Information Sciences 423 (2018) 172–186

Contents lists available at ScienceDirect

Information Sciences

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

Ensemble of differential evolution variants

Guohua Wu[a,b,*], Xin Shen[c,d], Haifeng Li[e], Huangke Chen[b], Anping Lin[f], P.N. Suganthan[g]

[a] School of Mathematics and Big Data, Foshan University, Foshan 528000, P.R. China
[b] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, Hunan, P.R. China
[c] State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, P.R. China
[d] Collaborative Innovation Centre of Geospatial Technology, Wuhan 430079, China
[e] School of Geosciences and Info-Physics, Central South University, Changsha 410004, P.R. China
[f] College of Electrical and Information Engineering, Hunan University, Changsha, 410082, P.R. China
[g] School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore

ARTICLE INFO

Article history:
Received 24 March 2017
Revised 31 July 2017
Accepted 18 September 2017
Available online 21 September 2017

Keywords:
Evolutionary algorithm
Differential evolution
Multi-population
Ensemble of differential evolution variants
Numerical optimization

ABSTRACT

Differential evolution (DE) is one of the most popular and efficient evolutionary algorithms for numerical optimization and it has gained much success in a series of academic benchmark competitions as well as real applications. Recently, ensemble methods receive an increasing attention in designing high-quality DE algorithms. However, previous efforts are mainly devoted to the low-level ensemble of mutation strategies of DE. This study investigates the high-level ensemble of multiple existing efficient DE variants. A multi-population based framework (MPF) is proposed to realize the **e**nsemble of multiple **DE v**ariants to derive a new algorithm named EDEV for short. EDEV consists of three highly popular and efficient DE variants, namely JADE (adaptive differential evolution with optional external archive), CoDE (differential evolution with composite trial vector generation strategies and control parameters) and EPSDE (differential evolution algorithm with ensemble of parameters and mutation strategies). The whole population of EDEV is partitioned into four subpopulations, including three indicator subpopulations with smaller size and one reward subpopulation with much larger size. Each constituent DE variant in EDEV owns an indicator subpopulation. After every predefined generations, the most efficient constituent DE variant is determined and the reward subpopulation is assigned to that best performed DE variant as an extra reward. Through this manner, the most efficient DE variant is expected to obtain the most computational resources during the optimization process. In addition, the population partition operator is triggered at every generation, which results in timely information sharing and tight cooperation among the component DE variants. Extensive experiments and comparisons have been done based on the CEC2005 and CEC2014 benchmark suit, which shows that the overall performance of EDEV is superior to several state-of-the-art peer DE variants. The success of EDEV reveals that, through an appropriate ensemble framework, different DE variants of different merits can support one another to cooperatively solve optimization problems.

* Corresponding author.
E-mail addresses: guohuawu@nudt.edu.cn (G. Wu), xshen@whu.edu.cn (X. Shen), lihaifeng@csu.edu.cn (H. Li), hkchen@nudt.edu.cn (H. Chen), anping719@126.com (A. Lin), EPNSugan@ntu.edu.sg (P.N. Suganthan).

https://doi.org/10.1016/j.ins.2017.09.053
0020-0255/

1. Introduction

Differential evolution (DE), initially introduced by Storn and Price [1], is among the most popular evolutionary algorithms (EAs) currently in use due to its simplicity and efficiency in dealing with complex optimization problems. DE is a populated stochastic search algorithm, which incorporates mutation, crossover and selection operators to move the population gradually toward the global optimum [2]. The significant difference between DE and other EAs is that it exploits solutions differences in mutation operator. DE has experienced noticeable progress since its inception [3]. Many performance enhanced DE variants has been proposed via the design of novel mutation and crossover strategies [4–10], the maintenance of population diversity [11,12], the adaptation of parameters [13–18], the hybridization with other EAs [19–22], and the intelligent combination of multiple search strategies [2,13,23,24]. DE is recognized as an efficient search engine for different optimization domains, such as constrained optimization [25], multi-objective optimization [26] and multimodal optimization [27]. In addition, DE has been widely applied to many real-world optimization problems from various fields, such as pattern recognition [28], power systems optimization [29], time series prediction [30], vehicle routing[31] and feature selection[32].

Many evidences reveal that the most appropriate mutation strategies and parameters of DE are generally different when it is applied to different optimization problems [13]. This is because different mutation and parameter settings enable DE to possess different exploitation and exploration capabilities [3]. Besides, previous studies further showed that the required mutation strategies and parameters may even vary during the evolution process when solving one optimization problem [24]. Nevertheless, traditional trial-and-error approaches for configuring strategies and parameters are usually tedious, time-consuming and ineffective [23]. In response to this challenge, several approaches have been presented for intelligent mutation strategy ensemble and automated parameter adaptation [2,13,23,24,27,33,34]. Qin et al. [13] 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. [35] 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 [23], 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. Gong et al. [36] 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. Zhao et al. proposed an ensemble of different neighborhood sizes with online self-adaptation to enhance the multiobjective evolutionary algorithm based on decomposition [37].

From a higher level perspective, different DE variants exhibit different capabilities in solving different optimization problems. Actually, the No-free-lunch theorem tells that all search algorithms will averagely own the same performance when they are applied to all potential optimization problems, that is to say, theoretically, there will not exist a general optimization algorithm being superior to all others [38,39]. Many popular and efficient DE variants have been put forward during last two decades. There is no doubt that these DE variants have their respective advantages in dealing with different optimization problems. As a result, it is meaningful yet challenging to make full use of the advantages of different DE variants to derive an even better DE.

In reality, the idea of mixing multiple search components or EAs has attracted much attention from researchers for a long time. In addition to ensemble algorithm, some other related concepts have been presented, such as memetic algorithm [40], hybrid algorithm [41–43], hyper-heuristics [44,45] and algorithm portfolios [46].

In this study, we focus on the high-level ensemble of different DE variants. A multi-population based framework (MPF) is proposed to combine multiple DE variants together and allocate computational resources to constituent DE variants dynamically in terms of their respective historical performance. This framework is inspired from our previous work [24], in which an ensemble of multiple mutation strategies of DE was implemented to derive a new DE variant named MPEDE, which showed very competitive performance. Two types of subpopulations are included in the MPF, namely, indicator subpopulation and reward subpopulation. Three highly popular and efficient DE variants, including JADE [15], EPSDE [23] and CoDE [2], are incorporated into MPF. Each of these three constituent DE variants has an indicator subpopulation. That is to say, there are total three indicator subpopulations in MPF. In addition, one reward subpopulation is contained in MPF (the overall population minus the three indicator subpopulations). Generally, the sizes of the indicator subpopulations are equal and much smaller than the reward subpopulation. During the evolutionary process, after every certain number of generations, the best performed constituent DE variant is determined in terms of the performance measured by the fitness improvements divided by consumed function evaluations. The reward subpopulation then is assigned to the current best performed DE variant, which is supposed to be able to continue the performance dominance in the near future evolutionary process. The best performing DE variant determination and the reward subpopulation reallocation operators are executed periodically. Hence, MPF ensures that the best DE variant can gain the most computational resources timely.

The paper is structured as follows. Section 2 reviews related works. Section 3 introduces the proposed ensemble of DE variants, including its framework, constituent DE variants and exploration-exploitation mechanisms. Section 4 reports related experimental studies. Section 5 concludes this work.

2. Related works

2.1. Differential evolution

Differential evolution (DE) is a competitive evolutionary algorithm (EA) and particularly it shows outstanding performance in solving complex numerical optimization problems. The population of DE, with *NP* individuals, utilize three operators, i.e., mutation, crossover and selection, at each generation to search for the solution of an optimization problem. In DE, a solution is coded as a vector [1]. At each generation, the three operators are executed on each vector (called target vector here). A mutant vector is obtained after mutation operator. And then a trial vector can be produced by performing crossover operator between a mutant vector and its associated target vector. The selection operator guarantees the survival of the better one between the target vector and the trial vector to the next generation.

At generation G, the ith vector (solution) is represented as $\mathbf{X}_{i,G} = \{x^1_{i,G}, x^2_{i,G}, \cdots, x^D_{i,G}\}$, $i = 1, 2, \cdots, NP$, where D denotes the dimensionality of the optimization problem being solved. Corresponding to a target vector $\mathbf{X}_{i,G}$, a mutant vector is generated using one of following six most popular mutation strategies.

"rand/1:"

$$\mathbf{V}_{i,G} = \mathbf{X}_{r1,G} + F \cdot (\mathbf{X}_{r2,G} - \mathbf{X}_{r3,G}) \tag{1}$$

"rand/2:"

$$\mathbf{V}_{i,G} = \mathbf{X}_{r1,G} + F \cdot (\mathbf{X}_{r2,G} - \mathbf{X}_{r3,G}) + F \cdot (\mathbf{X}_{r4,G} - \mathbf{X}_{r5,G}) \tag{2}$$

"best/1:"

$$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r1,G} - \mathbf{X}_{r2,G}) \tag{3}$$

"best/2:"

$$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r1,G} - \mathbf{X}_{r2,G}) + F \cdot (\mathbf{X}_{r3,G} - \mathbf{X}_{r4,G}) \tag{4}$$

"current-to-best/1:"

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot (\mathbf{X}_{best,G} - \mathbf{X}_{r1,G}) + F \cdot (\mathbf{X}_{r2,G} - \mathbf{X}_{r3,G}) \tag{5}$$

"current-to-pbest/1:"

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot (\mathbf{X}_{p,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r1,G} - \mathbf{X}_{r2,G}) \tag{6}$$

In the equations above, $r1$, $r2$, $r3$, $r4$ and $r5$ and exclusive integer numbers ranging from 1 to *NP* and they are different from i. F is a parameter called the scale factor that controls the magnitude of the difference vectors. $\mathbf{X}_{best,G}$ is the best individual in the current population.

