Computing and Informatics, Vol. 39, 2020, 481–509, doi: 10.31577/cai_2020_3_481

IMPROVED MULTI-POPULATION DIFFERENTIAL EVOLUTION FOR LARGE-SCALE GLOBAL OPTIMIZATION

Yongjie Ma*, Lin Zhu, Yulong Bai

College of Physics and Electronic Engineering
Northwest Normal University, Lanzhou, Gansu, China
e-mail: myjmyj@nwnu.edu.cn

Abstract. Differential evolution (DE) is an efficient population-based search algorithm with good robustness, however, it is challenged to deal with high-dimensional problems. In this paper, we propose an improved multi-population differential evolution with best-and-current mutation strategy (mDE-bcM). The population is divided into three subpopulations based on the fitness values, each of subpopulations uses different mutation strategy. After crossover, mutation and selection, all subpopulations are updated based on the new fitness values of their individuals. An improved mutation strategy is proposed, which uses a new approach to generate base vector that is composed of the best individual and current individual. The performance of mDE-bcM is evaluated on a set of 19 large-scale continuous optimization problems, a comparative study is carried out with other state-of-the-art optimization techniques. The results show that mDE-bcM has a competitive performance compared to the contestant algorithms and better efficiency for large-scale optimization problems.

Keywords: Differential evolution, large-scale global optimization, multiple populations, best-and-current mutation strategy, random migration strategy

Mathematics Subject Classification 2010: 11Y16

* Corresponding author

1 INTRODUCTION

Large-scale global optimization (LSGO), characterized by its high-dimensional decision variables and time-consuming objective functions, is a ubiquitous and spontaneous process that frequently appears in the real world problems. It has been increasingly considered as one of the most challenging issues in engineering optimization and the most active research fields. In recent years, the corresponding technologies that calculate the global optimal solution for LSGO have got some significant progresses in mechanical design, environment engineering, chemical engineering design, bioinformatics, image processing and other fields.

Differential evolution (DE), first proposed by Storn and Price [1, 2], is one of the most competitive Evolutionary Algorithms (EAs) currently in use [3, 4], and it has been applied to solve LSGO [5, 6]. Like the standard EAs, as a population-based, heuristic and stochastic search technique, DE first generates an initial population randomly in the feasible solution space as candidate solutions, and then generates a next generation of population through crossover, mutation, selection and other evolutionary operations, which move the population toward the global optimum. Different from the standard EAs, the mutation operation of DE is realized by the linear combination of the base vector and the difference vector of some individuals. Because it does not need derivative information of solving problems [7], and it has strong robustness in complex function optimization, DE receives wide attention and application, however, with the size and complexity of LSGO growing dramatically, the performance of DE algorithm will be terribly deteriorated. In other words, due to the curse of dimensionality, DE has encountered great difficulties in dealing with high-dimensional LSGO problems.

To improve performance of EAs for LSGO, numerous methods based on DE, including DE's variants and hybridization, were proposed. To solve slow convergence and stagnation caused by high-dimensional in LSGO, DE algorithm is improved from nine mechanisms, such as strategies and their adaptations, subpopulations, etc. Due to the difference of solving problems, no algorithm has been a winner on all LSGO benchmark functions so far [6].

In this paper, we propose a multi-populations DE with best-and-current mutation strategy, called mDE-bcM, aimed at improving the search precision and solving large-scale optimization problems. The mDE-bcM divides the population into three subpopulations based on the fitness value, each subpopulation uses a given mutation strategy. We propose a new improved mutation strategy that uses a linear combination of the best vector and current vector to produce the base vector. The mutation scale factor is not a fixed value, but a random one to improve the diversity of the population. After one evolution, every subpopulation updates on the base of the new fitness value rank. And this technology mitigates the risk of lower diversity, enhances the rate of convergence, promotes the whole algorithm performance. This paper also uses two different strategies on three subpopulations, one is elitism and the other is random migration. Elitism strategy keeps one subpopulation which has good fitness values from participating in the evolution of the next generation, and

it maintains the other two subpopulations to update. Random migration strategy randomly selects ten individuals in one subpopulation which has good fitness values, five of them exchange with a random sample of five individuals in the general population, another five exchange with a random sample of five individuals in the poor population.

The mDE-bcM performance is evaluated on two sets of large-scale global optimization benchmark functions with dimensions ranging from 50 to 1 000, the results show that mDE-bcM is capable to solve continuous large-scale problems effectively and produce competitive results when compared with the state-of-the-art algorithms which are designed specifically to solve such problems.

The rest of the paper is structured as follows. Classic DE is introduced in Section 2, a brief review of related works on multi-populations and mutation strategy is presented in Section 3, the proposed algorithm (mDE-bcM) is introduced in Section 4, experimental set-up is presented in Section 5, experimental results and analysis are presented in Section 6. Finally, the work is concluded in Section 7.

2 CLASSIC DIFFERENTIAL EVOLUTION

2.1 Initialization

An initial population $X_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_{NP}\}$ in DE is randomly generated in the entire search space (unconstrained) or the feasible solution space (constrained). Based on real number coding, the initial value of the j^{th} decision variable of the i^{th} individual at generation $g = 0$ is generated among the prescribed lower bound $\mathbf{x}_{min} = \{x^j_{min} \mid j = 1, \ldots, D\}$ and upper bound $\mathbf{x}_{\max} = \{x^j_{\max} \mid j = 1, \ldots, D\}$ by:

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

where D is the dimensions of the problem, NP is population size, $\text{rand}(0,1)$ is a uniformly distributed random variable within the range $(0,1)$.

2.2 Mutation Operation

In DE, the variation vector $\mathbf{v}_{i,g} = \{v^j_{i,g} \mid j = 1, \ldots, D\}$ is composed of a base vector $\mathbf{x}_{i,r1,g} = \{x^j_{i,r1,g} \mid j = 1, \ldots, D\}$ and the differential vector $\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}$ of two random individuals $\mathbf{x}_{i,r2,g}$ and $\mathbf{x}_{i,r3,g}$ in the g^{th} generation population, the differential vector $\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}$ is scaled by the scale factor F. How to generate the variation vector is called mutation strategy, the most commonly used mutation strategies are summarized in [8]:

- DE/rand/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,r1,g} + F \cdot (\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}) \quad (2)$$

- DE/rand/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,r1,g} + F \cdot (\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}) + F \cdot (\mathbf{x}_{i,r4,g} - \mathbf{x}_{i,r5,g}) \quad (3)$$

- DE/best/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g}) \tag{4}$$

- DE/best/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g}) + F \cdot (\mathbf{x}_{i,r3,g} - \mathbf{x}_{i,r4,g}) \tag{5}$$

- DE/rand-to-best/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + K \cdot (\mathbf{x}_{\text{best},g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g}) \tag{6}$$

- DE/rand-to-best/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + K \cdot (\mathbf{x}_{\text{best},g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g} + \mathbf{x}_{i,r3,g} - \mathbf{x}_{i,r4,g}) \tag{7}$$

- DE/current-to-rand/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + K \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}). \tag{8}$$

The parameters $r1$, $r2$, $r3$, $r4$ and $r5$ are exclusive integers generated randomly within the range of $[1, NP]$, $\mathbf{x}_{\text{best},g}$ is the individual which has the best fitness value in the g^{th} generation population, $\mathbf{x}_{i,r1,g}$, $\mathbf{x}_{i,r2,g}$, $\mathbf{x}_{i,r3,g}$, $\mathbf{x}_{i,r4,g}$ and $\mathbf{x}_{i,r5,g}$ are target vectors selected from the g^{th} generation population, K is randomly selected in $[0, 1]$, F is the scale factor.

