pop_4$ and $NP_k = NP_k + NP_4$;
End While

5.2. Experimental results and comparisons with other peer DE variants

The computational results obtained by running each of the six comparative DE variants 25 times on each benchmark function with 30 and 50 variables are reported in Tables 1 and 2, respectively. The mean error and standard deviation (in bracket) of the function error values are provided in the two tables. Results obtained by MPEDE are highlighted if they are the best. In addition, Wilcoxon's rank sum test at a 0.05 significance level is conducted between MPEDE and JADE, jDE, SaDE, EPSDE, CoDE and SHADE. Signs "−", "+", and "≈" indicate that the related comparative DE variant is significantly worse than, better than, and similar to MPEDE, respectively. From the data given in Table 1, we can make several observations and conclusions.

First, for unimodal functions $F1$–$F5$, JADE and SHADE show the best performance. MPEDE is also very competitive. Actually, MPEDE obtains significantly better results for $F3$ than all other peers. In addition, MPEDE is comparable to JADE and SHADE on functions $F1$ and $F4$. Although JADE and SHADE outperform MPEDE on functions $F2$ and $F5$, MPEDE is capable of finding the optimal solutions for these two functions at the cost of a few more functions evaluations. Whereas, compared with the other four DE variants, jDE, SaDE, EPSDE and CoDE, MPEDE exhibits better overall performance. MPEDE outperforms jDE, SaDE, EPSDE and CoDE on four, four, three and four benchmark functions, respectively. jDE, SaDE and CoDE are not able to produce a better solution than MPEDE on any of the considered unimodal functions. EPSDE is comparable to MPEDE on benchmark function $F2$.

Secondly, for basic multi-modal benchmark functions $F6$–$F12$, both MPEDE and CoDE show better performance than other comparative ones. MPEDE is superior to CoDE on functions $F7$, $F10$ and $F12$ while inferior to CoDE on functions $F6$, $F8$ and $F11$. MPEDE outperforms JADE, jDE, SaDE, EPSDE and SHADE on three ($F10$–$F12$), five ($F6$, $F7$ and $F10$–$F12$), four ($F7$ and $F10$–$F12$), four ($F7$ and $F10$–$F12$) and three ($F10$–$F12$) benchmark functions, respectively. Both JADE and jDE cannot beat MPEDE on any of the considered basic multi-modal functions. SaDE, EPSDE and SHADE are better than MPEDE on one ($F11$), one ($F6$) and two ($F6$ and $F8$) functions, respectively.

Thirdly, as for expanded multimodal functions, MPEDE generally performs worse than other peer DE variants (except SaDE) on function $F13$ while it outperforms (jDE, SaDE and EPSDE) or equals (JADE, CoDE and SHADE) to other DE variants on function $F14$.

Finally, with regard to the more complex hybrid composition functions, MPEDE exhibits better overall performance than these comparative DE variants. In fact, it outperforms JADE, jDE, SaDE, EPSDE, CoDE and SHADE on four ($F16$, $F17$, $F23$ and $F25$), three ($F16$, $F17$ and $F25$), five ($F16$, $F17$, $F21$, $F22$ and $F25$), six ($F16$, $F17$, $F21$ and $F23$–$F25$), four ($F16$, $F17$, $F19$ and $F25$) and three ($F16$, $F17$ and $F25$) benchmark functions, respectively. JADE, jDE, CoDE and SHADE cannot outperform MPEDE on any of the considered hybrid composition functions. The performance of SaDE is better than that of MPEDE only on function $F19$. EPSDE is superior to MPEDE on functions $F15$, $F18$–$F20$ and $F22$.

Table 1
Computational result of benchmark functions with 30 variables.

Functions		JADE	jDE	SaDE	EPSDE	CoDE	SHADE	MPEDE
Unimodal functions	F1	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)
	F2	1.26E−28 (1.22E−28)+	3.45E−06 (2.76E−06)−	2.77E−06 (8.52E−06)−	8.32E−26 (2.66E−26)≈	6.77E−15 (3.44E−15)−	4.51e−29 (7.28e−29)+	1.01E−26 (2.05E−26)
	F3	8.42E+03 (6.58E+03)−	2.44E+05 (3.22E+05)−	5.33E+05 (4.34E+05)−	6.34E+05 (3.44E+06)−	5.65E+05 (5.66E+04)−	6.20e+03 (5.14e+03)−	1.01E+01 (8.32E+00)
	F4	4.13E−16 (3.45E−16)≈	4.78E−02 (2.12E−01)−	1.93E+02 (3.22E+02)−	3.88E+02 (3.13E+03)−	6.21E−03 (4.67E−02)−	7.03e−16 (1.01e−15)≈	6.61E−16 (5.68E−16)
	F5	7.59E−08 (5.65E−07)+	5.56E+02 (5.62E+02)−	3.76E+03 (6.12E+02)−	1.38E+03 (7.43E+02)−	3.16E+02 (3.62E+02)−	3.15e−10 (6.91e−10)+	7.21E−06 (5.12E−06)
Basic multi-modal functions	F6	1.16E+01 (3.16E+01)−	2.65E+01 (2.32E+01)−	5.28E+01 (4.15E+01)−	6.44E−01 (1.24E+00)+	2.32E−01 (6.57E−01)+	2.64e−27 (1.32e−26)+	9.65E+00 (4.65E+00)
	F7	8.27E−03 (8.22E−03)−	1.14E−02 (7.28E−03)−	1.65E−02 (1.58E−02)−	1.58E−02 (2.54E−02)−	7.39E−03 (6.45E−03)−	2.17e−03 (4.29e−03)≈	2.36E−03 (1.15E−03)
	F8	2.09E+01 (1.68E - 01)≈	2.09E+01 (4.54E−01)≈	2.09E+01 (3.54E−01)≈	2.09E+01 (2.84E−01)≈	2.01E+01 (125E−01)+	2.05e+01 (3.39e−01)+	2.09E+01 (5.87E−01)
	F9	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)
	F10	2.42E+01 (5.44E+00)−	5.46E+01 (8.85E+00)−	4.76E+01 (1.26E+01)−	5.24E+01 (4.64E+01)−	4.21E+01 (2.84E+01)−	1.62e+01 (3.35e+00)−	1.52E+01 (2.98E+00)
	F11	2.57E+01 (2.21E+00)≈	2.88E+01 (2.61E+00)−	1.68E+01 (1.64E+00)+	3.77E+01 (6.22E+00)−	1.24E+01 (3.55E+00)+	2.71e+01 (1.57e+00)−	2.58E+01 (3.11E+00)
	F12	6.45E+03 (2.89E+03)−	8.23E+03 (8.54E+03)−	3.44E+03 (4.42E+03)−	3.67E+04 (5.66E+03)−	3.21E+03 (4.48E+03)−	2.90e+03 (3.11e+03)−	1.17E+03 (8.66E+02)
Expanded multimodal functions	F13	1.47E+00 (1.15E−01)+	1.67E+00 (1.56E−01)+	3.84E+00 (2.66E−01)−	2.04E+00 (2.12E−01)+	1.66E+00 (3.25E−01)+	1.15e+00 (9.33e−02)+	2.92E+00 (6.33E−01)
	F14	1.23E+01 (3.21E−01)≈	1.30E+01 (2.23E−01)−	1.26E+01 (2.83E−01)−	1.35E+01 (2.35E−01)−	1.23E+01 (3.56E−01)≈	1.25e+01 (3.67e−01)−	1.23E+01 (4.22E−01)
Hybrid composition functions	F15	3.61E+02 (2.24E+02)≈	3.75E+02 (5.34E+01)≈	3.85E+02 (4.42E+01)≈	2.12E+02 (2.25E+01)+	3.86E+02 (5.24E+01)≈	3.72e+02 (9.62e+01)≈	3.78E+02 (6.32E+01)
	F16	9.33E+02 (1.31E+02)−	7.64E+01 (3.16E+01)−	8.65E+01 (5.65E+01)−	1.18E+02 (8.25E+01)−	7.25E+01 (6.22E+01)−-	9.79e+01 (1.35e+02)−	3.77E+01 (5.22E+00)
	F17	1.21E+02 (1.08E+02)−	1.24E+02 (4.82E+01)−	8.15E+01 (3.46E+01)−	1.42E+02 (1.15E+02)−	7.16E+01 (2.35E+01)−	1.71e+02 (1.58e+02)−	4.36E+01 (6.35E+00)
	F18	9.04E+02 (1.24E+00)≈	9.04E+02 (1.21E+01)≈	8.73E+02 (5.44E+01)≈	8.24E+02 (5.84E+00)+	9.04E+02 (1.34E+00)≈	9.04e+02 (9.88e−01)≈	9.04E+02 (1.21E+00)
	F19	9.04E+02 (8.32E−01)≈	9.04E+02 (1.32E+00)≈	8.74E+02 (6.34E+01)+	8.31E+02 (4.25E+00)+	9.05E+02 (3.22E−00)−	9.04e+02 (8.06e−01)≈	9.04E+02 (1.24E+00)
	F20	9.04E+02 (7.65E−01)≈	9.04E+02 (1.24E+00)≈	8.81E+02 (5.22E+01)≈	8.26E+02 (3.44E+00)+	9.04E+02 (6.42E−01)≈	9.04e+02 (6.84e−01)≈	9.04E+02 (1.18E+00)
	F21	5.00E+02 (4.67E−13)≈	5.00E+02 (4.72E−13)≈	5.45E+02 (2.15E+02)−	8.35E+02 (1.21E+02)−	5.00E+02 (4.68E−13)≈	5.00e+02 (1.68e−13)≈	5.00E+02 (3.54E−14)
	F22	8.68E+02 (2.24E+01)≈	8.78E+02 (2.23E+01)≈	9.21E+02 (2.66E+01)−-	5.07E+02 (5.54E+00)+	8.78E+02 (3.54E+01)≈	8.67e+02 (2.23e+01)≈	8.72E+02 (2.98E+01)
	F23	5.48E+02 (8.62E+01)−	5.34E+02 (2.42E−04)≈	5.34E+02 (8.27E−04)≈	8.63E+02 (4.81E+01)−	5.34E+02 (4.45E−04)≈	5.34e+02 (9.99e−05)≈	5.34E+02 (3.87E−04)
	F24	2.00E+02 (2.12E−14)≈	2.00E+02 (3.05E−14)≈	2.00E+02 (8.54E−14)≈	2.13E+02 (1.68E+00)−	2.00E+02 (2.62E−14)≈	2.00e+02 (8.58e−13)≈	2.00E+02 (2.21E−14)
	F25	2.11E+02 (7.35E−01)−	2.11E+02 (5.11E−01)−	2.14E+02 (2.35E+00)−	2.13E+02 (2.86E+00)−	2.11E+02 (6.82E−01)−	2.11e+02 (9.97e−01)−	2.09E+02 (3.32E−01)
−		9	13	15	14	11	8	
+		3	1	2	7	4	5	
≈		13	11	8	4	10	12	