After mutation, crossover operation is performed between the target vector $\mathbf{X}_{i,G}$ and the mutant vector $\mathbf{V}_{i,G}$ to produce a trial vector $\mathbf{U}_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \cdots, u^D_{i,G}\}$. The binomial crossover operator is mostly used and it is defined as:

$$u^j_{i,G} = \begin{cases} v^j_{i,G}, & if \quad rand_j(0,1) \le Cr \quad or \quad j = j_{rand} \\ x^j_{i,G}, & otherwise \end{cases} \tag{7}$$

where $i = 1, 2, \cdots, NP$, $j = 1, 2, \cdots, D$. $Cr \in [0, 1]$ is parameter called crossover rate, which controls how many components of the trial vector $\mathbf{U}_{i,G}$ are inherited from target vector $\mathbf{X}_{i,G}$ and $\mathbf{V}_{i,G}$, respectively. $rand_j(0, 1)$ is random value uniformly distributed between 0 and 1. j_{rand} is a random integer from 1 to D, which guarantee the necessary difference (at least on one dimension) between target vector $\mathbf{X}_{i,G}$ and trial vector $\mathbf{U}_{i,G}$.

Once the trial vector is produced, the selection operator will make comparison between the target vector $\mathbf{X}_{i,G}$ and the trial vector $\mathbf{U}_{i,G}$ and choose the superior one to survive to the next generation. The selection operator is performed as below:

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

The basic search operations of a traditional DE at each generation are described above. Many successive studies investigated the trial vector (i.e. $\mathbf{U}_{i,G}$) generation strategies and parameter (i.e. population size *NP*, scale factor F and crossover rate Cr) control mechanisms in depth.

In addition to mutation and crossover operators introduced above, some other novel ones were also proposed, such as trigonometric mutation operator [47], current-to-rand/1 mutation operator [48], current-to-pbest/1 mutation operator [15], ranking-based mutation operator [8,49], directional mutation operator [50], eigenvector-based crossover operator [51], multiobjective sorting-based mutation operators [52], hybrid linkage crossover [53] and covariance matrix learning based crossover operator [5].

The performance of DE is highly depended on the appropriate configuration of its parameters, including population size *NP*, scale factor F and crossover rate Cr. Although much work has been done on parameter tuning of DE, it can be observed

that various and even inconsistent conclusions have been drawn in earlier studies [3,54]. The main reason is that the most appropriate parameters required by DE for solving different optimization problems are different.

So, it is natural when researchers use different optimization problems to analyze the parameters of DE, they may come to different parameter setting suggestions. In general, the traditional trial-and-error parameter tuning approach is time consuming and inefficient. As a result, research on adaptive or self-adaptive parameter controlling methods draws intensive attention [55]. Several parameter adaptation approaches have been designed, including fuzzy logic control [56], population diversity based parameter control [57] and simulated annealing based parameter control [58].

Brest et al. encoded the parameters into solutions and evolve them simultaneously [14]. Evidences show that adapting DE parameters according to their previous experiences in generating promising solutions can strengthen the performance of DE effectively [13,15]. Recently, a two-level parameter adaptation scheme [17] and dynamic parameter combination selection mechanism [16] were proposed. It is found that DE algorithm generally is very sensitive to the population size [59,60]. It is impressive that the DE with success-history based parameter adaptation and linear population reduction have gained much success [61,62]. To comprehensively know the latest progress of DE, several survey papers are recommended [3,63,64].

2.2. Ensemble algorithms

Now it is widely accepted that the best search strategies and parameters of an EA are generally different in solving different optimization problems. Therefore, in order to improve the performance of an EA like DE, researchers make efforts to realize an ensemble of multiple strategies and parameters [2,3,23,65]. Qin et al. [13] proposed a self-adaptive DE algorithm (SaDE) in which multiple mutation strategies are dynamically selected by each individual based on their previous experiences of generating promising solutions. Mallipeddi et al. [23] presented a DE variant 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. EPSDE records successful strategy and parameter combinations and select them with a higher probability in successive generations. Other ensemble schemes were also developed for constraint handling [65], neighbourhood size of decomposition based multiobjective EA [37] and strategies of evolutionary programming (EP) [66].

Gong et al. [35] 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 according to previous search performance. In another paper, Gong et al. [36] proposed a cheap surrogate model for the ensemble of multiple search operators in evolutionary optimization. Jia et al. [67] 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. Recently, a multi-population ensemble DE (mDE-bES) was introduced to solve large-scale global optimization problems [68]. In mDE-bES, the sizes of subpopulations are equal and constant. Each subpopulation has a different mutation and update strategy. After every certain number of generations, individuals between different subpopulations can be exchanged to share search information.

In addition to DE, the ensemble of multi-strategies has also been applied to other EAs, such as particle swarm optimization (PSO) [69–71], artificial bee colony (ABC) [72] and biogeography-based optimization (BBO) [73].

The literature discussed above mainly involves the low-level ensemble of strategies and parameters of EAs. Studies have been done on the high-level combination of different EAs. Some interrelated terms include memetic algorithm [40], hyper-heuristics [44,45] and algorithm portfolios [46].

The term of memetic algorithm (MA) is originated from Moscato [74] as being a paradigm for integrating EAs with one or more refinement methods [40]. In MA, population based EAs (e.g DE, PSO and ABC) are taken as global search techniques while refinement methods including deterministic and stochastic (*e.g Nelder-Mead simplex search method, Hill Climber with Sidestep and trust-region derivative-free methods*) play the role of local search. To pursue the well balance between exploration and exploitation capabilities of MA, simple hybrid, adaptive hybrid and memetic automation approaches were investigated [40,75,76].

A hyper-heuristic is a methodology which, when given a particular problem instance or a class of instances, and a number of low-level heuristics (or their components), automatically produces an adequate combination of the provided components to effectively solve the given problem(s) [44]. Burke et. al categorized hyper-heuristic into heuristic selection and heuristic generation [77]. It can be perceived that hyper-heuristic aims to automatically develop new search algorithms on the basis of a pool of low-level heuristics.

In response to the challenge that the performance of an algorithm may vary significantly from problem to problem, algorithm portfolio attempts to find a way being less risk to distribute the time among multiple different algorithms [46,78]. Peng et. al proposed a population-based algorithm portfolio (PAP) framework to implement the synergy of different EAs. In that framework, initial population is divided into multiple subpopulations of the same size. After every certain number of generations, the individual immigration operator is triggered [46]. Ke Tang et al. further studied the constituent algorithm selection based on estimated performance matrix [79].

3. Ensemble of differential evolution variants

In this study a multi-population based framework (MPF) is proposed for the ensemble of multiple DE variants. Rather than PAP [46] which realizes algorithm portfolio through a time budget assignment strategy and an individual immigration

operator, MPF divides the whole population into sub-populations including several indicator sub-populations and one reward subpopulation. Indicator sub-populations have the same sizes and is much smaller than the reward subpopulation. Each constituent DE variant has an indicator subpopulation. As the ensemble algorithm proceeds, after every certain number of generations, the reward subpopulation is adaptively assigned to the DE variant that recently performed the best. Through such manner, different DE variants evolves cooperatively and the one performs the best during the evolutionary process will obtain the most resources (represented by populations).

3.1. Constituent DE variants

To reach a higher performance for EDEV, it is crucial to make sure that the constituent DE variants are powerful while have different capabilities so that they can support each other during the evolutionary process rather than just compete for resources. Many studies showed the importance to have different operators in one algorithm [80–82]. Here, three highly classical and efficient DE variants are taken as constituent algorithms, i.e. JADE [15], CoDE [2] and EPSDE [23]. It is quite impossible to try to take every DE variant into the ensemble framework. The reason why we choose these three algorithms as the constituent components is that empirical studies have shown that JADE is a versatile DE variants and it often dominates other DE variants in solving unimodal optimization problems, and CoDE is very effective in solving some simple multimodal optimization problems while EPSDE exhibits extraordinary performance in dealing with some highly complex composite optimization problems [2,24]. A brief introduction of these three DE variants are given as below.

- **JADE**

JADE, initially developed by Zhang and Sanderson, is a simple yet efficient DE variants [15]. In JADE, a novel current-to-pbest/1 mutation strategy is employed. The “current-to-pbest/1” mutation strategy is shown as below.

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F_i \cdot (\mathbf{X}_{pbest,G} - \mathbf{X}_{r1,G}) + F_i \cdot (\mathbf{X}_{r2,G} - \mathbf{X}_{r3,G}) \tag{9}$$

Another version of “current-to-pbest/1” is incorporated with an archive **A** which contains some recently declared inferior solutions. Let **P** denote the current population. The $\widetilde{\mathbf{X}}_{r3,G}$ is from the union of **A** and **P**. The archived version of “current-to-pbest/1” is described as follows.

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F_i \cdot (\mathbf{X}_{pbest,G} - \mathbf{X}_{r1,G}) + F_i \cdot (\mathbf{X}_{r2,G} - \widetilde{\mathbf{X}}_{r3,G}) \tag{10}$$

A sophisticated parameter adaptation mechanism is exploited in JADE. Parameters F and Cr are generated by a normal distribution and Cauchy distribution for each target vector, respectively. Recently F and Cr values that contribute to successful generation of promising solutions are recorded and utilized for the new parameter generation and thus better control parameter values are propagated to the successive generations.

Evidences show that JADE have high performance in solving different kinds of problems [2,24].