2.3 Crossover Operation

Trial vector $\mathbf{u}_{i,g} = \{u_{i,g}^j \mid j = 1, \ldots, D\}$ is generated by crossover operator. The DE generally employs two kinds of crossover methods, namely binomial crossover (bin) and exponential crossover (exp).

In binomial crossover, at least one component $u_{i,g}^j$ of trial vector $\mathbf{u}_{i,g}$ is provided randomly by the variation vector $\mathbf{v}_{i,g}$, that is as follows:

$$u_{i,g}^j = \begin{cases} v_{i,g}^j, & \text{rand}(0,1) \le CR \text{ or } j = j_{\text{rand}}, \\ x_{i,g}^j, & \text{otherwise}, \end{cases} \tag{9}$$

where $j = 1, \ldots, D$, j_{rand} is a randomly chosen integer in the range $[1, D]$, $CR \in (0, 1)$ is the crossover rate.

Exponential crossover operation is as follows:

$$u_{i,g}^j = \begin{cases} v_{i,g}^j, & j = \langle l \rangle_D, \langle l+1 \rangle_D, \ldots, \langle l+L-1 \rangle_D, \\ x_{i,g}^j, & \text{otherwise}, \end{cases} \tag{10}$$

where integer $l \in [1, D]$, integer $L \in [l, D]$ depends on the crossover probability (CR), the angular bracket $\langle l \rangle_D$ denotes a modulo function with modulus D. That

is to say, starting with l^{th} component, L components are continuously selected from the variation vector $\mathbf{v}_{i,g}$ as the component of trial vector $\mathbf{u}_{i,g}$, the remaining D-L components of trial vector $\mathbf{u}_{i,g}$ are from $\mathbf{x}_{i,g}$.

2.4 Selection Operation

The selection operation chooses the best individual according to the fitness value of target vector and trial vector in the g^{th} generation population, as shown in the following:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g}, & \text{otherwise}, \end{cases} \tag{11}$$

where $\mathbf{x}_{i,g+1}$ is the next generation target vector.

3 RELATED WORK

The DE is a simple, effective, population-based optimization technique, but it is inefficient for solving LSGO because of the computational complexity of high-dimensional LSGO [9]. Many researches have been devoted to solve these problems, and their research directions are mainly focused on multiple populations, mutation strategy, parameters control, population size adaptation, problem decomposition and so on, all of which are aimed at reducing the computational complexity in high-dimensional LSGO.

Multiple populations are one of the most widely used methods of problem decomposition, which can enhance the DE performance by parallel computing [10] and enhance the population diversity by dividing the population into independent subpopulations [11], meanwhile smaller subpopulations are efficient at quickly improving their fitness values, but smaller subpopulations might likely dissipate diversity and cause premature convergence [10].

Lampinen first tried to apply multiple populations to DE algorithm [12], who divided a population into multiple subgroups in DE calculation. Zaharie [13] proposed a DE with a multi-population approach. Weber et al. [14] proposed a parallel differential evolution, named SOUPDE, which was a multi-population algorithm with a differential evolution logic. In SOUPDE, the subpopulations quickly exploited some areas of the decision space, a new search logics, integrated shuffling and updating mechanisms were introduced into the subpopulation to avoid a diversity loss and premature convergence. Chen et al. [15] proposed a parallel differential evolution with multi-population and multi-strategy, which divided an initial population into three subpopulations with different mutation strategies, and the subpopulations evolved independently at first, and then communicated with each other at regular intervals. Ali et al. [11] proposed a multi-population DE algorithm, called mDE-bES, the population was divided into four independent subgroups, each with different mutation and update strategies, a novel mutation strategy that used information from either the best individual or a random individual was used, individuals be-

tween the subgroups were exchanged at the end of each function evaluations epoch. Wu et al. [16] proposed a multi-population ensemble DE, called MPEDE, which simultaneously consisted of three mutation strategies, i.e., "current-to-$p$best/1", "current-to-rand/1" and "rand/1". MPEDE had three equally sized smaller indicator subpopulations and one much larger reward subpopulation, each constituent mutation strategy has one indicator subpopulation. After some iterations, the best performing mutation strategy was determined according to the ratios between fitness improvements and consumed function evaluations, the reward subpopulation was allocated to the determined best performing mutation strategy dynamically.

All these studies show that population decomposition may be an effective and feasible method, which can not only reduce the complexity of calculation by reducing the population size, but also facilitate parallel calculation to improve the efficiency of calculation.

Mutation strategy also has a significant impact on the performance of DE, for each individual, the number of successful generations related to a certain mutation strategy [9]. Mutation strategy is represented by DE/x/y/z. Among them, x means the way of selecting vectors in mutation operation, its possible values are "Rand", "best", "current", "Rand to best", etc.; y represents the number of differential vectors in mutation operation, usually y = 1 or 2; z refers to crossover method, mainly including binomial (bin) and exponential (exp).

In order to improve the performance of DE algorithm, some researchers have proposed many improved mutation strategies in recent years.

Price et al. have proposed 10 mutation strategies of DE [2], including DE/rand /1/bin, DE/rand/1/exp, DE/best/1/bin, DE/best/1/exp, DE/rand/2/bin, DE /rand/2/exp, DE/best/2/bin, DE/best/2/exp, DE/rand-to-best/1/bin, DE /rand-to-best/1/exp, some of them have been described in Section 2.2.

Zhang et al. [17] proposed a new DE algorithm, called JADE, with a new mutation strategy "DE/current-to-$p$best/1" as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{12}$$

where $\mathbf{x}^p_{\text{best},g}$ is an individual randomly chosen from the top $100p\,\%$ individuals in the current population, $p \in (0, 1]$, and F_i is the mutation factor randomly generated for each $\mathbf{x}_i$. The parameter adaptation in JADE automatically updated the control parameters, the "DE/current-to-$p$best/1" used the optional archive to provide information of progress direction by historical data.

Kong et al. [18] proposed mutation operator based on symbolic function strategy:

$$\mathbf{v}_{i,g} = (1 - w)\mathbf{x}_{i,g} + w \cdot (s_1\mathbf{x}_{\text{best},g} + s_2\mathbf{x}_{r1,g}) + F \cdot (\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) \tag{13}$$

where w is a random number between 0 and 1, s_1 and s_2 are symbolic functions.

Zhang et al. [19] adopted a directional mutation operator, which could be combined with any other DE mutation strategy:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F \cdot \delta_{r2,g} \tag{14}$$

where $\delta_{r2,g}$ is a difference vector from the pool of difference vectors. If $\mathbf{u}_{i,g}$ will survive and become $\mathbf{x}_{i,g+1}$ after crossover operation, which shows that $\mathbf{u}_{i,g}$ must contain some better components than $\mathbf{x}_{i,g}$, the difference vector $\mathbf{u}_{i,g} - \mathbf{x}_{i,g}$ should be a descent direction at $\mathbf{x}_{i,g}$, and was saved into the pool of difference vectors.

Qiu et al. [20] proposed fractal mutation factor differential evolution (FMDE) algorithm, which consisted of an additional mutation factor simulated by a different Hurst index Fractal Brownian Motion (FBM), the proposed mutation strategy, improved from JADE's "DE/current-to-$p$best/1", was divided into two parts to reflect the changes of overall and random of the target population, it is defined as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + (F_i + F_{i,FBM}) \cdot (\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g} + \lambda \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})) \tag{15}$$

where F_i is variation factor of the overall changes, it changes near its position parameter μ_F and obeys cauchy distribution, $F_{i,FBM}$ is variation factor of the random changes, λ is an additional control parameter.