In summary, MPEDE has the best overall performance compared with other five competitors, namely JADE, jDE, SaDE, EPSDE, CoDE and SHADE on all the 25 benchmark functions with 30 variables. Actually, the results of Wilcoxon's rank sum tests reported in the last three rows indicate that MPEDE is significantly better than JADE, jDE, SaDE, EPSDE, CoDE and SHADE on 9, 13, 15, 14, 11 and 8 functions, respectively. It is significantly worse than JADE, jDE, SaDE, EPSDE, CoDE and SHADE on 3, 1, 2, 7, 4 and 5 functions and similar to them on 13, 11, 8, 4, 10 and 12 functions, respectively. It can be seen from Table 1 that MPEDE is superior to all other peer algorithms on benchmark functions $F3$, $F10$, $F12$, $F16$, $F17$ and $F25$ with 30 variables.

From the data about benchmark functions with 50 variables given in Table 2, some observations can be made.

For unimodal functions, MPEDE and SHADE show better overall performance than other DE variants. In addition, JADE is also competitive. MPEDE outperforms JADE, jDE, SaDE, EPSDE, CoDE and SHADE on three ($F3$–$F5$), four ($F2$–$F5$), four ($F2$–$F5$), three ($F3$–$F5$), five ($F1$–$F5$) and two ($F4$ and $F5$) functions, respectively. jDE, SaDE and CoDE cannot outperform MPEDE on any of the considered unimodal functions. It is worth noting that JADE, EPSDE and SHADE surpass MPEDE on function $F2$, and SHADE generates the best results for function $F3$.

Table 2
Computational results of benchmark functions with 50 variables.