- **CoDE**

CoDE, proposed by Wang et al., is a DE variant with composite trial vector generation strategies and control parameters [2]. These strategies and parameters have distinct advantages confirmed by existing studies. It can be recognized as a DE with low-level ensemble of mutation strategies and control parameters. In CoDE, three trial vector generation strategies namely “rand/1/bin”, “rand/2/bin” and “current-to-rand/1” and three control parameter combinations are employed, i.e. [F = 1.0, Cr = 0.1], [F = 1.0, Cr = 0.9] and [F = 0.8, Cr = 0.2]. At every generation, each of the three trial vector generation strategy is used to create a new trial vector with a control parameter setting randomly chosen from the parameter candidate pool. Consequently, three trial vectors are generated for each target vector and the best one survives for the next generation if it is better than its corresponding target vector.

CoDE have shown high performance in solving various kinds of optimization problems and especially its advantage in dealing with some multimodal optimization problems is noticeable [2].

- **EPSDE**

Initially designed by Mallipeddi et al., EPSDE is a popular DE variant with ensemble of mutation strategies and parameters [23]. In EPSDE, mutation strategy pool and parameter pool are constructed, respectively. ”DE/best/2/bin”, ”DE/rand/1/bin” and ”DE/current-to-rand/1/bin” are contained in the strategy pool. The pool of CR values is taken in the range 0.1-0.9 in steps of 0.1. And the pool of F values is taken in the range 0.4-0.9 in steps of 0.1. Each member in the initial population is assigned a mutation strategy and parameter values randomly selected from the respective pools. The mutation strategy and parameter values producing better offsprings survive while those fail to produce better offsprings are reinitialized.

Experimental studies have shown that, compared with other DE variants, EPSDE is particularly efficient in solving some highly complex problems (e.g. Hybrid Composition Functions) [2,23,24].

3.2. Multi-population based ensemble framework

In MPF, the overall population is divided into several indicator subpopulations (each belongs to a constituent DE variant) and one reward subpopulation. Three DE variants, namely JADE, CoDE and EPSDE, are incorporated in the MPF, thus

we partition the whole population into three indicator subpopulation and one reward population. The partition operator is triggered at every generation. The three indicator subpopulations are denoted by pop_1, pop_2 and pop_3 and the reward subpopulation is represented as pop_4. Indicator subpopulations have the equal size. And the size of a indicator subpopulation is much smaller than that of the reward subpopulation. Let pop be the overall population. We have

$$pop = \bigcup_{i=1,\cdots,4} pop_i \tag{11}$$

Let NP be the size of pop and NP_i be the size of pop_i. λ_i denotes the proportion between pop_i and pop. Thus we have

$$NP_i = \lambda_i \cdot NP \tag{12}$$

$$\sum_{i=1,\cdots,4} \lambda_i = 1 \tag{13}$$

Here we just let $\lambda_1 = \lambda_2 = \lambda_3$. Initially, each indicator subpopulation is randomly assigned to a constituent DE variant and the reward subpopulation is also randomly allocated to a DE variant. At every generation, the population partition operation is executed for one time. As the algorithm proceeds, after every ng number of generations, we determine the most efficient DE variant (i_{best}) during the last period according to ratio between accumulated fitness improvements and consumed function evaluations.

$$i_{best} = argmax_{i=1,2,3}\left(\frac{\Delta f_i}{\Delta fes_i}\right) \tag{14}$$

where, Δf_i is the accumulated function fitness improvements during the last ng generations attributed by the ith constituent DE variant and Δfes_i is the consumed number of function evaluations.

During the next ng generations, the reward subpopulation will be rewarded to the best performed constituent DE variant. The best performing DE variant determination and reward subpopulation assignment operators described above are performed periodically with ng being the period. With this idea, we ensure that the best mutation strategy consumes the most computational resources.

The algorithm framework of EDEV is described in Algorithm 1.

Algorithm 1 Algorithm Framework of EDEV.

Initial parameters of EDEV including ng, NP, λ_i, $MaxG$ and $MaxFes$;
Initial the parameters for JADE, CoDE and EPSDE;
Set $\Delta f_i = 0$ and Δfes_i for $i = 1, 2, 3$;
Initialize the pop randomly distributed in the solution space;
Set $NP_i = \lambda_i \cdot NP$;
Randomly divide pop into pop_1, pop_2, pop_3 and pop_4 with respect to their sizes;
Randomly select a subpopulation pop_i(i = 1, 2, 3) and combine pop_i with pop_4. Let $pop_i = pop_i \bigcup pop4$ and $NP_i = NP_i + NP_4$;
Set g = 0;
while $g \leq MaxG$ **do**
    $g = g + 1$;
    Execute JADE on pop_1, update pop_1 and calculate Δf_1;
    Execute CoDE on pop_2, update pop_2 and calculate Δf_2;
    Execute EPSDE on pop_3, update pop_3 and calculate Δf_3;
    Combine updated pop_1, pop_2 and pop_3 into pop, i.e., $pop = \bigcup_{i=1,2,3} pop_i$;
    if mod(g, ng) == 0 **then**
        $k = arg(max_{i=1,2,3}(\frac{\Delta f_i}{ng \cdot NP_i}))$;
    end if
    Randomly partition pop into pop_1, pop_2, pop_3 and pop_4;
    Let $pop_k = pop_k \bigcup pop_4$, $k \in \{1, 2, 3\}$;
end while

4. Experimental study

4.1. Experimental setting

To test the efficiency of EDEV, a suit of 25 well-benchmarked optimization functions proposed in the CEC 2005 special session on real-parameter optimization are utilized [83]. This benchmark suit includes five unimodal functions (F1-F5), six

Table 1
Parameter configuration of each tested algorithm.

Algorithm	parameter settings
EDEV	$\lambda_1 = \lambda_2 = \lambda_3 = 0.1,\ \lambda_4 = 0.7,\ ng = 20,\ NP = 50$ or $NP = 100$
JADE	$p = 0.05,\ c = 0.1,\ NP = 100$
jDE	$\tau_1 = \tau_2 = 0.1,\ F_l = 0.1,\ F_u = 0.9,\ NP = 100$
SaDE	$\varepsilon = 0.01, LP = 50,\ NP = 50$
EPSDE	CR range [0.1, 0.9] and F range [0.4, 0.9], $NP = 50$
CoDE	Three trial vector generation strategies and three control parameter settings, $NP = 30$
MPEDE	$\lambda_1 = 0.2,\ ng = 20,\ NP = 250$

basic multimodal functions (F6-F12), two expanded multimodal functions (F13-F14) and eleven hybrid composition functions (F15-F25). The parameter of EDEV used in experiments are set as $\lambda_1 = \lambda_2 = \lambda_3 = 0.1,\ \lambda_4 = 0.7,\ ng = 20$. In addition, $NP = 60$ for benchmark functions with 30 variables and $NP = 100$ for benchmark functions with 50 and 100 variables, which are configured empirically. The EDEV algorithm is coded using Matlab 2014 software with the system configuration of: Intel core i7-4810MQ 2.80 GHz CPU, 16 GB RAM and Windows 7 operating system.

EDEV was compared with six other state-of-the-art DE variants including JADE [15], jDE [84], SaDE [13], EPSDE [23], CoDE [2] and MPEDE [24]. Considering the stochastic property of each algorithm in nature, 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. The parameter configurations of the comparative algorithms are listed in Table 1.

The computational results for benchmark functions with 30 and 50 variables are reported in Table 2 and Table 3, respectively. Wilcoxons rank sum test at a 0.05 significance level was conducted between EDEV and each comparative algorithm on every benchmark function. Signs "-" and "+" indicate that one comparative algorithm is significantly worse and better than EDEV, respectively, while "$\approx$" means there is no significant difference between their performance.

4.2. Computational results of functions with 30 variables

Several observations can be obtained from the data reported in Table 2. First, with regard to the unimodal functions (F1-F5), we can find that JADE and MPEDE perform the best. The reason is that both JADE and MPEDE incorporate the "current-to-pbest/1" mutation strategy and a historical solution archive, which are fairly efficient in dealing with unimodal optimization problems as validated by [2,24,85]. Although EDEV does not employ "current-to-pbest/1" mutation strategy straightforwardly, it encapsulates JADE as its constituent algorithm. Hence, it is not a surprise that EDEV also exhibit promising performance in solving the unimodal benchmark functions. Actually, it performs better than jDE, SaDE, EPSDE and CoDE on all the unimodal benchmark functions except the simplest function F1 on which all comparative algorithms can generate the optimal solution. In addition, EDEV even get better solution than MPEDE on function F2. Computational results on unimodal functions tells that the constituent JADE variant takes effect in EDEV.

For the basic multimodal functions, the overall performance of EDEV and CoDE is the best. EDEV is superior to JADE, jDE, SaDE, EPSDE and MPEDE on four (F6, F8, F11 and F12), six (F6-F8 and F10-F12), six (F6-F8 and F10-F12), six (F6-F8 and F10-F12) and four (F6, F8, F11 and F12) functions, while comparable to them on two (F9 and F10), one (F9), one (F9), two (F6 and F9) and one (F9) functions, and inferior to them on one (F7), zero, zero, zero and two (F7 and F10) functions. EDEV is comparable to CoDE on basic multi-modal functions, with the fact that although EDEV performs worse than CoDE on functions F6 and F7, it dominates CoDE on functions F10 and F12. In addition, they obtain the similar results for functions F8, F9 and F11. CoDE has demonstrated good performance in tackling the basic multi-modal problems. As a result, it is safe to conclude that EDEV inherits the advantages of CoDE successfully.

As for the expanded multi-modal functions, the performance of EDEV is undoubtedly the best. No other comparative algorithm can beat EDEV on any of the considered expanded multi-modal functions. This phenomenon reemphasize that EDEV has been endowed with the power of CoDE.