Raghav et al. [21] proposed a new "Memory based DE" (MBDE) where had two "swarm operators", and their operators based on the pBEST and gBEST mechanism of particle swarm optimization, their "Swarm Mutation" is as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + \left| \frac{f(p^{\text{BEST}}_{i,g})}{f(\mathbf{x}^{WORST}_{i,g})} \right| \left(p^{\text{BEST}}_{i,g} - \mathbf{x}_{i,g}\right) + \left| \frac{f(g^{\text{BEST}}_{g})}{f(\mathbf{x}^{WORST}_{i,g})} \right| \left(g^{\text{BEST}}_{g} - \mathbf{x}_{i,g}\right) \tag{16}$$

where $p^{\text{BEST}}_{i,g}$ is the personal best position of the vector $\mathbf{x}_{i,g}$, $g^{\text{BEST}}_{i,g}$ is the global best position of the vector $\mathbf{x}_{i,g}$, $f(p^{\text{BEST}}_{i,g})$, $f(g^{\text{BEST}}_{g})$ and $f(\mathbf{x}^{WORST}_{i,g})$ are the personal best function value, the global best function value and the worst function value of vector $\mathbf{x}_{i,g}$ in the current generation g, respectively.

Different mutation strategies adapt to different types of optimization problems. For example, DE/rand/1/exp is suitable for the optimization of multimodal functions, DE/best/1/exp is suitable for the optimization of unimodal functions. To further improve the performance of DE, multiple mutation strategies are used.

Elsayed et al. [22] proposed 4 new mutation strategies "DE/rand/3", "DE/best /3", "DE/rand-to-current/2" and "DE/rand-to-best and current/2" as follows:

- DE/rand/3:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F_i \cdot ((\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) + (\mathbf{x}_{r4,g} - \mathbf{x}_{r5,g}) + (\mathbf{x}_{r6,g} - \mathbf{x}_{r7,g})) \tag{17}$$

- DE/best/3:

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F_i \cdot ((\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) + (\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g}) + (\mathbf{x}_{r5,g} - \mathbf{x}_{r6,g})) \tag{18}$$

- DE/rand-to-current/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F_i \cdot ((\mathbf{x}_{r2,g} - \mathbf{x}_{i,g}) + (\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g})) \tag{19}$$

- DE/rand-to-best and current/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F_i \cdot ((\mathbf{x}_{\text{best},g} - \mathbf{x}_{r2,g}) + (\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g})). \tag{20}$$

These mutation strategies used three or two differential vectors, which get the benefit of the arithmetic recombination and the random effect to investigate the search space for both separable and non-separable unimodal test problems.

Elsayed et al. used four mutation strategies on four subpopulations, respectively.

Tong et al. [23] proposed an improved multi-population ensemble DE (IMPEDE) and used a new mutation strategy "DE/$p$bad-to-$p$best/1" instead of the mutation strategy "DE/rand/1", "DE/$p$bad-to-$p$best/1" is defined as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{pbest,g} - \mathbf{x}_{pbad,g}) \tag{21}$$

where $\mathbf{x}_{pbest,g}$, $\mathbf{x}_{pbad,g}$ are individuals randomly chosen from the best top or the bad top $100p\,\%$ individuals in the current population, respectively, $p \in (0, 1]$, the new strategy "DE/$p$bad-to-$p$best/1" utilized the information of the bad solution ($p$bad) and the good solution ($p$best) to balance exploration and exploitation.

Except "DE/$p$bad-to-$p$best/1" over subpopulation pop_1, Tong et al. used "DE /current-to-rand/1" over subpopulation pop_2 , "DE/$p$bad-to-$p$best/1" over subpopulation pop_3.

Xu et al. [24] proposed a new adaptive differential evolution named CAMDE, which defined two new multi-population-based mutation operators, denoted as "DE /current-to-nbest/u" and "DE/current-to-rand/u" as follows:

- DE/current-to-nbest/u:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{nbest,g} - \mathbf{x}_{i,g}) + F \cdot (\widetilde{\mathbf{x}}_{r1,g} - \mathbf{x}_{r2,g}) \tag{22}$$

- DE/current-to-rand/u:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{r2,g} - \mathbf{x}_{i,g}) + F \cdot (\widetilde{\mathbf{x}}_{r1,g} - \mathbf{x}_{r3,g}) \tag{23}$$

where $\mathbf{x}_{nbest,g}$ means the non-dominated best solution, and $\widetilde{\mathbf{x}}_{r1,g}$ is randomly chosen from the union $X_g \cup Y_g$ of the main population X_g and the external population Y_g, $Y_g = \{\mathbf{y}_{1,g}, \ldots, \mathbf{y}_{K,g}\}$ is composed of individuals better than the corresponding target vector $\mathbf{x}_{i,g}$, K is the maximal size of the population Y_g.

In CAMDE, each of two mutation operators is selected with equal probability.

4 IMPROVED MULTI-POPULATION DIFFERENTIAL EVOLUTION WITH BEST-AND-CURRENT MUTATION STRATEGY (MDE-BCM)

In DE algorithm, the reasonable control parameter setting and excellent mutation strategy directly affect the convergence of algorithm, the performance and reliability of the solution. Meanwhile, different strategies and parameters setting are

adapted to the different optimization problems. Optimizing different benchmarks with different characteristics such as uni-modal, multi-modal, continuous, discrete, low-dimension, high-dimension requires different mutation strategies and suitable parameters. For these reasons, the proposed algorithm in this paper divides the population into three subpopulations S1, S2 and S3 based on the fitness value, each of which follows a different mutation strategy and corresponding parameters. This technology could reduce the burden of parameter selection, increase the diversity of population, and improve the rate of convergence of the algorithm.

The structure of mDE-bcM algorithm is written in Algorithm 1.

Algorithm 1 Pseudo code of mDE-bcM

Set $CR = 0.9$, $F = 0.5$; Initialize NP, G_m, D; Set $g = 0$;
Initialize the population randomly distributed in the solution space;
for $i = 1$ to NP **do**
  Calculate $f(\mathbf{x}_{i,0})$ for each individual $\mathbf{x}_{i,0}$ in initial population X_0;
end for
Rank $\mathbf{x}_{i,0}$ based on $f(\mathbf{x}_{i,0})$;
Partition initial population X_0 into S1, S2, S3 with the same size $NP/3$;
while $(g < G_m)$ **do**
  for each $\mathbf{x}_{i,g}$ in Sk $(k = 1, 2, 3)$ **do**
    $\mathbf{v}_{i,g} = \text{mutation}(\mathbf{x}_{i,g})$;
    $\mathbf{u}_{i,g} = \text{crossover}(\mathbf{x}_{i,g}, \mathbf{v}_{i,g})$;
    $\mathbf{x}_{i,g+1} = \text{selection}(\mathbf{x}_{i,g}, \mathbf{u}_{i,g})$;
    Calculate $f(\mathbf{x}_{i,g+1})$ for each new individual $\mathbf{x}_{i,g+1}$ in subpopulation Sk;
  end for
  $X_{g+1} = \bigcup_1^k \text{S}k, (k = 1, 2, 3)$;
  Rank $\mathbf{x}_{i,g+1}$ based on $f(\mathbf{x}_{i,g+1})$;
  Partition X_{g+1} into new S1, S2, S3 with the same size;
  Update the best archive of S2 and S3 by selecting randomly 10 individuals from S1;
  Set $g = g + 1$;
end while

The mDE-bcM algorithm is described as follows.

4.1 Multiple Subpopulations

The mechanism of multi-population ensures that each subpopulation is not affected by the interference of other subpopulations in the process of evolution. Moreover, it also improves the diversity of the population to a certain degree. If the diversity of one subpopulation get worse, individuals with a large difference in the other two subpopulations do not get worse because they are evolved independently, the diversity of the population will be improved through migration among subpopula-

tions. Meanwhile, multi-population also makes it possible to parallelize which can effectively reduce the computational time.