		JADE	jDE	SaDE	EPSDE	CoDE	SHADE	MPEDE
Unimodal functions	F1	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)≈	5.25E−29 (1.28E−28)−	8.07E−30 (4.04E−29)−	0.00E+00 (0.00E+00)≈	0.00E+00 (0.00E+00)
	F2	3.57E−27 (1.67E−27)+	2.22E−02 (3.21E−02)−	6.55E−02 (4.34E−02)	5.77E−23 (8.87E−23)+	5.87E−09 (6.51E−09)−	7.04E−27 (3.12E−27)+	8.45E−12 (5.66E−12)
	F3	1.41E+05 (6.61E+03)−	4.48E+05 (2.21E+05)−	8.01E+05 (5.44E+05)−	7.82E+06 (1.35E+07)−	3.22E+05 (8.66E+04)−	2.31E+04 (1.62E+04)+	6.46E+04 (3.15E+04)
	F4	2.79E+00 (1.26E+01)−	3.88E+02 (3.13E+02)−	7.28E+02 (4.56E+02)−	3.42E+03 (3.51E+04)−	4.98E+02 (5.78E+02)−	1.31E+00 (7.65E−01)−	1.28E+00 (5.22E−01)
	F5	1.45E+03 (4.38E+02)−	3.65E+03 (6.29E+02)−	8.34E+03 (1.29E+03)−	4.66E+03 (8.85E+02)−	3.60E+03 (5.69E+02)−	1.12E+03 (7.95E+02)−	4.69E+02 (1.22E+02)
Basic multi-modal functions	F6	7.71E+00 (3.03E+01)−	4.43E+02 (2.78E+01)−	4.39E+02 (2.52E+01)−	1.43E+02 (1.95E+00)−	1.23E+02 (2.12E+00)−	1.59E+00 (1.89E+00)≈	1.61E+00 (1.78E+00)
	F7	7.57E−03 (1.09E−02)−	2.85E−03 (6.10E−03)+	9.23E−03 (5.45E−02)−	1.08E−02 (1.79E−02)−	6.54E−03 (9.36E−03)−	4.92E−03 (8.60E−03)−	3.27E−03 (2.65E−03)
	F8	2.11E+01 (5.93E−02)≈	2.11E+01 (3.72E−02)≈	2.11E+01 (4.33E−02)≈	2.11E+01 (3.35E−02)≈	2.01E+01 (1.09E−01)+	2.07E+01 (3.12E−01)+	2.11E+01 (2.87E−02)
	F9	0.00E+00 (0.00E+00)+	0.00E+00 (0.00E+00)+	9.94E−01 (2.21E−01)−	6.39E−16 (1.01E−15)−	2.38E+00 (4.33E−01)−	7.10E−17 (3.55E−16)+	2.68E−07 (2.66E−8)
	F10	6.55E+01 (6.93E+00)−	9.98E+01 (1.32E+01)−	1.14E+02 (1.54E+01)−	1.54E+02 (2.53E+01)−	8.29E+01 (1.93E+01)−	3.75E+01 (5.78E+00)≈	3.83E+01 (4.54E+00)
	F11	5.24E+01 (2.15E+00)−	5.42E+01 (2.10E+00)−	4.45E+01 (1.89E+00)≈	7.04E+01 (3.21E+00)−	3.14E+01 (5.35E+00)+	5.56E+01 (2.21E+00)−	4.42E+01 (3.16E+00)
	F12	1.54E+04 (1.27E+04)−	1.57E+04 (1.54E+04)−	5.65E+04 (2.04E+04)−	3.16E+05 (3.85E+04)−	1.54E+04 (1.73E+04)−	9.67E+03 (8.94E+03)−	8.89E+03 (6.24E+03)
Expanded multimodal functions	F13	2.78E+00 (1.99E−01)+	2.94E+00 (2.41E−01)+	7.27E+00 (7.34E−01)−	6.17E+00 (6.03E−01)−	3.23E+00 (4.15E−01)+	2.13E+00 (1.15E−01)+	5.68E+00 (9.34E−01)
	F14	2.16E+01 (4.76E−01)+	2.25E+01 (2.26E−01)−	2.23E+01 (2.42E−01)−	2.34E+01 (2.63E−01)−	2.19E+01 (4.39E−01)≈	2.20E+01 (3.59E−01)−	2.19E+01 (2.19E−01)
Hybrid composition functions	F15	3.77E+02 (9.03E+01)−	3.62E+02 (1.21E+02)−	3.86E+02 (7.62E+01)−	2.64E+02 (6.45E+01)+	3.88E+02 (6.00E+01)−	3.21E+02 (9.55E+01)≈	3.12E+02 (8.51E+01)
	F16	8.42E+01 (7.92E+01)−	8.35E+01 (1.03E+01)−	8.78E+01 (6.57E+01)−	1.50E+02 (4.25E+01)−	9.35E+01 (7.01E+01)−	3.72E+01 (4.07E+00)−	3.39E+01 (5.66E+00)
	F17	9.39E+01 (2.64E+01)−	1.81E+02 (2.31E+01)−	9.81E+01 (1.01E+02)−	2.38E+02 (7.01E+01)−	7.21E+01 (2.58E+01)−	9.05E+01 (3.11E+01)−	3.62E+01 (3.11E+01)
	F18	9.21E+02 (4.38E+00)−	9.20E+02 (3.35E+00)−	9.78E+02 (8.37E+01)−	8.53E+02 (2.42E+01)+	9.21E+02 (5.42E+00)−	9.19E+02 (4.25E+00)−	9.17E+02 (3.78E+00)
	F19	9.19E+02 (1.07E+01)−	9.20E+02 (2.88E+00)−	9.78E+02 (7.81E+01)−	8.59E+02 (1.54E+01)+	9.21E+02 (4.64E+00)−	9.19E+02 (3.98E+00)−	9.17E+02 (3.15E+00)
	F20	9.21E+02 (3.38E+00)−	9.20E+02 (3.14E+00)−	9.55E+02 (3.54E+01)−	8.56E+02 (3.03E+00)+	9.11E+02 (3.38E+01)+	9.19E+02 (4.66E+00)−	9.18E+02 (6.55E+00)
	F21	5.52E+02 (1.49E+02)−	7.21E+02 (2.55E+02)−	5.66E+02 (2.32E+02)−	7.29E+02 (2.82E+00)−	6.83E+02 (2.49E+02)−	6.41E+02 (2.32E+02)−	5.00E+02 (2.87E−12)
	F22	9.05E+02 (2.48E+01)−	9.05E+02 (1.23E+00)−	9.82E+02 (8.21E+01)−	5.00E+02 (6.61E−02)+	9.01E+02 (2.18E+01)−	8.96E+02 (1.90E+01)≈	8.98E+02 (3.01E+01)
	F23	5.82E+02 (1.31E+02)−	8.60E+02 (2.25E+02)−	5.98E+02 (7.29E+00)−	7.33E+02 (4.48+00)−	7.10E+02 (2.33E+02)−	7.48E+02 (2.39E+02)−	5.39E+02 (3.41+00)
	F24	2.00E+02 (1.49E−12)≈	2.00E+02 (1.62E−12)≈	2.89E+02 (6.55E+01)−	2.38E+02 (1.34E+01)−	2.00E+02 (5.81E−14)≈	2.31E+02 (1.57E+02)−	2.00E+02 (1.65E−12)
	F25	2.18E+02 (1.71E+00)−	2.16E+02 (1.41E+00)−	2.24E+02 (1.25E+01)−	2.47E+02 (1.87E+01)−	2.17E+02 (1.97E+00)−	2.17E+02 (1.78E+00)−	2.14E+02 (1.08E+00)
	−	18	19	22	18	19	15	
	+	4	3	0	6	4	5	
	≈	3	3	3	1	2	5	

For basic multi-modal benchmark functions, the overall performance of MPEDE is superior to all comparative DE variants. Actually, MPEDE performs better than JADE, jDE, SaDE, EPSDE, CoDE and SHADE on five (*F*6, *F*7 and *F*10–*F*12), four (*F*6 and *F*10–*F*12), five (*F*6, *F*7, *F*9, *F*10 and *F*12), six (*F*6, *F*7 and *F*9–*F*12) five (*F*6, *F*7, *F*9, *F*10 and *F*12) and three (*F*7, *F*11 and *F*12) functions, respectively. In contrast, MPEDE is inferior to JADE, jDE, CoDE and SHADE on one (*F*9), two (*F*7 and *F*9), two (*F*8 and *F*11) and two (*F*8 and *F*9) functions, respectively. SaDE and EPSDE do not show better performance than MPEDE on any multi-modal benchmark functions tested.

JADE performs the best in solving both expanded multimodal functions. MPEDE surpasses jDE and SHADE on function *F*14 while performs worse than them on function *F*13. MPEDE is inferior to CoDE on function *F*13 while comparable to it on function *F*14. MPEDE beats SaDE and EPSDE on both test functions. In other words, MPEDE's performance is not satisfactory in solving *F*13 but competitive in solving *F*14.

MPEDE exhibits the best overall performance on hybrid composition functions. In fact, MPEDE is superior to JADE, jDE, SaDE, EPSDE, CoDE and SHADE on ten (*F*15–*F*23 and *F*25), ten (*F*15–*F*23 and *F*25), eleven (*F*15–*F*25), six (*F*16, *F*17, *F*21 and *F*23–*F*25),

nine ($F15$–$F19$, $F21$–$F23$ and $F25$) and nine ($F16$–$F21$ and $F23$–$F25$) functions, respectively. JADE, jDE, SaDE and SHADE cannot outperform MPEDE on any one of the hybrid composition functions. However, EPSDE shows better performance than MPEDE on functions $F15$, $F18$–$F20$ and $F22$.

Overall, although MPEDE does not rank the best in solving expanded multimodal functions, its comprehensive performance is the best on the whole 25 benchmark functions with 50 variables. The results of Wilcoxon's rank sum tests indicate that MPEDE is superior to JADE, jDE, SaDE, EPSDE, CoDE and SHADE on 18, 19, 22, 18, 19 and 15 functions, respectively. It is worse than JADE, jDE, SaDE, EPSDE, CoDE and SHADE on 4, 3, 0, 6, 4 and 5 functions while similar to them on 3, 3, 3, 1, 2 and 5 functions, respectively. It can be found that MPEDE is significantly better than all other comparative algorithms on functions $F4$, $F5$, $F12$, $F16$, $F17$, $F21$, $F23$ and $F25$ with 50 variables.

It is worth noting that, in our experiments, the overall performance of MPEDE is the best, however no comparative algorithm can completely beat all other peer ones on all benchmark functions. Generally, different comparative algorithms may have their particular advantages on different benchmark functions. For example, EPSDE generally outperforms or at least as competitive as other peer algorithms on functions $F15$, $F18$–$F20$ and $F22$. Moreover, MPEDE outperforms all other peer algorithms on functions $F12$, $F16$, $F17$ and $F25$ for both 30 and 50 variables.

The superiority of MPEDE to other comparative algorithms is more apparent when the number of decision variables in the test functions increases to 50. This indicates that the multi-population based multi-strategy ensemble approach has potential in addressing problems with more complex landscapes resulted from an increase in the number of decision variables.

The reasons why MPEDE performs better than other comparative algorithms can be explained as follows. First, evidences have shown that one mutation strategy may be efficient in searching landscapes of some specific problems while inefficient in others. Therefore, the mixture of multiple well-studied mutation strategies can support one another in solving different kinds of optimization problems and improve the overall performance of MPEDE. Furthermore, multiple mutation strategies are able to help MPEDE to sample points in the complex landscape of a problem via different ways, thereby increasing the probability of finding the optimal solution. Second, compared to other DE variants with multiple mutation strategies (e.g. SaDE, CoDE and EPSDE), MPEDE adopts a simple but efficient multi-population based ensemble approach that dynamically allocates more computational resources (i.e. population) to the recent best performed mutation strategy. It is known that, in SaDE and EPSDE, each individual changes its search behaviors in a gradual manner. However, MPEDE is more reactive as it gets feedback on the search experience quickly and gives the best performed mutation strategy more computational resources directly.