EDEV shows the best performance in solving the eleven complex hybrid composition functions. It can be found that EDEV performs better than JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on six (F17-F20 and F22-F23), five (F17-F20 and F22), seven (F15, F18-F22 and F25), nine (F17-F25), five (F15, F18-F20 and F22) and five (F15,F18-F20 and F22) functions, respectively, equals to these comparative algorithms on four (F15, F16,F21 and F24), five (F16, F17, F21, F23 and F24), four (F16, F17, F23 and F24), two (F16 and F20), four (F17, F21, F23 and F24) and three (F21, F23 and F24) functions respectively, and is inferior to these six peer algorithms on one (F25), one (F25), zero, one (F15), two (F16 and F25) and three (F16, F17 and F25) functions, respectively. These achievements indicate that the ensemble of multiple DE variants are beneficial to the solution of complex optimization problems. This can be explained by following two reasons. First of all, different DE variants generally use different search strategies which enable EDEV to have richer sampling means to fit complex function landscapes. In addition, EDEV actually makes use of the previous research knowledge and experience of DE community, since all selected constituent DE variants in EDEV are well configured and validated by former researchers. They have different capabilities in solving different types of optimization problems. Hence they can supplement one another and cooperatively evolve during the problem optimization process, and finally yield a better EDEV algorithm.

Table 2
Computational Results of benchmark functions with 30 variables.

	JADE	jDE	SaDE	EPSDE	CoDE	MPEDE	EDEV
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)–	1.01E-26 (2.05E-26)–	3.39E-28 (1.22E-28)
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)–	1.01E+01 (8.32E+00)+	2.16E+04 (1.58E+04)
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)–	6.61E-16 (5.68E-16)+	2.81E-08 (3.24E-08)
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)–	7.21E-06 (5.12E-06)+	6.14E-02 (2.14E-02)
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)+	9.65E+00 (4.65E+00)–	4.97E-01 (5.65E-01)
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)+	2.36E-03 (1.15E-03)+	1.01E-02 (6.28E-03)
F8	2.09E+01 (8.22E-03)–	2.09E+01 (7.28E-03)–	2.09E+01 (1.58E-02)–	2.09E+01 (2.54E-02)–	2.01E+01 (6.45E-03)≈	2.09E+01 (1.15E-03)–	2.01E+01 (6.28E-03)
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)–	1.52E+01 (2.98E+00)+	2.51E+01 (6.54E+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)≈	2.58E+01 (3.11E+00)–	1.25E+01 (5.11E+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)–	1.17E+03 (8.66E+02)–	7.17E+02 (3.21E+02)
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)–	2.92E+00 (6.33E-01)–	1.09E+00 (8.21E-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)–	1.23E+01 (4.22E-01)–	1.21E+01 (5.21E-01)
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)–	3.78E+02 (6.32E+01)–	3.46E+02 (3.21E+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)+	3.77E+01 (5.22E+00)+	8.48E+01 (4.48E+01)
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)≈	4.36E+01 (6.35E+00)+	8.31E+01 (4.32E+01)
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 (1.21E+00)–	8.19E+02 (3.25E+00)
F19	9.04E+02 (1.24E+00)–	9.04E+02 (1.21E+01)–	8.74E+02 (5.44E+01))–	8.31E+02 (5.84E+00)–	9.05E+02 (1.34E+00)–	9.04E+02 (1.21E+00)–	8.23E+02 (3.25E+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 (1.18E+00)–	8.28E+02 (2.62E+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 (3.54E-14)≈	5.00E+02 (2.89E-13)
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)–	8.72E+02 (2.98E+01)–	5.05E+02 (2.44E+00)
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 (3.87E-04)≈	5.34E+02 (8.98E-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 (2.62E-14)≈	2.00E+02 (6.77E-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)+	2.09E+02 (3.32E-01)+	2.12E+02 (9.44E-01)
–	12	17	19	19	13	12	
+	5	1	0	1	4	8	
≈	8	7	6	5	8	5	

To summarise, EDEV exhibits very competitive overall performance in solving different kinds of CEC 2005 benchmark functions with 30 variables. Results of Wilcoxons rank sum tests given in the last three rows of Table 2 show that EDEV is significantly better than JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on 12, 17, 19, 19, 13 and 12 functions, respectively. It is inferior to JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on 5, 1, 0, 1, 4 and 8 functions and similar to them on 8, 7, 6, 5, 8 and 5 functions, respectively. It is worth noting that EDEV is able to generate the best solutions for functions F1, F2, F8, F9, F11-F14 and F18-F24.

4.3. Computational results of functions with 50 variables

Similarly, some interesting observations can be obtained from the data reported in Table 3. First, it is observed that the performance of EDEV ranks the best when solving the five unimodal benchmark functions of 50 variables. In more detail, EDEV outperforms JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on two (F3 and F4), four (F2-F5), four (F2-F5), four (F1 and

Table 3
Computational Results of benchmark functions with 50 variables.

	JADE	jDE	SaDE	EPSDE	CoDE	MPEDE	EDEV
F1	0.00E+00	0.00E+00	0.00E+00	5.25E-29	8.07E-30	0.00E+00	0.00E+00
	(0.00E+00)≈	(0.00E+00)≈	(0.00E+00)≈	(1.28E-28)−	(4.04E-29)−	(0.00E+00)≈	(0.00E+00)
F2	3.57E-27	2.22E-02	6.55E-02	5.77E-23	5.87E-09	8.45E-12	3.69E-20
	(1.67E-27)+	(3.21E-02)−	(4.34E-02)−	(8.87E-23)+	(6.51E-09)−	(5.66E-12)−	(6.42E-20)
F3	1.41E+05	4.48E+05	8.01E+05	7.82E+06	3.22E+05	6.46E+04	3.63E+04
	(6.61E+03)−	(2.21E+05)−	(5.44E+05)−	(1.35E+07)−	(8.66E+04)−	(3.15E+04)−	(2.63E+04)
F4	2.79E+00	3.88E+02	7.28E+02	3.42E+03	4.98E+02	1.28E+00	6.26E-01
	(1.26E+01)−	(3.13E+02)−	(4.56E+02)−	(3.51E+04)−	(5.78E+02)−	(5.22E-01)−	(9.12E-01)
F5	1.45E+03	3.65E+03	8.34E+03	4.66E+03	3.60E+03	4.69E+02	1.60E+03
	(4.38E+02)≈	(6.29E+02)−	(1.29E+03)−	(8.85E+02)−	(5.69E+02)−	(1.22E+02)+	(8.43E+02)
F6	7.71E+00	4.43E+02	4.39E+02	1.43E+02	1.23E+02	1.61E+00	6.36E-25
	(3.03E+01)−	(2.78E+01)−	(2.52E+01)−	(1.95E+00)−	(2.12E+00)+	(1.78E+00)−	(9.22E-25)
F7	7.57E-03	2.85E-03	9.23E-03	1.08E-02	6.54E-03	3.27E-03	7.86E-03
	(1.09E-02)≈	(6.10E-03)+	(5.45E-02)−	(1.79E-02)−	(9.36E-03)≈	(2.65E-03)+	(8.65E-03)
F8	2.11E+01	2.11E+01	2.11E+01	2.11E+01	2.01E+01	2.11E+01	2.11E+01
	(5.93E-02)≈	(3.72E-02)≈	(4.33E-02) ≈	(3.35E-02)≈	(1.09E-01)+	(2.87E-02)≈	(2.25E-02)
F9	0.00E+00	0.00E+00	9.94E-01	6.39E-16	2.38E+00	2.68E-07	0.00E+00
	(0.00E+00)≈	(0.00E+00)≈	(2.21E-01)−	(1.01E-15)−	(4.33E-01)−	(2.66E-8)−	(0.00E+00)
F10	6.55E+01	9.98E+01	1.14E+02	1.54E+02	8.29E+01	3.83E+01	4.05E+02
	(6.93E+00)−	(1.32E+01)−	(1.54E+01)−	(2.53E+01)−	(1.93E+01)−	(4.54E+00)≈	(8.33E+00)
F11	5.24E+01	5.42E+01	4.45E+01	7.04E+01	3.14E+01	4.42E+01	4.46E+02
	(2.15E+00)−	(2.10E+00)−	(1.89E+00)≈	(3.21E+00)−	(5.35E+00)+	(3.16E+00)≈	(4.65E+00)
F12	1.54E+04	1.57E+04	5.65E+04	3.16E+05	1.54E+04	8.89E+03	6.04E+03
	(1.27E+04)−	(1.54E+04)−	(2.04E+04)−	(3.85E+04)−	(1.73E+04)−	(6.24E+03)−	(5.43E+03)
F13	2.78E+00	2.94E+00	7.27E+00	6.17E+00	3.23E+00	5.68E+00	3.95E+00
	(1.99E-01)+	(2.41E-01)+	(7.34E-01)−	(6.03E-01)−	(4.15E-01)−	(9.34E-01)−	(6.43E-01)
F14	2.16E+01	2.25E+01	2.23E+01	2.34E+01	2.19E+01	2.19E+01	2.19E+01
	(4.76E-01)≈	(2.26E-01)−	(2.42E-01)−	(2.63E-01)−	(4.39E-01)−	(2.19E-01)≈	(3.67E-01)
F15	3.77E+02	3.62E+02	3.86E+02	2.64E+02	3.88E+02	3.12E+02	3.20E+02
	(9.03E+01)−	(1.21E+02)−	(7.62E+01)−	(6.45E+01)+	(6.00E+01)−	(8.51E+01)≈	(7.43E+01)
F16	8.42E+01	8.35E+01	8.78E+01	1.50E+02	9.35E+01	3.39E+01	4.98E+01
	(7.92E+01)−	(1.03E+01)−	(6.57E+01)−	(4.25E+01)−	(7.01E+01)−	(5.66E+00)+	(1.46E+01)
F17	9.39E+01	1.81E+02	9.81E+01	2.38E+02	7.21E+01	3.62E+01	9.25E+01
	(2.64E+01)≈	(2.31E+01)−	(1.01E+02)≈	(7.01E+01)−	(2.58E+01)≈	(3.11E+01)+	(6.23E+01)
F18	9.21E+02	9.20E+02	9.78E+02	8.53E+02	9.21E+02	9.17E+02	8.69E+02
	(4.38E+00)−	(3.35E+00)−	(8.37E+01)−	(2.42E+01)≈	(5.42E+00)−	(3.78E+00)−	(2.44E+01)
F19	9.19E+02	9.20E+02	9.78E+02	8.59E+02	9.21E+02	9.17E+02	8.65E+02
	(1.07E+01)−	(2.88E+00)−	(7.81E+01)−	(1.54E+01)≈	(4.64E+00)−	(3.15E+00)−	(3.12E+01)
F20	9.21E+02	9.20E+02	9.55E+02	8.56E+02	9.11E+02	9.18E+02	8.71E+02
	(3.38E+00)−	(3.14E+00)−	(3.54E+01)−	(3.03E+00)≈	(3.38E+01)−	(6.55E+00)−	(2.77E+01)
F21	5.52E+02	7.21E+02	5.66E+02	7.29E+02	6.83E+02	5.00E+02	5.00E+02
	(1.49E+02)−	(2.55E+02)−	(2.32E+02)−	(2.82E+00)−	(2.49E+02)−	(2.87E-12)≈	(3.44E-12)
F22	9.05E+02	9.05E+02	9.82E+02	5.00E+02	9.01E+02	8.98E+02	5.04E+02
	(2.48E+01)−	(1.23E+00)−	(8.21E+01)−	(6.61E-02)≈	(2.18E+01)−	(3.01E+01)−	(4.32E-02)
F23	5.82E+02	8.60E+02	5.98E+02	7.33E+02	7.10E+02	5.39E+02	5.39E+02
	(1.31E+02)−	(2.25E+02)−	(7.29E+00)−	(4.48E+00)−	(2.33E+02)−	(3.41+00)≈	(2.67E+00)
F24	2.00E+02	2.00E+02	2.89E+02	2.38E+02	2.00E+02	2.00E+02	2.00E+02
	(1.49E-12)≈	(1.62E-12)≈	(6.55E+01)−	(1.34E+01)−	(5.81E-14)≈	(1.65E-12)≈	(1.88E-12)
F25	2.18E+02	2.16E+02	2.24E+02	2.47E+02	2.17E+02	2.14E+02	2.17E+02
	(1.71E+00)≈	(1.41E+00)≈	(1.25E+01)−	(1.87E+01)−	(1.97E+00)≈	(1.08E+00)+	(2.11E+01)
−	14	18	21	18	18	11	
+	2	2	0	2	3	5	
≈	9	5	4	5	4	9	