4.2 Evolutionary Process

The mDE-bcM firstly initializes the entire population by randomly generating individuals with D dimensions, then the fitness values are calculated and sorted for all individuals. Based on the fitness values, the population is divided into three subpopulations that are S1, S2 and S3. the size of each subpopulation is $NP1 = NP/3$. S1 is a subpopulation with better fitness, S2 is a subpopulation with general fitness, S3 is a subpopulation with poor fitness. Three subpopulations evolve respectively and concurrently within each subpopulation. After crossover, mutation and selection, three subpopulations S1, S2 and S3 are combined into one population, the fitness values of individuals in the combined population are recalculated and all individuals are sorted, then combined population is divided into three subpopulations according to the fitness values. Finally, the algorithm enters the next iteration.

4.3 Best and Current Mutation Strategy

Traditional mutation strategy in dealing with high-dimensional problems can not get good convergence effect. Inspired by a greedy mutation strategy "DE/current-to-*p*best/1" in [11], we proposed a new mutation strategy called "DE/best-and-current/1".

Different from the classic mutation strategy which selects one of the population members as the base vector, and also different from mDE-bES in [11] which selects the best or a random individual as the base vector, we use a linear combination of the best vector $\mathbf{x}_{\text{best},g}$ and current vector $\mathbf{x}_{i,g}$. Mutation scale factor F also is a random value, instead of a fixed value. Mutant vector $\mathbf{v}_{i,g}$ is created as follows:

$$\mathbf{v}_{i,g} = (a_1\mathbf{x}_{\text{best},g} + a_2\mathbf{x}_{i,g}) + \text{rand}(0,1) \cdot (\mathbf{x}_{r_1,g} + \mathbf{x}_{r_2,g}) \tag{24}$$

where r_1, r_2 are random exclusive integers within the interval $[1, NP/3]$, a_1, a_2 are scalars randomly selected between $(0, 1)$, and $a_1 + a_2 = 1$.

In order to avoid the degradation of the offspring, S1 is an elitist-population, it adopts the traditional mutation strategy "DE/rand/1". S2 and S3 use a random migration strategy and user-defined best-and-current mutation strategy (as shown in formula (24), the random migration strategy randomly selects 10 individuals from S1, 5 of them are $\mathbf{x}_{\text{best},g}$ in S2 and 5 of them are $\mathbf{x}_{\text{best},g}$ in S3. Then, S1, S2 and S3 evolve in parallel using two different evolution strategies.

5 EXPERIMENTAL SET-UP

5.1 Benchmark Functions

The algorithm was tested on 19 benchmark large-scale global continuous optimization functions (F_1–F_{19}). The functions were taken from the special issue of Soft Computing on scalability of evolutionary algorithms [25, 26], the dimensions of these functions in the special issue were 50, 100, 200, 500 and 1 000, respectively, the optimal solutions for all these functions are known. Each function runs 25 times independently to evaluate the performance of the algorithm. The definitions of the functions F_1–F_{11} are shown in Table 1. In Table 2, the functions F_{12}–F_{19} are generated by hybridizing a non-separable function F_{ns} with other function F' using a splitting mechanism that defines the ratio of variables which are evaluated by F_{ns} using parameter m_{ns}.

F	Name	Definition	Range	$f(x^*)$
F_1	Shif. Sphere Function	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	-450
F_2	Shif. Schwefel Problem 2.21	$f_2(x) = \max_{i\in[1,D]} \lvert x_i\rvert$	$[-100, 100]^D$	-450
F_3	Shif. Rosenbrock's Function	$f_3(x) = \sum_{i=1}^{D}(100(x_i^2 + x_{i+1})^2 + (x_i - 1)^2)$	$[-100, 100]^D$	390
F_4	Shif. Rastrigin's Function	$f_4(x) = \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5, 5]^D$	-330
F_5	Shif. Griewank's Function	$f_5(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\frac{x_i}{\sqrt{i}} + 1$	$[-600, 600]^D$	-180
F_6	Shif. Ackley's Function	$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^D$	-140
F_7	Shif. Schwefel's Problem 2.22	$f_7(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	$[-10, 10]^D$	0
F_8	Shif. Schwefel's Problem 1.2	$f_8(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-65.536, 65.536]^D$	0
F_9	Shif. Extended f10	$f_9(x) = \left(\sum_{i=1}^{(D-1)} f_{10}(x_i, x_{i+1})\right) + f_{10}(x_D, x_1)$ $f_{10}(x) = (x^2 + y^2)^{0.25}(\sin^2(50(x^2 + y^2)^{0.1}) + 1)$	$[-100, 100]^D$	0
F_{10}	Shif. Bohachevsky	$f_{10}(x) = \sum_{i=1}^{D}[x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i)$ $-0.4\cos(4\pi x_{i+1}) + 0.7]$	$[-15, 15]^D$	0
F_{11}	Shif. Schaffer	$f_{11}(x) = \sum_{i=1}^{D-1}(x_i^2 + x_{i+1}^2)^{0.25}(\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$	$[-100, 100]^D$	0

Table 1. Properties of functions F_1–F_{11}

Function	F_{ns}	F'	m_{ns}	Range	$f(x^*)$	Function	F_{ns}	F'	m_{ns}	Range	$f(x^*)$
F_{12}	F_9	F_1	0.25	$[-100, 100]^D$	0	F_{16}	F_9	F_1	0.75	$[-100, 100]^D$	0
F_{13}	F_9	F_3	0.25	$[-100, 100]^D$	0	F_{17}	F_9	F_3	0.75	$[-100, 100]^D$	0
F_{14}	F_9	F_4	0.25	$[-5, 5]^D$	0	F_{18}	F_9	F_4	0.75	$[-5, 5]^D$	0
F_{15}	F_{10}	F_7	0.25	$[-10, 10]^D$	0	F_{19}	F_{10}	F_7	0.75	$[-10, 10]^D$	0

Table 2. Properties of functions F_{12}–F_{19} (functions F_{12}–F_{19} are hybridized by a non-separable function F_{ns} with other function F')

5.2 Parameter Settings

During experimentation, control parameters of the mDE-bcM algorithm are set as follows based on parameter tuning simulation results:

Number of subpopulations is set to 3.

The population size ($NP = 60$) is maintained constant during the evolution process.

DE crossover operator: binomial.

Crossover rate: $CR = 0.9$.

The mDE-bcM and other DE variants were coded in Matlab environment.

The computations were carried out using a PC with Intel(R) Core(TM) i3-2350M @2.3 GHz CPU and 2 GB RAM while running Matlab R2012a on 64-bit Windows operating system.

6 NUMERICAL RESULTS AND DISCUSSIONS

6.1 Simulation Results

The error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) for all test functions, including the best, median, mean, worst values and standard deviation, are reported in Tables 3 and 4, where $f(\mathbf{x})$ is the global optimum we found, $f(\mathbf{x}^*)$ is the global optimum in Tables 3 and 4, and dimensions $D = 50$, 100, 200, 500 and 1 000. The error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) was adopted as a performance metric of algorithms. The number of function evaluations (FEs) for each category of dimensions for these problems is set as 30 000.