5.3. Evolution of mutation strategies and parameters

It would be interesting to find out which mutation strategy is the most frequently used for each benchmark function and how the parameters are adapted. We select six benchmark functions as representative ones (i.e. functions *F4, F8, F12, F16, F20* and *F24*) and show changes of the portion (probability) of each mutation strategy being the best mutation strategy and the parameter adaptation during the evolutionary process. From Fig. 1, we can observe that mutation strategies 1 and 2 generally dominate the evolutionary process. Mutation 3 usually exerts least effect on the function optimization processes. In addition, mutation 1 tends to perform best at the start while it is outperformed by mutation 2 rapidly with MPEDE proceeding. The proportion of each mutation strategy being the best varies with different problems or even different evolutionary stages of one problem,

As the mutation 2 ("current-to-rand/1") is not controlled by a crossover operator, we plot the crossover rate changes for mutation 1 and 3. It can be observed that different mutation strategies may require significantly differing crossover rates. For example, when solving functions *F12* (Fig. 1-c2) and *F24* (Fig. 1-f2), the appropriate crossover rate of mutation 3 is close to zero, that of mutation 1 is close to 1 and around 0.5 respectively. This phenomenon stresses that when designing DE variants with multiple mutation strategies, it is necessary to independently set parameters for each mutation strategy. In addition, we can find that one mutation generally needs different crossover rates to solve different optimization problems more efficiently. For instance, the most proper crossover rate for mutation 1 solving functions *F4* (Fig. 1-a2) and *F12* (Fig. 1-c2) is close to 1 while for solving functions *F16* (Fig. 1-d2) and *F20* (Fig. 1-e2), it is close to zero. Similar situations also happen if we look at the crossover rate changes of mutation 3 when solving different benchmark functions.

With regard to scaling factor, we have three main observations. First, the proper scaling factor values of these three different mutation strategies are different when solving one optimization problem. For example, when solving function *F12* (Fig. 1-c3), the proper scaling factor values of mutation strategies 1 and 2 are around 0.5, while that of mutation 3 is around 0.8. Second, the proper scaling factor values of one mutation strategy are different when solving different optimization problems. Take mutation strategy 2 as an example, its best scaling factor values for solving function *F8* (Fig. 1-b3) is around 0.8. In contrast, the best scaling factor values for mutation 2 solving function *F12* gradually stabilize at 0. 5. Third, the appropriate scaling factor value for one mutation strategy solving one optimization problem may change during the evolutionary process. For instance, there are obvious changes on the scaling factor values of mutation strategy 2 when solving functions *F8* (Fig. 1-b3), *F16* (Fig. 1-d3) and *F20* (Fig. 1-e3).

5.4. Parameter analysis

There are three tunable parameters in MPEDE, including the population size and two newly introduced ones, namely the ratio λ_1 (as $\lambda_1 = \lambda_2 = \lambda_3$) between indicator population and whole population, and generation gap *ng* for determining the recent best performing mutation strategy periodically. We analyze the impacts of parameters λ_1, *ng* and *NP* on the performance of MPEDE.

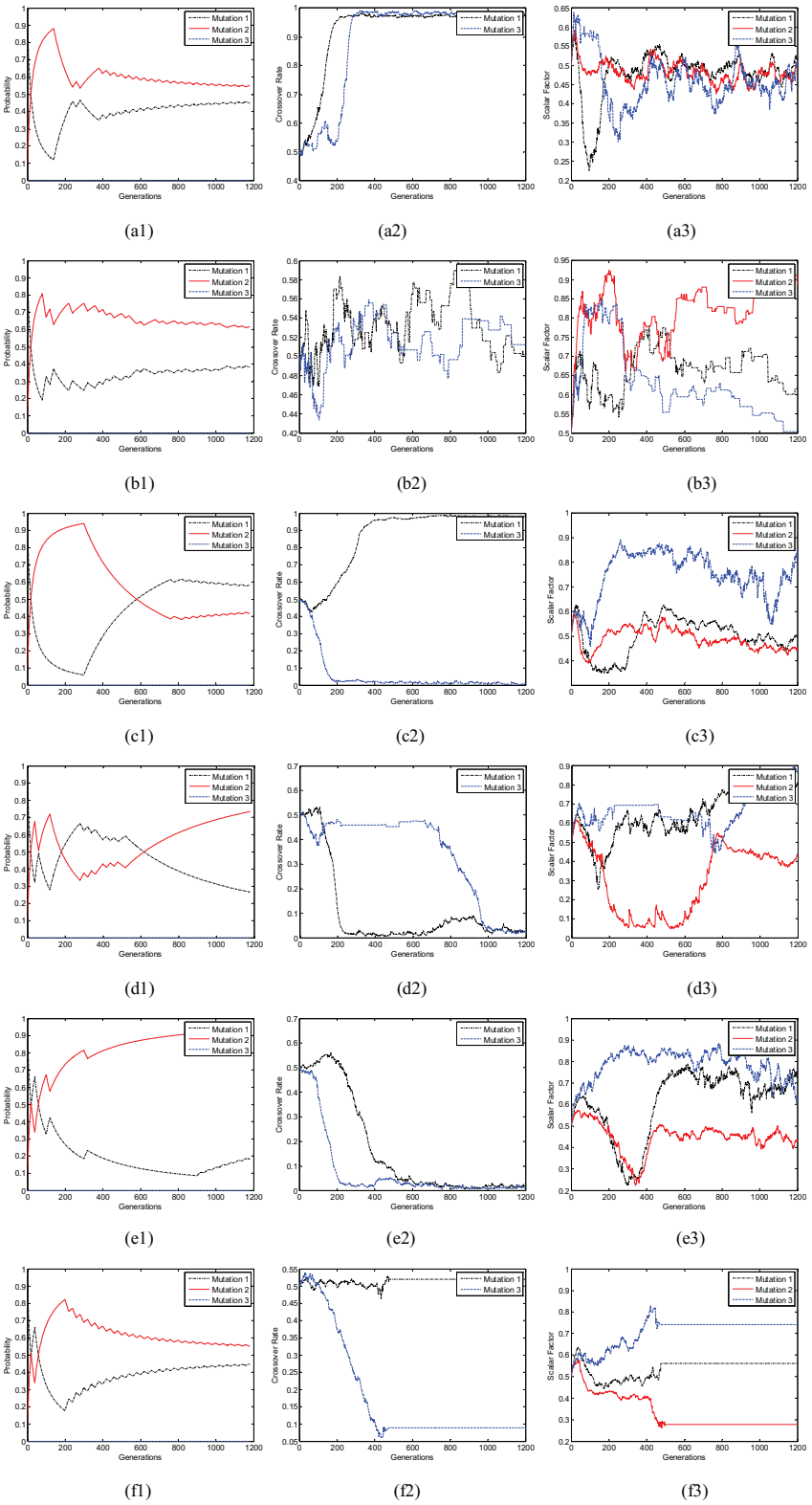

Fig. 1. Best mutation strategy portion and parameter evolution of six representative functions. (a1), (a2) and (a3) are the evolution of portion being best mutation strategy crossover rate and scaling factor of function *F4*, respectively; (b1), (b2) and (b3) are related to function *F8*; (c1), (c2) and (c3) are related to function *F12;* (d1), (d2) and (d3) are related to function *F16;* (e1), (e2) and (e3) are related to function *F20;* (f1), (f2) and (f3) are related to function *F24*.

Table 3
Computational results of MPEDE with different λ_1 and *ng* settings over benchmark functions with 30 variables. Best results of each function are highlighted.