F3-F5), five (F1-F5) and three (F2-F4) functions, respectively. In addition, it is comparable to JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on two (F1 and F5), one (F1), one (F1), zero, zero and one (F1) functions, respectively. By contrast, it is inferior to JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on one (F2), zero, zero ,one (F1), zero and one (F5) functions, respectively. Recall that EDEV is slightly worse than JADE and MPEDE when the unimodal functions are of 30 variables, therefore, the scalability of EDEV is relatively better.

Second, as for the basic multimodal functions, EDEV together with CoDE possess the highest overall performance. The reason is that the merits that EDEV inherits from CoDE enable it to get strong capability in dealing with these basic multimodal functions. More detailed comparison results are that EDEV is superior to JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on four (F6 and F10-F12), four (F6, F10-F12), five (F6, F7, F9, F10 and F12), six (F6, F7 and F9-F12), three (F9, F10 and F12) and three (F6, F9 and F12) functions, respectively. Besides, EDEV obtains similar results as JADE, jDE, SaDE, EPSDE, CoDE and

MPEDE on three (F7-F9), two (F8 and F9), two (F8 and F11), one (F8), one (F7) and three (F8, F10 and F11) functions, while it in not comparable to them on zero, one (F7), zero, zero, three (F6, F8 and F11) and one (F7) functions, respectively.

Third, EDEV demonstrates promising performance on the two expanded multimodal functions, though it is slightly less competitive than JADE. In fact, EDEV generates similar solutions as JADE for F14, while JADE performs better than EDEV on F13. EDEV outperforms jDE on F14 while inferior to it on F13. Moreover, EDEV exhibits significantly better performance than SaDE, EPSDE, CoDE and MPEDE on both F13 and F14, except that MPEDE is comparable to EDEV on F14.

Finally, it can be seen that with respect to the complex hybrid composition functions with 50 variables, the advantages of EDEV are more apparent. Actually, EDEV achieves better results than JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on eight (F15, F16 and F18-F23), nine (F15-F23), ten (F15, F16, and F18-F25), six (F16, F17, F21 and F23-F25), eight (F15, F16 and F18-F23) and four (F18-F20 and F22) functions, respectively. Moreover, the performance of EDEV is equivalent to JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on three (F17, F24 and F25), two (F24 and F25), one (F17), four (F18-F20), three (F17, F24 and F25) and four (F15, F21, F23 and F24) functions, while EDEV is inferior to them on zero, zero, zero, one (F15), zero and three (F16, F17 and F25) functions, respectively. The comparisons made above indicate that, the ensemble of multiple efficient DE variants makes EDEV a highly competitive algorithm for solving optimization problems with complex landscapes.

In conclusion, EDEV again demonstrates high efficiency in dealing with CEC 2005 benchmark functions with 50 variables and its comprehensive performance is the best among all the selected comparative algorithms. The results of Wilcoxons rank sum tests show that EDEV is significantly better than JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on 14, 18, 21, 18, 18 and 11 functions, respectively. It is significant worse than JADE, jDE, SaDE, EPSDE, CoDE and MPEDE on 2, 3, 0, 2, 3 and 5 functions while there is no significant different between EDEV and other comparative algorithms on 9, 5, 4, 5, 4 and 9 functions, respectively. Noticeably, EDEV generates the best solutions for 17 functions, including F1, F3, F4, F6, F8, F9, F10, F12, F14 and F18-F25.

4.4. Computational results of functions with 100 variables

It is very interesting to know the performance of EDEV when confronting relatively larger scale optimization problems. As the CEC2005 benchmark problems can not be scaled to be more than 50 dimensions, we test EDEV on the CEC2014 benchmark problems [86] with 100 variables. In addition, some more recently proposed DE variants are selected as comparative algorithms, including AEPD-JADE [12], rank-jDE [49], DE-VNS [87], sinDE [88] and MPEDE [24]. All the parameters of each comparative algorithm here are the same to that given in the corresponding original paper. Optimization problems in CEC2014 benchmark suit are classified into four categories including unimodal functions (F1-F3), simple multimodal functions (F4-F16), hybrid function (F17-F22) and composition function (F23-F30) The computational results are reported in 4.

It can be observed that EDEV shows competitive performance in dealing with the CEC2014 benchmark problems when they are scaled up to the dimensionality of 100. For unimodal functions, EDEV is superior to or at least comparable to all the peer DE variants. Especially, DE-VNS, sinDE and MPEDE cannot beat EDEV on any unimmodal function. In addition, AEPD-JADE shows the best performance in dealing with function F1 and rank-jDE performs best on function F3.

For the simple multimodal functions, EDEV also demonstrates promising efficiency. Generally, the best algorithms for different optimization problems are different, as different algorithms have their respective advantages. For example, EDEV is the best algorithm for functions F7, F8, F12 and F14. However, its performance for functions F6, F9, F15 and F16 is not so significant. Therefore, although EDEV is an ensemble algorithm that combines several constituent algorithms. The No Free Lunch Theorems still works on EDEV.

With regard to hybrid functions, EDEV achieves better results than AEPD-JADE, DE-VNS, rank-jDE, sinDE and MPEDE on four (F17, F18, F20 and F21), four (F17, F18, F20 and F21), three (F17, F18 and F21), four (F17, F18, F20 and F21) and two (F19 and F20) functions, respectively. In addition, EDEV is comparable to AEPD-JADE, DE-VNS, rank-jDE, sinDE and MPEDE on zero, zero, zero, zero and four (F17, F18, F21 and F22) functions, respectively. Meanwhile, EDEV fails to surpass AEPD-JADE, DE-VNS, rank-jDE, sinDE and MPEDE on two (F19 and F22), two (F19 and F22), three (F19, F20 and F22), two (F19 and F22) and zero functions, respectively.

It is impressive noting that EDEV exhibits significantly better performance when solving the highly complex composition functions. Actually, EDEV is superior to all other comparative algorithms on six optimization functions, including functions F23-F25 and functions F28-F30. It is outperformed by the other algorithms on functions F26 and F27. This phenomenon indicates that EDEV is suited to solving functions with complex landscapes, as it has diverse search manners.

In summary, this group of experiments show that EDEV has a great potential to generalize to large-scale optimization problems.

4.5. Discussion on the comparison results

Extensive experiments and comparisons demonstrate the high efficiency of EDEV. Reasons why EDEV possesses high performance can be explained as follows.

Table 4
Computational Results of CEC2014 benchmark functions with 100 variables.