D	Values	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
50	Best	4.62E−97	4.04E−44	4.90E+01	0.00E+00	0.00E+00	8.88E−16	1.32E−49	3.94E−81	1.76E−23	1.76E−23
	Median	1.55E−94	5.83E−43	4.90E+01	0.00E+00	0.00E+00	8.88E−16	1.07E−47	2.04E−79	1.24E−22	1.24E−22
	Mean	2.05E−92	8.26E−43	4.90E+01	0.00E+00	0.00E+00	8.88E−16	4.44E−47	8.30E−79	1.87E−22	1.87E−22
	Worst	1.59E−91	3.18E−42	4.90E+01	0.00E+00	0.00E+00	8.88E−16	2.02E−46	6.89E−78	5.58E−22	5.58E−22
	Std	2.71E−93	1.16E−43	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.25E−47	2.92E−79	3.83E−23	3.83E−23
100	Best	7.18E−195	2.13E−86	9.90E+01	0.00E+00	0.00E+00	8.88E−16	2.00E−97	1.06E−159	4.88E−47	0.00E+00
	Median	1.68E−190	1.99E−85	9.90E+01	0.00E+00	0.00E+00	8.88E−16	9.96E−96	1.82E−157	3.34E−46	0.00E+00
	Mean	1.42E−188	3.07E−85	9.90E+01	0.00E+00	0.00E+00	8.88E−16	1.07E−94	6.65E−157	6.52E−46	0.00E+00
	Worst	9.55E−188	1.44E−84	9.90E+01	0.00E+00	0.00E+00	8.88E−16	5.30E−94	8.03E−156	2.07E−45	0.00E+00
	Std	0.00E+00	1.10E−85	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.04E−95	9.30E−158	1.46E−46	0.00E+00
200	Best	0.00E+00	3.41E−163	1.99E+02	0.00E+00	0.00E+00	8.88E−16	1.52E−161	0.00E+00	0.00E+00	0.00E+00
	Median	0.00E+00	1.05E−162	1.99E+02	0.00E+00	0.00E+00	8.88E−16	4.69E−161	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	1.03E−162	1.99E+02	0.00E+00	0.00E+00	8.88E−16	4.82E−161	0.00E+00	0.00E+00	0.00E+00
	Worst	0.00E+00	1.55E−162	1.99E+02	0.00E+00	0.00E+00	8.88E−16	9.20E−161	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.64E−162	0.00E+00	0.00E+00	0.00E+00
500	Best	0.00E+00	3.62E−163	4.99E+02	0.00E+00	0.00E+00	8.88E−16	1.51E−161	0.00E+00	0.00E+00	0.00E+00
	Median	0.00E+00	1.07E−162	4.99E+02	0.00E+00	0.00E+00	8.88E−16	5.42E−161	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	1.05E−162	4.99E+02	0.00E+00	0.00E+00	8.88E−16	5.79E−161	0.00E+00	0.00E+00	0.00E+00
	Worst	0.00E+00	1.55E−162	4.99E+02	0.00E+00	0.00E+00	8.88E−16	1.17E−160	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.68E−162	0.00E+00	0.00E+00	0.00E+00
1 000	Best	0.00E+00	3.87E−163	9.99E+02	0.00E+00	0.00E+00	8.88E−16	1.71E−161	0.00E+00	0.00E+00	0.00E+00
	Median	0.00E+00	1.10E−162	9.99E+02	0.00E+00	0.00E+00	8.88E−16	5.99E−161	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	1.08E−162	9.99E+02	0.00E+00	0.00E+00	8.88E−16	6.46E−161	0.00E+00	0.00E+00	0.00E+00
	Worst	0.00E+00	1.55E−162	9.99E+02	0.00E+00	0.00E+00	8.88E−16	1.36E−160	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.14E−162	0.00E+00	0.00E+00	0.00E+00

Table 3. Experimental results obtained by mDE-bcM on functions F_1–F_{10}

D	Values	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
50	Best	7.47E−24	7.80E−23	3.70E+01	3.61E−49	2.16E−23	1.20E+01	3.48E−24	1.17E−45	3.61E−49
	Median	6.55E−23	4.59E−22	3.70E+01	2.66E−47	1.18E−22	1.20E+01	3.19E−23	1.24E−44	2.66E−47
	Mean	1.07E−22	6.56E−22	3.70E+01	1.48E−46	1.69E−22	1.20E+01	4.27E−23	2.48E−44	1.48E−46
	Worst	3.24E−22	1.90E−21	3.70E+01	7.37E−46	4.76E−22	1.20E+01	1.20E−22	1.04E−43	7.37E−46
	Std	2.27E−23	1.19E−22	0.00E+00	5.16E−47	3.23E−23	0.00E+00	8.31E−24	2.36E−45	5.16E−47
100	Best	4.02E−47	7.06E−45	8.70E+01	4.35E−97	8.64E−46	1.20E+01	6.39E−48	3.09E−90	4.35E−97
	Median	4.54E−46	6.65E−44	8.70E+01	2.83E−95	8.34E−45	1.20E+01	7.86E−47	9.52E−89	2.83E−95
	Mean	1.33E−45	1.12E−43	8.70E+01	5.84E−94	1.17E−44	1.20E+01	1.59E−46	3.56E−87	5.84E−94
	Worst	4.74E−45	3.91E−43	8.70E+01	3.19E−93	3.62E−44	1.20E+01	5.11E−46	2.99E−86	3.19E−93
	Std	3.04E−46	2.67E−44	0.00E+00	1.56E−94	2.13E−45	0.00E+00	3.33E−47	3.28E−87	1.56E−94
200	Best	0.00E+00	0.00E+00	1.87E+02	1.28E−161	0.00E+00	1.20E+01	0.00E+00	9.04E−168	1.28E−161
	Median	0.00E+00	0.00E+00	1.87E+02	4.21E−161	0.00E+00	1.20E+01	0.00E+00	1.34E−166	4.21E−161
	Mean	0.00E+00	0.00E+00	1.87E+02	4.39E−161	0.00E+00	1.20E+01	0.00E+00	3.51E−166	4.39E−161
	Worst	0.00E+00	0.00E+00	1.87E+02	8.68E−161	0.00E+00	1.20E+01	0.00E+00	2.05E−165	8.68E−161
	Std	0.00E+00	0.00E+00	0.00E+00	5.56E−162	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.56E−162
500	Best	0.00E+00	0.00E+00	4.87E+02	1.52E−161	0.00E+00	1.20E+01	0.00E+00	1.76E−171	1.52E−161
	Median	0.00E+00	0.00E+00	4.87E+02	5.45E−161	0.00E+00	1.20E+01	0.00E+00	4.05E−170	5.45E−161
	Mean	0.00E+00	0.00E+00	4.87E+02	5.77E−161	0.00E+00	1.20E+01	0.00E+00	7.99E−170	5.77E−161
	Worst	0.00E+00	0.00E+00	4.87E+02	1.17E−160	0.00E+00	1.20E+01	0.00E+00	4.55E−169	1.17E−160
	Std	0.00E+00	0.00E+00	0.00E+00	7.65E−162	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.65E−162
1 000	Best	0.00E+00	0.00E+00	9.87E+02	1.76E−161	0.00E+00	1.20E+01	0.00E+00	4.66E−173	1.76E−161
	Median	0.00E+00	0.00E+00	9.87E+02	5.85E−161	0.00E+00	1.20E+01	0.00E+00	9.61E−172	5.85E−161
	Mean	0.00E+00	0.00E+00	9.87E+02	6.35E−161	0.00E+00	1.20E+01	0.00E+00	6.07E−171	6.35E−161
	Worst	0.00E+00	0.00E+00	9.87E+02	1.36E−160	0.00E+00	1.20E+01	0.00E+00	4.19E−170	1.36E−160
	Std	0.00E+00	0.00E+00	0.00E+00	9.33E−162	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.33E−162

Table 4. Experimental results obtained by mDE-bcM on functions F_{11}–F_{19}

Tables 3 and 4 show that there are 11 functions where median values are equal to 0, then 4 functions whose median values are less than 10E−160, and 4 functions where median values are worse than 10E−160 in 19 functions, when $D = 1\,000$. But when $D = 50$, there are 2 functions whose median values are equal to 0, and 17 functions whose median values are worse than 10E−160, what shows that the algorithm mDE-bcM performs well in solving high-dimensional problems.