	MPEDE $\lambda_1 = 0.1$	MPEDE $\lambda_1 = 0.15$	MPEDE $\lambda_1 = 0.25$	MPEDE $\lambda_1 = 0.3$	MPEDE $ng = 10$	MPEDE $ng = 30$	MPEDE $ng = 50$	MPEDE $ng = 80$	MPEDE Standard
F1	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00
F2	5.39E−24−	4.59E−26≈	1.19E−24−	5.45E−23−	4.65E−26≈	9.89E−26≈	5.92E−26≈	2.13E−25−	1.01E−26
F3	4.03E+01−	1.04E+01≈	1.43E+02−	1.76E+01−	1.49E+01−	3.09E+01−	4.22E+01−	1.37E+01−	1.01E+01
F4	8.58E−07−	3.43E−12−	5.80E−16≈	2.22E−15−	1.05E−14−	9.16E−16≈	8.32E−16≈	4.81E−13−	6.61E−16
F5	5.32E−04−	7.82E−06≈	4.85E−05−	1.71E−05−	1.07E−05−	8.24E−06≈	7.45E−05−	6.41E−05−	7.21E−06
F6	1.05E+01−	9.89E+00≈	2.85E−01+	6.41E+00+	8.22E+00+	8.07E+00+	1.19E+01−	1.54E+01−	9.65E+00
F7	3.77E−03−	4.92E−03−	4.43E−03−	4.45E−03−	3.59E−03−	2.29E−03≈	2.31E−03≈	3.95E−03−	2.36E−03
F8	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01
F9	0.00E+00≈	0.00E+00≈	9.90E−09−	4.25E−09−	2.70E−08−	0.00E+00≈	1.74E−07−	2.12E−07−	0.00E+00
F10	2.05E+01−	2.08E+01−	1.81E+01−	1.78E+01−	1.60E+01−	1.64E+01−	1.74E+01−	1.87E+01−	1.52E+01
F11	2.64E+01−	2.79E+01−	2.78E+01−	2.83E+01−	2.70E+01−	2.70E+01−	2.78E+01−	2.71E+01−	2.58E+01
F12	1.86E+03−	1.26E+03≈	2.20E+03−	2.39E+02−	1.52E+03−	1.38E+03−	1.48E+03−	2.22E+03−	1.17E+03
F13	2.88E+00≈	2.95E+00≈	2.95E+00≈	3.03E+00≈	2.90E+00≈	2.97E+00≈	2.98E+00≈	3.00E+00≈	2.92E+00
F14	1.25E+01−	1.24E+01−	1.25E+01−	1.26E+01−	1.25E+01−	1.25E+01−	1.25E+01−	1.25E+01−	1.23E+01
F15	3.80E+02≈	3.91E+02−	3.86E+02≈	3.93E+02−	4.13E+02−	3.93E+02−	3.86E+02≈	3.80E+02≈	3.78E+02
F16	6.64E+01−	4.62E+01−	3.88E+01≈	3.90E+01−	6.27E+01−	3.85E+01−	4.31E+01−	4.96E+01−	3.77E+01
F17	4.94E+01−	4.87E+01−	7.60E+01−	6.75E+01−	4.70E+01−	5.33E+01−	6.02E+01−	6.59E+01−	4.36E+01
F18	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02
F19	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02
F20	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02
F21	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02
F22	8.68E+02≈	8.69E+02≈	8.64E+02≈	8.67E+02≈	8.66E+02≈	8.61E+02≈	8.71E+02≈	8.66E+02≈	8.72E+02
F23	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02
F24	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02
F25	2.09E+02≈	2.09E+02≈	2.09E+02≈	2.09E+02≈	2.09E+02≈	2.09E+02≈	2.09E+02≈	2.09E+02≈	2.09E+02
−	12	9	9	13	12	8	10	13	
+	0	0	1	1	1	1	0	0	
≈	13	16	15	11	12	16	15	12	

In the parameter sensitivity analyses, the candidate values for λ_1 include 0.1, 0.15, 0.25, and 0.3, candidate values for *ng* include 10, 30, 50, and 80 and candidate values for *NP* include 50, 100, 150, 200, 250, 300 and 400. When one parameter is analyzed, other parameters are set to the default values (i.e. $NP = 250$, $\lambda_1 = 0.2$ and $ng = 20$). MPEDE with default parameter values here is called standard MPEDE. We also perform Wilcoxon's rank sum test at a 0.05 significance between standard MPEDE and other MPEDE versions of different parameter values. "−", "+", and "≈" mean that the related MPEDE versions is worse than, better than, and similar to standard MPEDE. The sensitivity analysis results of parameters λ_1 and *ng* are recorded in Table 3 and that of parameter *NP* are listed in Table 4.

From the data given in Table 3, it can be observed that MPEDE is not sensitive to parameters λ_1 and *ng* on many of the benchmark functions, including *F1, F8, F13* and *F18–F25*. In addition, MPEDE versions with other parameter values seldom outperform the standard MPEDE, which indicates the reasonable parameter setting of the standard MPEDE. It is worth noting that, however, MPEDE with $\lambda_1 = 0.15$ surpasses standard MPEDE on function *F6*.

Table 4 provides the analysis results of parameter *NP*. In contrast with parameters λ_1 and *ng*, we can find that parameter *NP* exerts higher impact on the performance of MPEDE. Some interesting phenomena can be observed. First, MPEDE is not sensitive to parameter *NP* in solving problems *F9*, *F13*, and *F21–F25*. Second, MPEDE shows better performance with population size increase in solving problems *F3, F7* and *F25*. Third, MPEDE's performance improves with population size decrease in solving problems *F6, F11, F13* and *F14*. Statistical results show that standard MPEDE with $NP = 250$ obtains the overall best performance.

It was traditionally thought that larger population is good for diversity maintenance of DE and thus is good for the solution of multi-modal optimization problems, while smaller population is suggested in dealing with unimodal optimization problems. However, our experiments reveal that this is not always the case. For example, when dealing with unimodal problem *F4,* the performance of MPEDE degrades rapidly with the population size being less than 150. In addition, MPEDE can generate much better solution for unimodal function *F3* if we increase its population size. Moreover, evidences show that smaller population size is beneficial to MPEDE in solving multi-modal problems *F11, F13* and *F14*. As a result, the appropriate setting of *NP* actually is not tightly related to whether the targeted problems are unimodal or multi-modal. Current parameter adaptation methods mainly focus on crossover rate and scalar factor of DE. The automated configuration for population size of DE deserves more investigations.

5.5. Comparisons between MPEDE and its variants with different mutation strategies

MPEDE consists of three different mutation strategies. To show the efficiency of the multi-population based ensemble of mutation strategies, we compare MPEDE with its variants derived by setting mutation strategies differently. MPEDE-M1,

Table 4
Computational results of MPEDE with different population sizes over benchmark functions with 30 variables. Best results of each function are highlighted.

	MPEDE Pop 50	MPEDE Pop 100	MPEDE Pop 150	MPEDE Pop 200	MPEDE Pop 300	MPEDE Pop 400	MPEDE Standard
F1	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00 ≈	0.00E+00≈	1.36E−27−	0.00E+00
F2	9.67E−27≈	9.32E−27≈	1.65E−26≈	8.28E−27≈	5.23E−25−	1.43E−21−	1.01E−26
F3	1.24E+04−	2.08E+04−	9.03E+03−	9.57E+02−	1.43E−10+	1.09E−11+	1.01E+01
F4	3.26E−01−	4.24E−05−	1.17E−09−	2.24E−13−	1.62E−15−	5.73E−13−	6.61E−16
F5	2.48E+02−	2.41E−03−	4.93E−05−	4.54E−05−	2.98E−05−	4.21E−04−	7.21E−06
F6	1.20E+00+	8.34E−01+	1.02E+00+	6.02E+00+	8.81E+00≈	1.19E+01−	9.65E+00
F7	1.32E−02−	6.98E−03−	8.17E−03−	4.43E−03−	2.32E−03≈	9.85E−04+	2.36E−03
F8	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+00≈	2.09E+01
F9	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00≈	3.17E−02−	7.26E+00−	0.00E+00
F10	3.45E+01−	2.26E+01−	2.18E+01−	1.90E+01−	1.82E+01−	3.88E+01−	1.52E+01
F11	2.19E+01+	2.18E+01+	2.31E+01+	2.50E+01≈	2.84E+01−	2.94E+01−	2.58E+01
F12	2.24E+03−	3.39E+03−	1.85E+03−	1.89E+03−	1.15E+03≈	2.13E+03−	1.17E+03
F13	1.30E+00+	1.62E+00+	2.18E+00+	2.51E+00+	3.19E+00−	3.78E+00−	2.92E+00
F14	1.20E+01+	1.19E+01+	1.22E+01+	1.23E+01≈	1.26E+01−	1.28E+01−	1.23E+01
F15	3.72E+02≈	3.72E+02≈	3.56E+02≈	3.65E+02≈	3.84E+02≈	4.10E+02−	3.78E+02
F16	1.82E+02−	1.08E+02−	4.51E+01−	3.97E+01≈	5.24E+01−	5.27E+01−	3.77E+01
F17	1.67E+02−	6.95E+01−	4.59E+01−	4.23E+01≈	5.63E+01−	9.93E+01−	4.36E+01
F18	9.06E+02−	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02
F19	9.05E+02−	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02
F20	9.05E+02−	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02≈	9.04E+02
F21	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02
F22	8.80E+02≈	8.70E+02≈	8.66E+02≈	8.71E+02≈	8.68E+02≈	8.63E+02≈	8.72E+02
F23	6.15E+02−	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02
F24	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02
F25	2.12E+02−	2.10E+02−	2.10E+02−	2.09E+02≈	2.09E+02≈	2.09E+02≈	2.09E+02
−	13	9	9	6	11	12	
+	4	4	4	2	2	2	
≈	8	12	12	17	12	11	