	AEPD-JADE	DE-VNS	rank-jDE	sinDE	MPEDE	EDEV
F1	7.23E+03	6.19E+05	3.56E+05	1.60E+06	1.83E+05	1.02E+05
	(5.34E+03)+	(2.33E+05)−	(1.92E+05)−	(1.94E+05)−	(1.24E+05)−	(1.21E+05)
F2	4.38E+04	6.05E+03	0.00E+00	3.63E+03	0.00E+00	0.00E+00
	(2.40E+03)−	(7.44E+03)−	(0.00E+00)≈	(3.26E+03)−	(0.00E+00)≈	(0.00E+00)
F3	5.01E+03	4.38E+00	3.25E-06	4.89E+02	6.43E+02	1.80E+00
	(4.32E+03)−	(5.43E+00)−	(1.60E-05)+	(4.79E+02)−	(5.34E+02)−	(8.25E-01)
F4	1.49E+01	6.44E+01	6.99E+01	9.00E+01	4.48E+01	3.61E+01
	(2.63E+01)+	(4.56E+01)−	(3.68E+01)−	(4.44E+00)−	(1.25E+01)−	(2.24E+01)
F5	2.00E+01	2.08E+01	2.04E+01	2.07E+01	2.07E+01	2.00E+01
	(2.06E-02)≈	(4.36E-02)−	(3.11E-02)−	(3.96E-02)−	(3.57E-02)−	(3.52E-02)
F6	1.20E+01	2.18E+01	1.12E+01	3.00E-02	4.16E+01	4.61E+01
	(5.50E+00)+	(4.84E+00)+	(4.50E+00)+	(1.63E-01)+	(1.28E+00)≈	(2.21E+00)
F7	1.26E-02	1.14E-02	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	(2.69E-02)−	(1.95E-02)−	(0.00E+00)≈	(0.00E+00)≈	(0.00E+00)≈	(0.00E+00)
F8	0.00E+00	1.01E+02	0.00E+00	2.29E-01	0.00E+00	0.00E+00
	(0.00E+00)≈	(7.48E+00)−	(0.00E+00)≈	(2.48E+00)−	(0.00E+00)≈	(0.00E+00)
F9	1.03E+02	2.17E+02	8.60E+01	6.15E+01	1.07E+02	1.60E+02
	(2.19E+01)+	(7.04E+01)−	(1.21E+01)+	(9.95E+00)+	(1.01E+02)+	(8.24E+01)
F10	8.40E+00	3.98E+03	5.47E-02	3.12E+01	6.52E-01	1.46E+00
	(3.00E+01)−	(2.24E+02)−	(1.67E-01)+	(3.58E+01)−	(2.84E-01)+	(1.02E+00)
F11	3.56E+03	1.03E+04	5.20E+03	4.23E+03	1.09E+04	9.12E+03
	(4.82E+02)+	(3.56E+02)≈	(3.35E+02)+	(5.53E+02)+	(8.21E+03)≈	(8.96E+03)
F12	1.45E-01	1.43E+00	4.77E-01	5.90E-01	4.96E-01	1.64E-01
	(3.40E-02)≈	(1.50E-01)−	(5.42E-02)−	(1.42E-01)≈	(3.24E-01)−	(2.35E-01)
F13	4.08E-01	4.83E-01	3.27E-01	2.27E-01	3.04E-01	3.91E-01
	(6.92E-02)≈	(6.64E-02)−	(4.60E-02)≈	(5.53E-02)+	(6.23E-02)≈	(6.32E-02)
F14	2.29E-01	2.97E-01	4.33E-01	2.29E-01	2.12E-01	2.09E-01
	(1.87E-02)−	(3.56E-02)−	(1.73E-01)−	(3.77E-02)−	(2.54E-02)≈	(2.34E-02)
F15	1.12E+01	2.56E+01	1.14E+01	6.67E+00	1.13E+01	3.71E+01
	(1.68E+00)+	(2.52E+00)+	(1.07E+00)+	(1.35E+00)+	(3.24E+00)+	(1.25E+00)
F16	1.81E+01	2.08E+01	1.84E+01	1.83E+01	4.00E+01	4.10E+01
	(6.02E-01)+	(2.38E-01)+	(3.12E-01)+	(7.26E-01)+	(5.24E-01)≈	(8.25E-01)
F17	3.90E+04	4.93E+04	1.61E+04	2.83E+05	1.14E+04	1.06E+04
	(7.16E+04)−	(2.51E+04)−	(9.06E+03)−	(1.49E+05)−	(1.12E+04)≈	(8.95E+03)
F18	7.30E+02	9.67E+02	9.14E+02	3.55E+02	2.56E+02	2.64E+02
	(8.98E+02)−	(1.00E+03)−	(8.16E+02)−	(3.05E+02)−	(1.25E+02)≈	(1.62E+02)
F19	2.00E+01	2.47E+01	1.20E+01	9.28E+00	6.61E+01	3.39E+01
	(1.27E+01)+	(2.50E+01)+	(1.37E+00)+	(7.70E-01)+	(2.45E+01)−	(2.12E+01)
F20	5.36E+03	1.96E+02	9.42E+01	4.22E+02	4.24E+02	2.04E+02
	(6.56E+03)−	(9.39E+01)−	(4.85E+01)+	(3.98E+02)−	(1.25E+02)−	(1.35E+02)
F21	6.77E+04	1.23E+04	7.93E+03	2.29E+05	5.16E+03	5.13E+03
	(9.47E+04)−	(1.40E+04)−	(8.15E+03)−	(1.42E+05)−	(3.28E+03)≈	(2.25E+03)
F22	5.50E+02	6.26E+02	4.71E+02	3.13E+02	1.40E+03	1.60E+03
	(1.95E+02)+	(2.52E+02)+	(1.42E+02)+	(1.63E+02)+	(9.81E+02)≈	(7.41E+02)
F23	3.38E+02	3.44E+02	3.44E+02	3.44E+02	3.45E+02	3.44E+02
	(1.58E+00)≈	(3.04E-13)≈	(2.02E-13)≈	(2.60E-13)≈	(1.21E+01)≈	(1.02E+01)
F24	2.76E+02	2.86E+02	2.72E+02	2.65E+02	3.92E+02	2.03E+02
	(7.12E+00)−	(5.29E+00)−	(2.45E+00)−	(3.47E+00)−	(2.21E+00)−	(3.54E+01)
F25	2.28E+02	2.26E+02	2.08E+02	2.09E+02	2.00E+02	2.00E+02
	(4.52E+00)−	(6.26E+00)−	(3.14E+00)−	(1.14E+00)−	(5.42E-01)≈	(2.45E-01)
F26	1.07E+02	1.31E+02	1.04E+02	1.04E+02	2.01E+02	2.00E+02
	(2.53E+01)+	(4.63E+01)+	(1.82E+01)+	(1.82E+01)+	(8.42E+01)≈	(2.45E+01)
F27	7.79E+02	1.06E+03	5.15E+02	3.26E+02	9.68E+02	1.19E+03
	(1.14E+02)+	(8.73E+01)+	(8.73E+01)+	(2.19E+01)+	(8.25E+01)+	(1.02E+03)
F28	6.79E+02	1.47E+03	1.24E+03	1.10E+03	3.21E+03	6.65E+02
	(2.66E+02)−	(2.59E+02)−	(2.15E+02)−	(3.27E+01)−	(9.52E+02)−	(1.97E+02)
F29	1.10E+03	1.09E+03	9.95E+02	1.73E+03	8.79E+02	2.47E+02
	(2.90E+02)−	(2.39E+02)−	(1.55E+02)−	(3.52E+02)−	(8.53E+02)−	(1.05E+02)
F30	4.83E+03	1.07E+04	9.39E+03	8.87E+03	7.98E+03	2.64E+03
	(2.25E+03)−	(1.32E+03)−	(6.23E+02)−	(3.82E+02)−	(1.34E+02)−	(8.65E+02)
−	14	21	14	18	13	
+	11	7	11	9	4	
≈	5	2	5	3	13	

First, it is widely agreed that there will be no single algorithm being suitable to all potential optimization problems as proved by No Free Lunch Theorems [38]. Actually, if we look at the piles of publications about swarm intelligence and evolutionary algorithms, we will seldom find that one algorithm is significantly better than all other selected comparative algorithms on every optimization problems of a well-established benchmark suit. There are already plenty of sophisticated and efficient DE variants. Generally they have their own advantages on different kinds of optimization problems. In practice, it is promising to make full use of previous knowledge of DE community to create a new DE algorithm. EDEV incorporates JADE, CoDE and EPSDE. These three DE variants are very efficient and classical and each of them has special merits on some special kinds of optimization problems. For example, JADE is highly efficient in dealing with unimodal optimization problems, and CoDE shows competitive performance for some basic multi-modal optimization problems, while EPSDE exhibits extraordinary performance in solving some complex optimization problems with rotated and hybrid landscapes. EDEV takes these three algorithms as its constituent components, such that, it is able to inherit the advantages from them simultaneously. Experiments on CEC 2005 benchmark suit indicate that EDEV possesses strong capability in dealing with unimodal, multimodal and hybrid composite functions.

Second, landscapes of hard optimization problems are usually complex, especially those of multimodal ones. Different DE variants usually utilize different search strategies, and even EPSDE and CoDE employ multiple search strategies themselves. As a result, EDEV is equipped with diverse search mechanisms, which enables it to sample the solution space of an optimization problem in different manners and raise the probability to locate the optimal point. Moreover, diverse search strategies are helpful for maintaining the population diversity and delay convergence.

Third, unlike some other multi-method selection approaches which focus on computational resource allocation and choosing the most appropriate method for a problem, EDEV also brings in effective interaction and cooperation mechanisms. In EDEV, the whole population is partitioned into three indicator subpopulations and a reward subpopulation. Each constituent DE variant in EDEV owns one indicator subpopulation and the reward subpopulation will be assigned to the best performed DE variant periodically. As a result, we realize the dynamic resource allocation among constituent DE variants in EDEV. More importantly, at every generation, we mix all the updated subpopulation into a whole one and then redivide it, which means that one solution generated by one constituent DE variant can be further optimized by others. Through this way, we realize the interaction and cooperation among the three constituent DE variants. Agreed balance between exploration and exploitation is crucial to get a good stochastic population-based algorithm. The dynamic reallocation of reward subpopulation realizes the switch between exploration and exploitation. Consequently, there is no surprise that EDEV could perform better than all its constituent DE variants on some benchmark functions.

Fourth, previous studies have shown that the ensemble of multiple search strategies into one algorithm is beneficial. Here we further demonstrate that the high-level ensemble of multiple DE variants could result in a better DE algorithm. CoDE and EPSDE themselves are ensemble DE variants, which contain multiple search strategies. Therefore, the proposed multi-population framework can be a general framework to realize the hierarchical ensemble of optimization algorithms.