Tables 3 and 4 also show that there are 10 functions whose median values are equal to 0, 5 functions whose median values are less than 10E−160, and 4 functions whose median values are worse than 10E−160 in 19 functions, when $D > 200$, which shows that the algorithm mDE-bcM performs well when the dimension D exceeds 200.

But Tables 3 and 4 also reveal that the mDE-bcM appears to have difficulties on function F_3 (Shifted Rosenbrock's function), F_5 (Shifted Griewank's function), F_6 (Shifted Ackley's function), F_{13} (hybrid composition function) and F_{16} (hybrid composition function), regardless of the dimension. Functions F_3 and F_6 are multimodal functions, functions F_{13} and F_{16} are hybrid composition functions and hybridized by a non-separable function F_9 with other function F_3 or F_1. This shows that the algorithm mDE-bcM performs poorly on some multimodal functions, although it performs well on multimodal function F_4. If it is used in multimodal function optimization, premature convergence must be avoided.

In Table 3 and 4, the mean error is better than its corresponding median, which means that individuals are evenly distributed in the population, without much worse values.

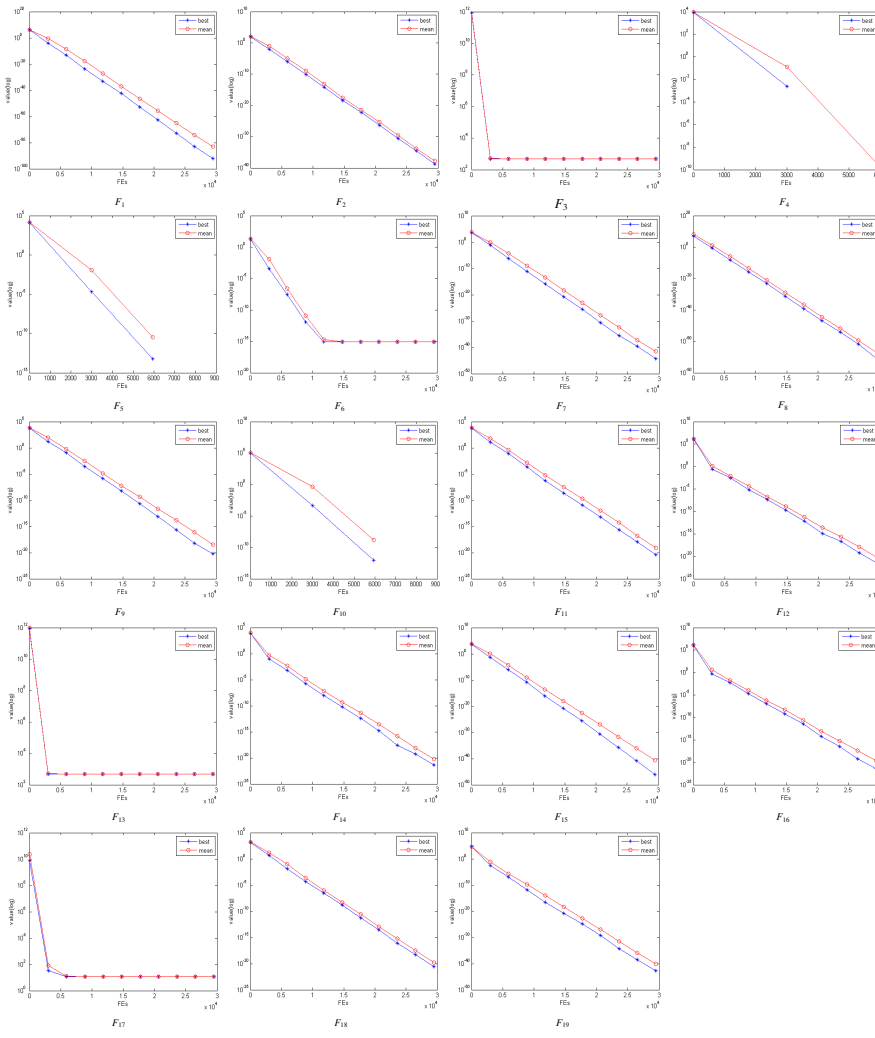


Figure 1. The best value and the mean value, where dimension $D = 500$, the test functions are from F_1 to F_{19}

The best values and the mean values of 19 test functions are shown in Figure 1, where dimension $D = 500$, number of iteration is 30 000. The horizontal axis is the number of iterations (FEs), and the vertical axis is the error values of fitness (log).

From Figure 1, the continuous optimization process of the mDE-bcM on functions F_1–F_{19} can be observed, showing that the mDE-bcM has excellent performance on most functions. But the best error and mean error were rapidly reduced in the

early stages of iteration and slowly changed in the later stages of iteration on F_3, F_6, F_{13} and F_{17}, uniformly reduced on other functions. By analyzing individuals in population, we observed that rapid convergence occurred on F_3, F_6, F_{13} and F_{17}, approximate optimal solutions are found after about 300 generations, and after that, improvements are very slow, and more generations only consume the processing time, what shows that the mDE-bcM has general performance on these functions. In fact, a single mutation strategy always performs poorly.

In addition, the curve is interrupted on F_4, F_5 and F_{10}, because the error values $(f(\mathbf{x}) - f(\mathbf{x}^*))$ are less than 0 and the vertical axis is logarithmic.

Overall, the mDE-bcM was able to optimize on 2 functions at $D = 50$, 3 functions at $D = 100$, 11 functions at $D = 200$, 11 functions at $D = 500$, and 11 functions at $D = 1\,000$. It was also able to obtain high quality results for the rest of the functions at different dimensions with small errors.

In summary, the mDE-bcM has excellent performance on most functions, especially in high dimension, although it encounters difficulties in a few functions.

6.2 Parameter Tuning

In this subsection, we will evaluate the sensitivity of the proposed algorithm to some parameters. The decision variables to be adjusted in mDE-bcM include the population size (NP) and mutation strategies.

In these tests, we varied NP at a time while keeping the other parameters fixed, or used different strategies in the evolutionary process. We performed 25 independent runs for every set of parameters. In order to test the effect of the population size (NP) and mutation strategies more clearly, the population was not grouped, only one population was used.

Table 5 shows results of parameter tuning while population size NP changes from 30 to 300, where $D = 500$. Figure 2 shows comparison between our best-and-current mutation strategy and 7 classic mutation strategies, where $NP = 60$, $D = 500$.

NP	30	60	90	180	300
F_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.28E−189
F_3	4.99E+02	4.99E+02	4.99E+02	4.99E+02	4.99E+02
F_4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_7	1.41E−161	1.51E−161	1.73E−161	3.95E−158	4.87E−95
F_8	0.00E+00	0.00E+00	0.00E+00	1.27E−258	3.34E−155
F_{15}	1.08E−161	1.52E−161	1.60E−161	4.27E−159	1.01E−94
F_{19}	2.45E−169	1.76E−171	6.06E−172	5.02E−154	7.32E−94

Table 5. Population size (NP) tuning. The data in Table 5 are the average of 25 calculations, where $D = 500$.

Table 5 exhibits different errors during the fitness evaluations while the mDE-bcM selects different population size NP, it is obvious that better results can be

obtained where $NP < 90$, and optimal results can be obtained on most functions while $NP = 60$. Bigger population size makes it easier for individuals in the population to carry more abundant genes, maintain the diversity of the population and quickly obtain high-quality solutions, but it also means that more extra fitness evaluations are done, what consumes more computing time. A smaller population size means shorter computing time, but it leads to poor population diversity and calculation success rate. Although the mDE-bcM can also obtain better results on some functions while $NP = 30$, as a compromise, we select $NP = 60$ in this paper, and sub-population size is 20 after grouping.