MPEDE-M2, MPEDE-M3 are DE with "current-to-pbest/1", "current-to-rand/1" and "rand/1", respectively. They can be thought as special MPEDE variants with a single mutation strategy. MPEDE-MR denotes an MPEDE variant in which three mutation strategies are randomly selected. Note that the *Cr* and *F* parameter adaptation strategies are employed in each MPEDE variant.

We can find from Table 5 that the standard MPEDE outperforms its other variants noticeably. In fact, statistical tests reveal that MPEDE is superior to MPEDE-M1, MPEDE-M2, MPEDE-M3 and MPEDE-MR on 9, 18, 20 and 13 functions respectively, and inferior to them on 3, 2, 1 and 0 functions respectively, and equal to them on 13, 5, 4 and 12 functions, respectively. These results clearly show the advantage of MPEDE caused by incorporating three mutation strategies and assigning different population resources to each mutation strategy dynamically. It is worth noting that MPEDE-M1 show competitive performance on three unimodal functions F2, F4 and F5 and two multi-modal functions F13 and F14. In addition, MPEDE-M2 and MPEDE-M3 obtain much better results for F15 than the other variants.

6. An application to economic load dispatch problems

To test the efficiency of MPEDE in dealing with real-world optimization problems, we apply MPEDE to the economic load dispatch (ELD) problem.

6.1. Formulation of the ELD problem

The ELD problem is about minimizing the fuel cost of generating units for a specific period of operation, usually 1 h of operation, so as to accomplish optimal generation dispatch among operating units and in return satisfying the system load demand, generator operation constraints with ramp rate limits and prohibited operating zones.

The objective function corresponding to the production cost is represented as:

$$\text{Minimize} : f = \sum_{i=1}^{N_G} f_i(P_i) \tag{20}$$

where,

$$f_i(P_i) = a_i P_i^2 + b_i P_i + c_i + \left| e_i \sin\left(f_i\left(P_i^{\min} - P_i\right)\right)\right|, \quad i = 1, 2, 3, \ldots, N_G \tag{21}$$

is the expression for cost function corresponding to ith generating unit and a_i, b_i and c_i are its coefficients.

Table 5
Comparison results between MPEDE and its variants derived by different setting of mutation strategies. Best results of each function are highlighted.

	MPEDE–M1	MPEDE–M2	MPEDE–M3	MPEDE–MR	MPEDE
F1	0.00E+00≈	9.29E−17−	4.12E−08−	0.00E+00≈	0.00E+00
F2	1.26E−28+	2.81E+02−	1.23E+04−	7.07E−24−	1.01E−26
F3	8.42E+03−	6.87E+05−	2.07E+07−	1.94E+01−	1.01E+01
F4	4.13E−16≈	3.63E−03−	1.79E+04−	1.17E−15−	6.61E−16
F5	7.59E−08+	3.51E+03−	4.69E+03−	2.38E−04−	7.21E−06
F6	1.16E+01−	3.07E+01−	9.53E+01−	1.01E+01−	9.65E+00
F7	8.27E−03−	4.07E−01−	1.44E+00−	4.01E−03−	2.36E−03
F8	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01
F9	0.00E+00≈	2.08E+00−	1.00E−05−	6.41E−09−	0.00E+00
F10	2.42E+01−	6.86E+01−	1.10E+02−	1.73E+01−	1.52E+01
F11	2.57E+01≈	2.80E+01	2.78E+01	2.55E+01≈	2.58E+01
F12	6.45E+03−	3.16E+04−	3.96E+04−	1.87E+03−	1.17E+03
F13	1.47E+00+	2.69E+00+	3.06E+00−	3.10E+00−	2.92E+00
F14	1.23E+01≈	1.28E+01−	1.30E+01−	1.26E+01−	1.23E+01
F15	3.61E+02≈	1.59E+02+	2.90E+02+	4.12E+02−	3.78E+02
F16	9.33E+02−	1.57E+02−	1.42E+02−	3.85E+01≈	3.77E+01
F17	1.21E+02−	1.92E+02−	1.91E+02−	5.66E+01−	4.36E+01
F18	9.04E+02≈	9.08E+02−	9.09E+02−	9.04E+02≈	9.04E+02
F19	9.04E+02≈	9.08E+02−	9.09E+02−	9.04E+02≈	9.04E+02
F20	9.04E+02≈	9.08E+02−	9.09E+02−	9.04E+02≈	9.04E+02
F21	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02≈	5.00E+02
F22	8.68E+02≈	8.91E+02≈	9.62E+02−	8.63E+02≈	8.72E+02
F23	5.48E+02−	5.34E+02≈	5.34E+02≈	5.34E+02≈	5.34E+02
F24	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02≈	2.00E+02
F25	2.11E+02−	2.10E+02−	2.33E+02−	2.09E+02≈	2.09E+02
−	9	18	20	13	
+	3	2	1	0	
≈	13	5	4	12	

P_i is the real power output (in MW) of ith generator corresponding to time period t.
N_G is the number of online generating units to be dispatched.
e_i and f_i are the cost coefficients corresponding to valve point loading effect.

Various constraints need to be satisfied which are discussed as below.

Power balance constraint: This constraint is based on the principle of equilibrium between total system generation and total system loads (P_D) and losses (P_L). That is,

$$\sum_{i=1}^{N_G} P_i = P_D + P_L \tag{22}$$

where P_L is obtained using B- coefficients, given by

$$P_L = \sum_{i=1}^{N_G}\sum_{j=1}^{N_G} P_i B_{ij} P_j + \sum_{i=1}^{N_G} B_{0i} P_i + B_{00} \tag{23}$$

Capacity constraints: The output power of each generating unit has a lower and upper bound so that it lies in between these bounds. This constraint is represented by a pair of inequality constraints as follows:

$$p_i^{\min} \le P_i \le P_i^{\max} \tag{24}$$

where $p_i^{\min}$ and $P_i^{\max}$ are lower and upper bounds for power outputs of the ith generating unit.

Ramp rate limits: Increasing or decreasing the output generation of each unit is restricted to an amount of power over a time interval due to the physical limitations of each unit. The generator ramp rate limits change the effective real power operating limits as follows:

$$\max\left(p_i^{\min}, P_i(t-1) - DR_i\right) \le P_i(t) \le \min\left(P_i^{\max}, P_i(t-1) - UR_i\right) \tag{25}$$

where $P_i(t-1)$ is the output power of generator i in the previous dispatch.

Prohibited operating zones (POZ): Modern generators with valve point loading have many prohibited operating zones. Therefore, in practical operation, when adjusting the generation output P_i of unit i, the operation of the unit in the prohibited zones must be avoided. The feasible operating zones of the unit i can be described as follows:

Table 6
Comparison of optimization results of the instance with 6 units.