4.6. Dominance frequency of each constituent DE variant

It is useful to get the knowledge about how frequently each constituent DE variant dominates the evolutionary process when using EDEV to solve the optimization problems, which helps us to understand the interaction and cooperation among the component algorithms. Due to space limitation, we select nine representative benchmark functions (i.e. F2, F11, F12, F13, F15, F18, F22, F23 and F25) with 30 variables and plot in Fig. 1 the dominance frequency of each constituent algorithm during the problem solving process.

It can be observed that in major cases, JADE dominate the optimization process. This reveals that JADE is versatile in solving different kinds of optimization problems. On the contrary, CoDE usually occupies less computational resources along with the generations. The probability that EPSDE dominates the others tends to gradually increase during the run of EDEV. It worth noting that for functions F18 and F22, EPSDE dominates the optimization during the whole process instead of JADE. This is consistent to the fact that EPSDE is especially effective for functions F18-F20. Actually, the dominance frequencies of EPSDE and JADE tend to be close at the end of the optimization process for many functions, including F2, F13, F15, F18, F22, F23 and F25. In addition, the dominance frequency of CoDE for functions F11 and F13 are relatively higher than other functions, which means that CoDE exerts more impacts in solving functions F11 and F13. This is natural as we already know that CoDE is effective when solving some basic multimodal functions including F11 and F13.

The change in dominance frequency (i.e. increase and decrease) of each constituent DE variant in a certain degree reflects the interaction among the component algorithms. Take F12, F13 and F15 as examples, the dominance ratio of JADE decrease from high to medium while EPSDE increase from low to medium. This phenomenon can be explained as that JADE is good at exploitation, which quickly gets the priority at the beginning. While EPSDE gradually takes effect as the optimization proceeding to prevent the algorithm from premature, as EPSDE has stronger exploration capability. In addition, EPSDE requires longer time to identify the best mutation strategy and parameter combination.

In addition, for functions F18 and F22, we can see that the search manner in EPSDE is more suitable to the patterns of their complex landscapes. EPSDE performs the best during the evolutionary process with the most frequencies. However, the frequency of EPSDE goes down while that of JADE rises up. The reason might be that at

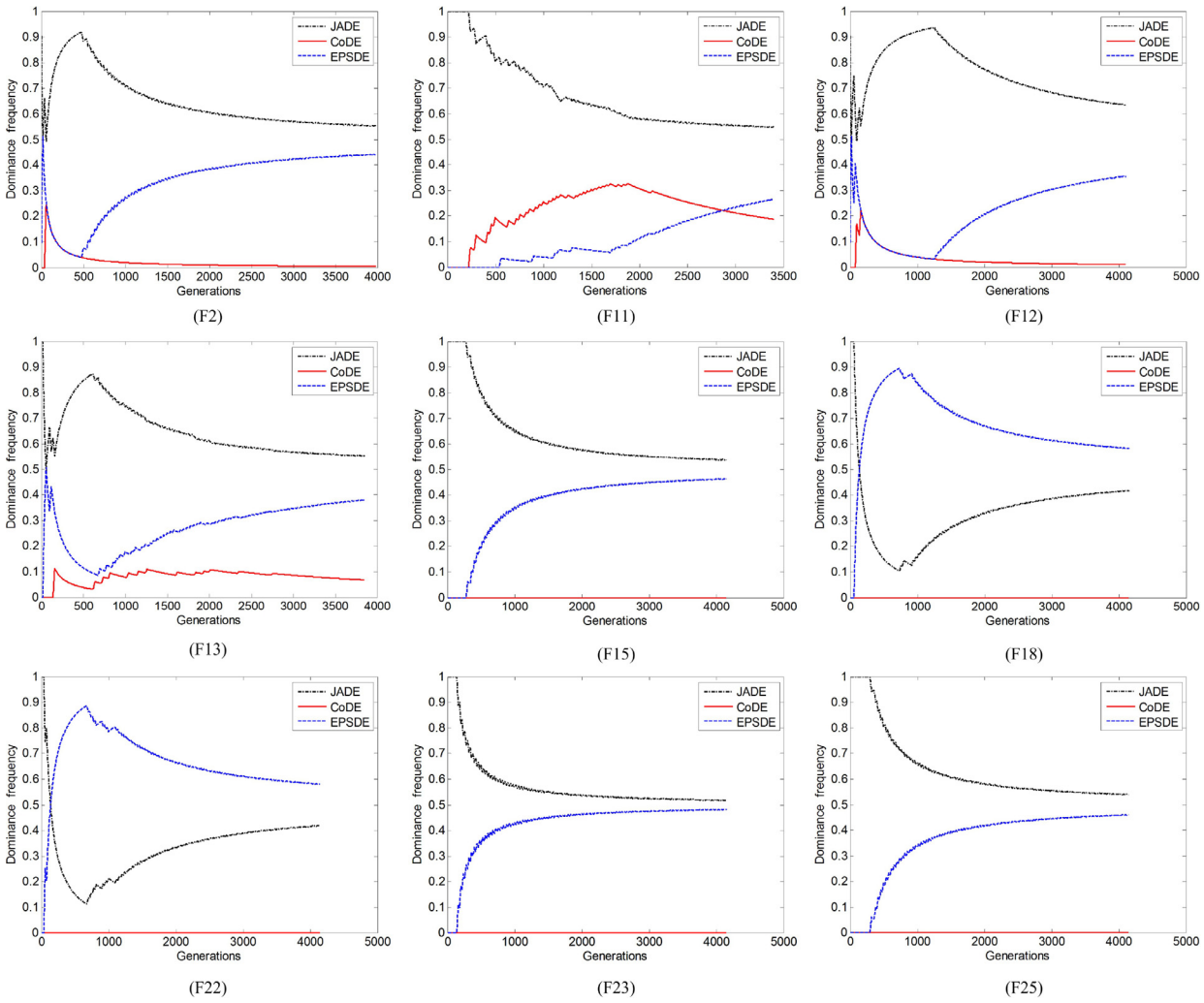

Fig. 1. Dominance frequency of each constituent DE variant

the later stage of the optimization process, JADE is given more computational resources to enhance the exploitation capability.

5. Conclusions

The popular and efficient DE variants are well designed and undoubtedly they represents the precious knowledge and experience of the DE community. It is meaningful to make full use of these knowledge to develop even more powerful and versatile DE algorithms. In this study, we propose a multi-population based framework (MPF) to realize the ensemble of multiple DE variants, i.e. JADE, EPSDE and CoDE, such that come to a new DE algorithm named EDEV. The entire population of EDEV is divided into three indicator subpopulations and one reward subpopulation. Each constituent DE variant in EDEV has an indicator subpopulation, while the reward subpopulation is dynamically allocated to the DE variant that performs the best during the recent run of EDEV. The population partition occurs at every generation while the reward subpopulation reallocation is triggered periodically. EDEV successfully inherits the advantages from its component DE variants. More importantly, the tight interaction among constituent DE variants facilitates to their cooperation between exploration and exploitation, which results a highly efficient EDEV in solving a variety of benchmark functions.

We believe that the MPF proposed in this study is a simple yet efficient framework not only for the low-level ensemble of multiple search strategies in an EA but also for the high-level ensemble of multiple EAs. In our future study, we plan to further improve MPF through designing better performance metric for determining the best constituent algorithm or search strategy during the evolutionary process. In addition, previous studies mainly focus on the ensemble of two or three homogeneous search strategies or EAs. Therefore, it could be very interesting to investigate the large-scale ensemble of heterogeneous search strategies and EAs.

Acknowledgement

This work was partly supported by the National Natural Science Foundation of China under Grant (Nos. 61603404 and 41501383), the Natural Science Foundation of Hunan Province (No. 2017JJ3369), the Research Foundation of National University of Defense Technology (ZK16-03-30), and the China Postdoctoral Science Foundation (No. 2014M562006).