The mean values of our best-and-current mutation strategy and other 7 mutation strategies on functions F_1, F_2, F_6, F_7, F_8 and F_9 are shown in Figure 2.

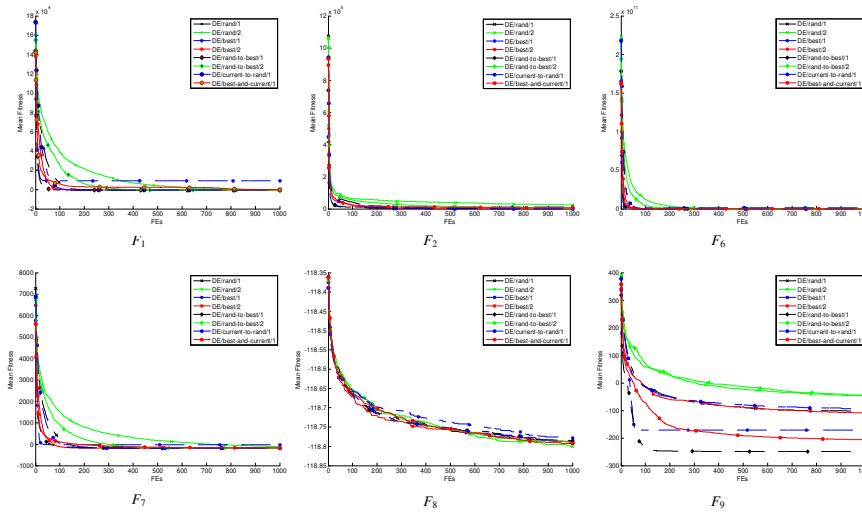


Figure 2. Comparisons between our best-and-current mutation strategy and 7 mutation strategies on functions F_1, F_2, F_6, F_7, F_8 and F_9

Figure 2 illustrates that our best-and-current mutation strategy performs better than most traditional mutation strategies, because its base vector is linear combination of the best vector $\mathbf{x}_{\text{best},g}$ and current vector $\mathbf{x}_{i,g}$, instead of the best vector $\mathbf{x}_{\text{best},g}$ or the random/current vector $\mathbf{x}_{i,g}$, so it inherits two excellent genes of the parent as base vectors at the same time. Like other methods, best-and-current mutation strategy also converges rapidly on functions F_1, F_2, F_6 and F_7, that means the mDE-bcM and other mutation strategies fall into prematurity and individuals become the same, test results on 19 functions also show that a single mutation strategy will always perform poorly on some functions, so it is necessary to adopt multiple mutation strategies or combinations of them.

6.3 Computational Complexity Analysis

The mDE-bcM divides initial population X_0 into three subpopulations S1, S2 and S3, subpopulation size is reduced to $NP/3$, then mDE-bcM carries out the evolutionary operations such as crossover, mutation and selection in parallel, but it is still a serial algorithm structure between two runs, so computational complexity is determined by the number of calls to genetic operators, as shown in the following [27, 28]:

$$O(D \cdot NP/3 \cdot G_m). \tag{25}$$

In mDE-bcM, the outer loop controls the number of iterations and its maximum value is G_m, and the inner loop controls the number of individuals involved in evolution and its maximum value is $NP/3$, each individual contains D components. In formula (25), D and G_m are necessary for high dimension and high precision, so reducing the number of individuals from NP to $NP/3$ can effectively reduce the computational complexity, which is the value of multiple populations with parallel computing.

6.4 Comparison with Other State-of-the-Art Optimization Techniques

In this section, we compared mDE-bcM with two groups of state-of-the-art optimization algorithms, which is because these two groups of algorithms were tested on SOCO 2011 [25, 26]. Like the analysis in literature [29], different algorithms have different performance in different test function suites, no algorithm is always excellent in all test function suites. For example, MOS-CEC2013 [30] performs best on the CEC 2010 benchmark suite [31] and CEC 2013 benchmark suite [32], but ranks 6$^{\text{th}}$ on the SOCO 2011 benchmark suite, MOS-SOCO2011 [33] performs best on the CEC 2010 benchmark suite, but ranks 4$^{\text{th}}$ on the SOCO 2011 benchmark suite and 8$^{\text{th}}$ CEC 2013 benchmark suite. On the other hand, the algorithms from the first group are all tested in different dimensions, which can reflect the ability of the algorithm to deal with large-scale optimization problems in different dimensions, whereas the algorithms from the second group are all tested in the dimension of 1 000, which only reflects the high-dimensional processing ability of the algorithm.

For this reason, both groups of algorithms were tested by researchers on 19 benchmark large-scale global continuous optimization functions, these functions were taken from the special issue of Soft Computing on scalability of evolutionary algorithms [25, 26], the algorithms from the first group were compared in [11], whereas the algorithms from the second group were compared in [34].

We have also used the Friedman test for multiple comparisons to check differences among the considered algorithms, the statistical methods include:

1. Overall ranking according to the Friedman test: we firstly rank each algorithm according to its mean value for each function, then compute its average ranking on all the functions.

2. # Best: This is the number of functions for which each algorithm obtains the best results compared to all the other algorithms.
3. nWins: This is the number of our mDE-bcM is better than, similar to and worse than that of the corresponding algorithm according to the Wilcoxon Signed Rank Test in a pair-wise comparison [35].
4. The Friedman test: The Friedman test can detect whether there are significant differences between the behavior of multiple algorithms.

The comparison algorithms in the first group are as follows:

1. The classic DE algorithm [1]. The strategy is "DE/rand/1/exp", $CR = 0.9$, $F = 0.5$.
2. Real-coded Genetic Algorithm (CHC) [36]. It is a real-coded genetic algorithm using interval-schemata as an analysis tool and was tested by 13 test functions.
3. CMA-ES [37]. It introduced a restart-CMA evolution strategy and was evaluated on 25 test functions of the CEC 2005 whose dimension $D = 10$, 30 and 50. In CMA-ES, the default population size prescribed for the (μ_W, λ)-CMA-ES grew with $logD$ and equaled to $\lambda = 10$, 14, 15 for $D = 10$, 30, 50, respectively. On multi-modal functions, the optimal population size λ could be considerably greater than the default population size.
4. MA-SSW [38]. Molina et al. proposed a memetic algorithm based on local search chains for high dimensionality, MA-SSW-Chains used the Subgrouping Solis Wets' algorithm as its local search method, in which only a random subset of the variables was explored and this subset would change after a certain number of evaluations. MA-SSW-Chains is considered to be one of the best algorithms in CEC 2010 benchmark suite.
5. MTS-LS1 [39]. It introduced multiple agents to search the solution space and was evaluated on 7 test functions of the CEC 2008 special session and competition on large scale global optimization. Each agent in MTS-LS1 did an iterated local search using one of three candidate local search methods and might find its way to a local optimum or the global optimum. MOS-SOCO2011 based on MTS-LS1 is considered to be one of the best algorithms in SOCO 2011 benchmark suite.
6. SaDE [7]. In SaDE, the trial vector generation strategies with their associated parameters values were gradually self-adapted by learning from previous successful experiences. SaDE was evaluated on 26 test functions, two of which were chosen from [40].
7. EvoPR [41]. Rahnamayan et al. proposed the evolutionary path relinking (EvoPR) to finding a global optimum of multi-modal functions and unconstrained large-scale problems, EvoPR was tested on 19 test functions for the special issue of Soft Computing on scalability of evolutionary algorithms and

other metaheuristics for large scale continuous optimization problems [25] with dimensions ranging from 50 to 1 000.