Method	Minimum	Average	Maximum
MPEDE	**1.5444e+04**	**1.5444e+04**	**1.5444e+04**
SOH–PSO [9]	1.5446e+04	1.5497e+04	1.5610e+04
NPSO–LRS [44]	1.5450e+04	NA	NA
GA [44]	1.5451e+04	NA	NA
GAAPI [11]	1.5450e+04	15450e+04	1.5450e+04
SA–PSO [26]	1.5447e+04	1.5447e+04	1.5455e+04

Table 7
Comparison of optimization results of the instance with 15 units.

Method	Minimum	Average	Maximum
MPEDE	**3.2692e+04**	**3.2693e+04**	**3.2693e+04**
SOH–PSO [9]	3.2751e+04	3.2878e+04	3.2945e+04
GAAPI [11]	3.2733e+04	32735e+04	32756e+04
SA–PSO [26]	32708e+04	32747e+04	32807e+04
FA [62]	32705e+04	32856e+04	33175e+04

$$\begin{aligned} & P_i^{\min} \le P_i \le P_{i,1}^{LB} \\ & P_{i,j}^{LB} \le P_i \le P_{i,j}^{LB}, \quad j = 2, 3, \ldots, NP_i \\ & P_{i,NP_i}^{UB} \le P_i \le P_i^{\max} \end{aligned} \tag{26}$$

where NP_i is the number of prohibited zones of unit i.

6.2. Experimental results

The ELD problem is a constrained optimization problem. Therefore, it is necessary to combine MPEDE with a constraint handling technique. Several constraint handling techniques have been presented in literature, such as penalty function, feasible rules, stochastic ranking and ε-constrained method [34]. We adopt the idea of variable reduction to tackle equality power balance constraint [58]. In brief, at every generation of MPEDE, we calculate the value of power output P_1 through formula (27), which is derived from (22).

$$P_1 = (-b + \sqrt{b^2 - 4ac})/2a, \tag{27}$$

where,

$$a = B_{11}, \tag{28}$$

$$b = \sum_{i=2} B_{1,i} \cdot P_i + \sum_{j=2} P_j \cdot B_{j,1} + B_{01} - 1, \tag{29}$$

$$c = \sum_{i=2} \sum_{j=2} P_i B_{ij} P_j + \sum_{i=2} P_i B_{oi} + B_{00} + P_D - \sum_{i=2} P_i. \tag{30}$$

In addition, the inequality constraints are addressed by following penalty function.

$$F(x) = f(x) + \alpha_1 \phi_1(x) + \alpha_2 \phi_2(x) + \alpha_3 \phi_3(x), \tag{31}$$

where, $f(x)$ is the power cost, and $\phi_1(x)$, $\phi_2(x)$ and $\phi_3(x)$ denote capacity constraints, ramp rate limits and prohibited operating zones constraint, respectively. Penalty coefficients α_1, α_2 and α_3 are set to 1e+3, 1e+5 and 1e+5, respectively.

Two instances with 6 and 15 units are solved by MPEDE, respectively. The computational results of MPEDE are compared with those reported in recent literature. Details are listed in Tables 6 and 7, respectively. The symbol "NA" in tables means that the related data are not reported in the reference. From the data recorded in Table 6 and Table 7, it can be observed that MPEDE is able to find the best results for the considered ELD instances. This means that MPEDE is a good alterative algorithm for dealing with ELD problems and may have potential in dealing with general real-world optimization problems.

7. Conclusions

Mutation strategies can significantly affect the performance of differential evolution (DE) on different optimization problems. In addition, different mutation strategies with different advantages can complement one another even when DE is applied to

one optimization problem. As a result, an efficient integration of different mutation strategies is a promising way to enhance the performance DE. In this paper, a multi-population based approach is proposed to realize the adapted ensemble of three mutation strategies into a novel DE variant, i.e. MPEDE. The population of MPEDE is dynamically partitioned into several subpopulations including three indicator subpopulations and one reward subpopulation as the algorithm proceeds. Each indicator subpopulation with relatively smaller size is assigned to a constituent mutation strategy and the reward subpopulation with relatively lager size is assigned to the currently best performed mutation strategy as an extra reward. Through this manner, we effectively realize the dynamic computation resource allocation among mutation strategies and the best mutation strategy is expected to timely obtain most computational resources (given with the reward subpopulation). In addition to the adaptive ensemble of multiple mutation strategies, the control parameters of each mutation strategy are adapted independently. Extensive experiments on CEC 2005 benchmark suit show that MPEDE outperforms several other efficient and popular peer DE variants including JADE, jDE, SaDE, EPSDE, CoDE and SHADE.

Experimental analyses show that the appropriate control parameters required by different mutation strategies are generally different. Two new parameters are brought to MPEDE, namely the ratio between indicator population and whole population, and the generation gap for periodically determining the recently best performed mutation strategy. We experimentally show that MPEDE is not sensitive to these two new parameters on majority of benchmark functions.

In our future work, we plan to apply MPEDE to solve additional real-world optimization problems to further test its performance. In addition, the multi-population based approach can be a general ensemble framework. It is meaningful to apply it to other evolutionary algorithms (EAs), such as particle swarm optimization (PSO) and Biogeography-based optimization (BBO).

Acknowledgment

This work was partly supported by the National Natural Science Foundation of China under grant nos. 71271213, 41571397 and 51178193.

References

[1] H.A. Abbass, The self-adaptive pareto differential evolution algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, 2002, pp. 831–836.
[2] A. Al-Ani, A. Alsukker, R.N. Khushaba, Feature subset selection using differential evolution and a wheel based search strategy, Swarm Evol. Comput. 9 (2013) 15–26.
[3] M.Z. Ali, N.H. Awad, P.N. Suganthan, Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization, Appl. Soft Comput. 33 (2015) 304–327.
[4] M.Z. Ali, M. Pant, A. Abraham, Improved differential evolution algorithm with decentralisation of population, Int. J. Bio-Insp. Comput. 3 (2011) 17–30.
[5] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, IEEE Trans.Cybern. 44 (2014) 1726–1737.
[6] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657.
[7] H. Cai, C. Chung, K. Wong, Application of differential evolution algorithm for transient stability constrained optimal power flow, IEEE Trans. Power Syst. 23 (2008) 719–728.
[8] U.K. Chakraborthy, S. Das, A. Konar, Differential evolution with local neighborhood, in: Proceedings of Congress on Evolutionary Computation, IEEE Press, Vancouver, BC, Canada, 2006, pp. 2042–2049.
[9] K.T. Chaturvedi, M. Pandit, L. Srivastava, Self-organizing hierarchical particle swarm optimization for nonconvex economic dispatch, IEEE Trans. Power Syst. 23 (2008) 1079–1087.
[10] J. Cheng, G. Zhang, F. Neri, Enhancing distributed differential evolution with multicultural migration for global numerical optimization, Inform. Sci. 247 (2013) 72–93.
[11] I. Ciornei, E. Kyriakides, A GA-API solution for the economic dispatch of generation in power system operation, IEEE Trans. Power Syst. 27 (2012) 233–242.
[12] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (2009) 526–553.
[13] S. Das, A. Konar, U.K. Chakraborty, Two improved differential evolution schemes for faster global search, in: Conference on Genetic and Evolutionary Computation, 2005, pp. 991–998.
[14] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (2011) 4–31.
[15] R. Dash, P.K. Dash, R. Bisoi, A self adaptive differential harmony search based optimized extreme learning machine for financial time series prediction, Swarm Evol. Comput. (2014) 25–42.
[16] R. Gämperle, S.D. Müller, P. Koumoutsakos, A parameter study for differential evolution, Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, WSEAS Press, Interlaken, Switzerland, 2002, pp. 293–298.
[17] W. Gao, G.G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Trans. Cybernet. 44 (2014) 1314–1327.
[18] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Trans. Cybernet. 43 (2013) 2066–2081.
[19] W. Gong, Z. Cai, C.X. Ling, C. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 41 (2011) 397–413.
[20] W. Gong, A. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Inform. Sci. 181 (2011) 5364–5386.
[21] Gong, W., Zhou, A., Cai, Z., A multi-operator search strategy based on cheap surrogate models for evolutionary optimization, IEEE Trans. Evol. Comput., DOI:10.1109/TEVC.2015.2449293.
[22] H. Guo, Y. Li, J. Li, H. Sun, D. Wang, X. Chen, Differential evolution improved with self-adaptive control parameters based on simulated annealing, Swarm Evol. Comput. (2014) 52–67.
[23] Guo, S., Yang, C., Hsu, P., Tsai, J., Improving differential evolution with successful-parent-selecting framework, IEEE Trans. Evol. Comput., doi:10.1109/TEVC.2014.2375933.
[24] A. Iorio, X. Li, Solving rotated multi-objective optimization problems using differential evolution, in: Austr. Conf. Artif. Intellig., Australia, Cairns, 2004, pp. 861–872.
[25] G. Jia, Y. Wang, Z. Cai, Y. Jin, An improved (μ+λ)-constrained differential evolution for constrained optimization, Inform. Sci. 222 (2013) 302–322.
[26] C.-C. Kuo, A novel coding scheme for practical economic dispatch by modified particle swarm approach, IEEE Trans. Power Syst. 23 (2008) 1825–1835.