References

[1] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[2] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, Evol. Comput., IEEE Trans. 15 (1) (2011) 55–66.
[3] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, Evol,. Comput., IEEE Trans. 15 (1) (2011) 4–31.
[4] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, Cybern., IEEE Trans. 44 (10) (2014) 1726–1737.
[5] 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.
[6] S.-M. Guo, C.-C. Yang, P.-H. Hsu, J.S.-H. Tsai, Improving differential evolution with a successful-parent-selecting framework, IEEE Trans. Evol. Comput. 19 (5) (2015) 717–730.
[7] L. Tang, Y. Dong, J. Liu, Differential evolution with an individual-dependent mechanism, Evol. Comput., IEEE Trans. 19 (4) (2015) 560–574.
[8] W. Gong, Z. Cai, D. Liang, Adaptive ranking mutation operator based differential evolution for constrained optimization, Cybern., IEEE Trans. 45 (4) (2015) 716–727.
[9] S.Y. Park, J.J. Lee, Stochastic opposition-based learning using a beta distribution in differential evolution, IEEE Trans. Cybern. 46 (10) (2016) 2184–2194.
[10] M.A. Ahandani, Opposition-based learning in the shuffled bidirectional differential evolution algorithm, Swarm Evol. Comput. 26 (2016) 64–85.
[11] W. Zhu, Y. Tang, J.-A. Fang, W. Zhang, Adaptive population tuning scheme for differential evolution, Inf. Sci. (Ny) 223 (2013) 164–191.
[12] M. Yang, C. Li, Z. Cai, J. Guan, Differential evolution with auto-enhanced population diversity, Cybern., IEEE Trans. 45 (2) (2015) 302–315.
[13] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, Evol. Comput., IEEE Trans. 13 (2) (2009) 398–417.
[14] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, Evol. Comput., IEEE Trans. 10 (6) (2006) 646–657.
[15] J. Zhang, A.C. Sanderson, Jade: adaptive differential evolution with optional external archive, Evol. Comput., IEEE Trans. 13 (5) (2009) 945–958.
[16] R. Sarker, S.M. Elsayed, T. Ray, et al., Differential evolution with dynamic parameters selection for optimization problems, Evol. Comput., IEEE Trans. 18 (5) (2014) 689–707.
[17] 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, Cybern., IEEE Trans. 44 (7) (2014) 1080–1099.
[18] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies, IEEE Trans. Cybern. 46 (1) (2016) 219–232.
[19] S. Sayah, A. Hamouda, A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems, Appl. Soft. Comput. 13 (4) (2013) 1608–1619.
[20] 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.
[21] A. Trivedi, D. Srinivasan, S. Biswas, T. Reindl, A genetic algorithm differential evolution based hybrid framework: case study on unit commitment scheduling problem, Inf. Sci. (Ny) 354 (2016) 275–300.
[22] Z. Xu, A. Unveren, A. Acan, Probability collectives hybridised with differential evolution for global optimisation, International Journal of Bio-Inspired Computation 8 (3) (2016) 133–153.
[23] 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 (2) (2011) 1679–1696.
[24] G. Wu, R. Mallipeddi, P. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inf. Sci. (Ny) 329 (2016) 329–345.
[25] N.M. Hamza, D.L. Essam, R.A. Sarker, Constraint consensus mutation-based differential evolution for constrained optimization, IEEE Trans. Evol. Comput. 20 (3) (2016) 447–459.
[26] X. Qiu, J.X. Xu, K.C. Tan, H.A. Abbass, Adaptive cross-generation differential evolution operators for multiobjective optimization, IEEE Trans. Evol. Comput. 20 (2) (2016) 232–244.
[27] S. Hui, P.N. Suganthan, Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization., Cybern. IEEE Trans. 46 (1) (2015) 64–74.
[28] U. Maulik, I. Saha, Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery, Pattern Recognit. 42 (9) (2009) 2135–2149.
[29] G.Y. Yang, Z.Y. Dong, K.P. Wong, A modified differential evolution algorithm with fitness sharing for power system planning, Power Syst., IEEE Trans. 23 (2) (2008) 514–522.
[30] R. Dash, P. Dash, R. Bisoi, A self adaptive differential harmony search based optimized extreme learning machine for financial time series prediction, Swarm Evol. Comput. 19 (2014) 25–42.
[31] B.E. Teoh, S. Ponnambalam, G. Kanagaraj, Differential evolution algorithm with local search for capacitated vehicle routing problem, Int. J. Bio-Inspired Comput. 7 (5) (2015) 321–342.
[32] 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.
[33] W. Gong, Z. Cai, C.X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, Syst., Man, Cybern., Part B: Cybern., IEEE Trans. 41 (2) (2011) 397–413.
[34] S.M. Elsayed, R. Sarker, D.L. Essam, et al., An improved self-adaptive differential evolution algorithm for optimization problems, Ind. Inf., IEEE Trans. 9 (1) (2013) 89–99.
[35] W. Gong, Á. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Inf. Sci. (Ny) 181 (24) (2011) 5364–5386.
[36] W. Gong, A. Zhou, Z. Cai, A multioperator search strategy based on cheap surrogate models for evolutionary optimization, Evol. Comput., IEEE Trans. 19 (5) (2015) 746–758.
[37] S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, Evol. Comput., IEEE Trans. 16 (3) (2012) 442–446.
[38] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[39] G. Wu, Across neighborhood search for numerical optimization, Inf. Sci. (Ny) 329 (2016) 597–618.
[40] X. Chen, Y.-S. Ong, M.-H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 591–607.

[41] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft. Comput. 11 (6) (2011) 4135–4151.
[42] C. Grosan, A. Abraham, Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews, in: Hybrid evolutionary algorithms, Springer, 2007, pp. 1–17.
[43] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, Cade: A hybridization of cultural algorithm and differential evolution for numerical optimization, Inf. Sci. (Ny) 378 (2017) 215–241.
[44] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art, J. Oper. Res. Soc. 64 (12) (2013) 1695–1724.
[45] P. Cowling, G. Kendall, E. Soubeiga, A Hyperheuristic Approach to Scheduling a Sales Summit, in: Practice and Theory of Automated Timetabling III, Springer, 2001, pp. 176–190.
[46] F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, Evol. Comput., IEEE Trans. 14 (5) (2010) 782–800.
[47] H.-Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, J. Global Optim. 27 (1) (2003) 105–129.
[48] A.W. Iorio, X. Li, Solving Rotated Multi-objective Optimization Problems Using Differential Evolution, in: AI 2004: Advances in artificial intelligence, Springer, 2005, pp. 861–872.
[49] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, Cybern., IEEE Trans. 43 (6) (2013) 2066–2081.
[50] X. Zhang, S.Y. Yuen, A directional mutation operator for differential evolution algorithms, Appl. Soft. Comput. 30 (2015) 529–548.
[51] S.-M. Guo, C.-C. Yang, Enhancing differential evolution utilizing eigenvector-based crossover operator, Evol. Comput., IEEE Trans. 19 (1) (2015) 31–49.
[52] J. Wang, J. Liao, Y. Zhou, Y. Cai, Differential evolution enhanced with multiobjective sorting-based mutation operators, Cybern., IEEE Trans. 44 (12) (2014) 2792–2805.
[53] Y. Cai, J. Wang, Differential evolution with hybrid linkage crossover, Inf. Sci. (Ny) 320 (2015) 244–287.
[54] F. Peñuñuri, C. Cab, O. Carvente, M.A. Zambrano-Arjona, J. Tapia, A study of the classical differential evolution control parameters, Swarm. Evol. Comput. 26 (2016) 86–96.
[55] G. Karafotias, M. Hoogendoorn, A.E. Eiben, Parameter control in evolutionary algorithms: trends and challenges, Evol. Comput., IEEE Trans. 19 (2) (2015) 167–187.
[56] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft. Comput. 9 (6) (2005) 448–462.
[57] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: Proceedings of the 9th International Conference on Soft Computing, 9, 2003, pp. 41–46.
[58] 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. 19 (2014) 52–67.
[59] A.P. Piotrowski, Review of differential evolution population size, Swarm. Evol. Comput. 32 (2017) 1–24.
[60] A.V. Kononova, D.W. Corne, P. De Wilde, V. Shneer, F. Caraffini, Structural bias in population-based algorithms, Inf. Sci. (Ny) 298 (2015) 468–490.
[61] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 71–78.
[62] R. Tanabe, A.S. Fukunaga, Improving the search performance of shade using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1658–1665.
[63] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.
[64] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106.
[65] R. Mallipeddi, P.N. Suganthan, Ensemble of constraint handling techniques, IEEE Trans. Evol. Comput. 14 (4) (2010) 561–579.
[66] R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, Ensemble strategies with adaptive evolutionary programming, Inf. Sci. (Ny) 180 (9) (2010) 1571–1581.
[67] G. Jia, Y. Wang, Z. Cai, Y. Jin, An improved (μ+ λ)-constrained differential evolution for constrained optimization, Inf. Sci. (Ny) 222 (2013) 302–322.
[68] 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.
[69] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inf. Sci. (Ny) 178 (15) (2008) 3096–3109.
[70] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Inf. Sci. (Ny) 181 (20) (2011) 4515–4538.
[71] C. Li, S. Yang, T.T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, Syst., Man, Cybern., Part B: Cybern., IEEE Trans. 42 (3) (2012) 627–646.
[72] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-s. Pan, Multi-strategy ensemble artificial bee colony algorithm, Inf. Sci. (Ny) 279 (2014) 587–603.
[73] G. Xiong, D. Shi, X. Duan, Multi-strategy ensemble biogeography-based optimization for economic dispatch problems, Appl. Energy 111 (2013) 801–811.
[74] P. Moscato, et al., On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms, Caltech concurrent computation program, C3P Report (1989) 826 (1989).
[75] N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, Evol. Comput., IEEE Trans. 9 (5) (2005) 474–488.
[76] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive memetic algorithms: a comparative study, Systems, Man, Cybern., Part B: Cybern., IEEE Trans. 36 (1) (2006) 141–152.
[77] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, A classification of Hyper-Heuristic approaches, in: Handbook of Metaheuristics, Springer, 2010, pp. 449–468.
[78] C.P. Gomes, B. Selman, Algorithm portfolios, Artif. Intell. 126 (1) (2001) 43–62.
[79] K. Tang, F. Peng, G. Chen, X. Yao, Population-based algorithm portfolios with automated constituent algorithms selection, Inf. Sci. (Ny) 279 (2014) 94–104.
[80] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Inf. Sci. (Ny) 227 (4) (2013) 60–82.
[81] F. Caraffini, F. Neri, I. Poikolainen, Micro-differential evolution with extra moves along the axes, in: 2013 IEEE Symposium on Differential Evolution (SDE), 2013, pp. 46–53.
[82] G. Iacca, F. Caraffini, F. Neri, Continuous parameter pools in ensemble differential evolution, in: Computational Intelligence, 2015 IEEE Symposium, 2015, pp. 1529–1536.
[83] J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C.C. Coello, K. Deb, Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization, J. Appl. Mech. 41 (8) (2006).
[84] 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 (6) (2006) 646–657.
[85] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Evolutionary Computation (CEC), 2013 IEEE Congress on, 2013, pp. 71–78.
[86] J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2013).
[87] D. Kovačević, N. Mladenović, B. Petrović, P. Milošević, De-vns: self-adaptive differential evolution with crossover neighborhood search for continuous global optimization, Comput. Oper. Res. 52 (2014) 157–169.
[88] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimisation, Appl. Soft. Comput. 27 (2015) 99–126.