[27] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Sixth International Mendel Conference on Soft Computing, 2000, pp. 76–83.
[28] Y.-I. Li, J. Zhang, A new differential evolution algorithm with dynamic population partition and local restart, in: Gecco-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference, 2011, pp. 1085–1092.
[29] J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings 2005 IEEE Swarm Intelligence Symposium, IEEE, 2005, pp. 124–129.
[30] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Comput. 9 (2005) 448–462.
[31] R. Mallipeddi, P.N. Suganthan, Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies, in: B.K. Panigrahi, S. Das, P.N. Suganthan, S.S. Dash (Eds.), Swarm, Evolutionary, and Memetic Computing, 2010, pp. 71–78.
[32] R. Mallipeddi, P.N. Suganthan, Ensemble of constraint handling techniques, IEEE Trans. Evol. Comput. 14 (2010) 561–579.
[33] R. Mallipeddi, P.N. Suganthan, Q.-K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679–1696.
[34] E. Mezura-Montes, C.A.C. Coello, Constraint-handling in nature-inspired numerical optimization: past, present and future, Swarm Evol. Comput. 1 (2011) 173–194.
[35] P. Novoa-Hernandez, C. Cruz Corona, D.A. Pelta, Self-adaptive, multipopulation differential evolution in dynamic environments, Soft Comput. 17 (2013) 1861–1881.
[36] M.G.H. Omran, A. Salman, A.P. Engelbrecht, Self-adaptive differential evolution, Computational Intelligence and Security, Lecture Notes in Artificial Intelligence, 2005, pp. 192–199.
[37] N. Pholdee, S. Bureerat, Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses, Inform. Sci. 223 (2013) 136–152.
[38] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, Berlin, 2005.
[39] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417.
[40] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution, IEEE Trans. Evol. Comput. 12 (2008) 64–79.
[41] J. Ronkkonen, S. Kukkonen, K.V. Price, Real-parameter optimization with differential evolution, in: IEEE Congress of Evolutionary Computation (CEC 2005), 2005, pp. 506–513.
[42] R.A. Sarker, S.M. Elsayed, T. Ray, Differential evolution with dynamic parameters selection for optimization problems, IEEE Trans. Evol. Comput. 18 (2014) 689–707.
[43] S. Sayah, A. Hamouda, A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems, Appl. Soft Comput. 13 (2013) 1608–1619.
[44] A.I. Selvakumar, K. Thanushkodi, A new particle swarm optimization solution to nonconvex economic dispatch problems, IEEE Trans. Power Syst. 22 (2007) 42–51.
[45] R. Shang, Y. Wang, J. Wang, L. Jiao, S. Wang, L. Qi, A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem, Inform. Sci. 277 (2014) 609–642.
[46] R. Storn, On the usage of differential evolution for function optimization, in: Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS), IEEE, Berkeley, 1996, pp. 519–523.
[47] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optimiz. 11 (1997) 341–359.
[48] R. Storn, K. Price, Differential evolution: a simple evolution strategy for fast optimization, Dr. Dobbs J. 22 (1997) 18–24.
[49] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, KanGAL Report, 2005005, 2005.
[50] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2013, pp. 71–78.
[51] Tang, L., Dong, Y., Liu, J., Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput., doi:10.1109/TEVC.2014.2360890.
[52] J. Wang, J. Liao, Y. Zhou, Y. Cai, Differential evolution enhanced with multiobjective sorting-based mutation operators, IEEE Trans. Cybernet. 44 (2014) 2792–2805.
[53] R. Wang, P.J. Fleming, R.C. Purshouse, General framework for localised multi-objective evolutionary algorithms, Inform. Sci. 258 (2014) 29–53.
[54] R. Wang, R.C. Purshouse, P.J. Fleming, Preference-inspired coevolutionary algorithms for many-objective optimization, IEEE Trans. Evol. Comput. 17 (2013) 474–494.
[55] R. Wang, R.C. Purshouse, P.J. Fleming, Preference-inspired co-evolutionary algorithms using weight vectors, Eur. J. Oper. Res. 243 (2015) 423–441.
[56] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66.
[57] Y. Wang, H.-X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[58] Wu, G., Pedrycz, W., Suganthand, P.N., Mallipeddie, R., A variable reduction strategy for evolutionary algorithms handling equality constraints, Appl. Soft Comput., doi:10.1016/j.asoc.2015.09.007.
[59] G. Wu, D. Qiu, Y. Yu, W. Pedrycz, M. Ma, H. Li, Superior solution guided particle swarm optimization combined with local search techniques, Exp. Syst. Appl. 41 (2014) 7536–7548.
[60] G.Y. Yang, Z.Y. Dong, K.P. Wong, A modified differential evolution algorithm with fitness sharing for power system planning, IEEE Trans. Power Syst. 23 (2008) 514–522.
[61] M. Yang, C. Li, Z. Cai, J. Guan, Differential evolution with auto-enhanced population diversity, IEEE Trans. Cybernet. 45 (2015) 302–315.
[62] X.-S. Yang, S.S. Sadat Hosseini, A.H. Gandomi, Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect, Appl. Soft Comput. 12 (2012) 1180–1186.
[63] W.-J. Yu, M. Shen, W.-N. Chen, Z.-H. Zhan, Y.-J. Gong, Y. Lin, O. Liu, J. Zhang, Differential evolution with two-level parameter adaptation, IEEE Trans. Cybernet. 44 (2014) 1080–1099.
[64] W.-j. Yu, J. Zhang, Multi-population differential evolution with adaptive parameter control for global optimization, in: Gecco-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference, 2011, pp. 1093–1098.
[65] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: Proceedings of the 9th International Conference on Soft Computing, Brno, 2003, pp. 41–46.
[66] D. Zaharie, Influence of crossover on the behavior of differential evolution algorithms, Appl. Soft Comput. 9 (2009) 1126–1138.
[67] J. Zhang, S. A. C., JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.
[68] J. Zhang, X. Ding, A multi-swarm self-adaptive and cooperative particle swarm optimization, Eng. Appl. Artif. Intellig. 24 (2011) 958–967.
[69] S.-Z. Zhao, S. P. N., Q. Zhang, Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Trans. Evol. Comput. 16 (2012) 442–446.
[70] S.-Z. Zhao, P.N. Suganthan, Q.-K. Pan, M. Fatih Tasgetiren, Dynamic multi-swarm particle swarm optimizer with harmony search, Exp. Syst. Appl. 38 (2011) 3735–3742.
[71] S.Z. Zhao, P.N. Suganthan, Empirical investigations into the exponential crossover of differential evolutions, Swarm Evol. Comput. 9 (2013) 27–36.
[72] Y.-J. Zheng, X.-L. Xu, H.-F. Ling, S.-Y. Chen, A hybrid fireworks optimization method with differential evolution operators, Neurocomputing 148 (2015) 75–82.
[73] W. Zhu, Y. Tang, J.-a. Fang, W. Zhang, Adaptive population tuning scheme for differential evolution, Inform. Sci. 223 (2013) 164–191.