8. mDE-bES [11]. The mDE-bES utilized exponential crossover, and divided the population into four subgroups, each of which employed a certain modified mutation strategy, $CR \in [0.2, 0.9]$ and $F \in [0.5, 0.9]$.

The algorithms from the first group were initially tested on different functions, but were finally tested on the same initial population, same benchmark suite $F1$–$F19$ and same stopping rule in [11], and dimension changed from 50 to 1 000, but G-CMA-ES is not evaluated at dimension 1 000.

The comparison algorithms in the second group are from literature [29, 34]. [29] provided a comprehensive comparison of the performance of several algorithms evaluated on the CEC 2010, CEC 2013 and SOCO 2011 benchmark suites, [34] compared its algorithm CCJADE with six state-of-the-art algorithms, according to their analysis, the algorithms with excellent performance in recent years are as follows:

1. MOS-CEC2013 [30]. MOS-CEC2013 is a hybrid algorithm that combines a population-based search algorithm, a Genetic Algorithm (GA), with two powerful local searches (the Solis Wets' algorithm and a variation of the MTS-LS1 local search). It was the champion of the competition on LSGO used the CEC 2013 LSGO benchmark suite in 2013 which defined 15 test functions with dimension 1 000 or 905 in [32].
2. MOS-SOCO2011 [29, 33]. MOS-SOCO2011 combined a Differential Evolution (DE) algorithm and the first of the local searches of the MTS algorithm (MTS-LS1), and used the Multiple Offspring (MOS) framework which made the seamless combination of multiple search algorithm in a dynamic way. It obtained the best overall results among all the algorithms included in the 2011 special issue of the Soft Computing journal.
3. jDElscop [42]. The jDElscop was a self-adaptive DE algorithm that used three different DE strategies (DE/rand/1/bin, DE/rand/1/exp and DE/best/1/bin), a sign-change control mechanism for the F parameter and a new population size reduction mechanism. It was the runner-up in the 2011 special issue of the Soft Computing journal.
4. GaDE [43]. GaDE was a generalization of the adaptive DE algorithm, which used a probability distribution to adapt the value of each of the parameters of the algorithm for each of the individuals of the population. It obtained the third place among all the algorithms included in 2011 special issue of the Soft Computing journal.
5. DECC-G [44]. The DECC-G was a cooperative coevolution algorithm, which divided large problems into small components that were optimized independently by certain EAs. It won the second place in the competition on LSGO in 2013 and the fourth place in 2015.

6. CCJADE [34]. The CCJADE used a surrogate-assisted CC (SACC) optimizer, in which fitness surrogates are exploited within the low-dimensional subcomponents resulting from the problem decomposition, and it was tested on the SOCO 2011 benchmark suite where $D = 1\,000$.

7. L-SHADE [45]. The L-SHADE further extended SHADE with Linear Population Size Reduction (LPSR), which continually decreased the population size according to a linear function, and L-SHADE was evaluated on CEC2014 benchmarks.

8. LM-CMA-ES [46]. The LM-CMA-ES was a computationally efficient limited memory Covariance Matrix Adaptation Evolution Strategy for large scale optimization, which sampled candidate solutions according to a covariance matrix reproduced from m direction vectors selected during the optimization process, the decomposition of the covariance matrix into Cholesky factors could reduce the time and memory complexity of the sampling. LM-CMA-ES could efficiently solve fully non-separable problems and reduce the overall run-time.

The algorithms from the second group were initially tested on different functions, but were finally tested on the same functions at dimension 1 000 in [34].

Because of the difference of the running environment, such as hardware platform, programming language and program efficiency, etc., the performance of different algorithms is very different, and the test results in different environments are actually not comparable. Therefore, the test results of the algorithms in the first group are coming from [11], the test results of the algorithms in the second group are coming from [34], because they give the best results that these algorithms can get.

The comparison results of the mDE-bcM and the first group of algorithms is shown in Tables 6, 7, 8, 9 and 10, we run for 25 times on every function $F_1 - F_{19}$, where D changes from 50 to 1 000. Results highlighted in boldface show the best mean values for each function. As suggested in the special issue [25], all values below 1.00E−14 are approximated to 0.00E+00.

Tables 6, 7, 8, 9 and 10 show that the proposed mDE-bcM has superior performance compared with the first group of algorithms on functions $F_1 - F_{19}$, while D changes from 50 to 1 000. As indicated in Tables 6, 7 and 8, the mDE-bcM can obtain optima except functions F_3, F_{13} and F_{17}, where D changes from 50 to 200. In Tables 9 and 10, the mDE-bcM can obtain optima except functions F_3 and F_{13}, where D changes from 500 to 1 000. The number of times the mDE-bcM can obtain the optimal solution is higher than that of other algorithms.

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	**0.00E+00**	1.67E−11	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.68E+01	1.22E−02	**0.00E+00**	**0.00E+00**
F_2	8.84E−11	6.19E+01	2.75E−11	7.61E−02	8.84E−14	1.21E+02	3.71E−01	1.52E+01	**0.00E+00**
F_3	1.63E+02	1.25E+06	7.97E−01	4.79E+01	1.63E+02	7.46E+04	1.12E+02	**4.76E−05**	4.90E+01
F_4	**0.00E+00**	7.43E+01	1.05E+02	r1.19E−01	**0.00E+00**	1.07E+01	4.96E−02	1.77E+01	**0.00E+00**
F_5	7.68E−03	1.67E−03	2.96E−04	**0.00E+00**	7.68E−03	1.87E−01	5.13E−02	**0.00E+00**	**0.00E+00**
F_6	**0.00E+00**	6.15E−07	2.09E+01	4.89E−14	**0.00E+00**	4.63E−02	6.85E−03	3.97E−14	**0.00E+00**
F_7	**0.00E+00**	2.66E−09	1.01E−10	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.63E−02	**0.00E+00**	**0.00E+00**
F_8	9.56E−12	2.24E+02	**0.00E+00**	3.06E−01	9.65E−12	6.92E+05	2.08E+02	1.64E−09	**0.00E+00**
F_9	1.03E+02	3.10E+02	1.66E+01	2.94E+02	1.03E+02	3.00E−02	8.02E+00	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	7.30E+00	6.81E+00	**0.00E+00**	**0.00E+00**	2.94E−02	4.80E−02	**0.00E+00**	**0.00E+00**
F_{11}	1.04E+02	2.16E+00	3.01E+01	r4.49E−03	1.04E+02	8.35E−02	9.68E+00	1.15E−08	**0.00E+00**
F_{12}	1.34E+01	9.57E−01	1.88E+02	**0.00E+00**	1.34E+01	4.80E+01	r2.27E+00	**0.00E+00**	**0.00E+00**
F_{13}	2.94E+01	2.08E+06	1.97E+02	3.02E+01	2.94E+01	3.42E+09	4.22E+01	**2.50E−01**	3.70E+01
F_{14}	5.52E+01	6.17E+01	1.09E+02	**0.00E+00**	5.52E+01	4.22E+03	9.97E−01	9.60E+00	**0.00E+00**
F_{15}	**0.00E+00**	3.98E−01	9.79E−04	**0.00E+00**	**0.00E+00**	8.50E−03	6.38E−02	**0.00E+00**	**0.00E+00**
F_{16}	4.06E+01	2.95E−09	4.27E+02	4.06E−03	4.06E+01	1.36E+01	5.63E+00	**0.00E+00**	**0.00E+00**
F_{17}	2.17E+02	2.26E+04	6.89E+02	2.60E+01	2.17E+02	2.36E+05	6.77E+01	2.42E−01	1.20E+01
F_{18}	5.65E+01	1.58E+01	1.31E+02	**0.00E+00**	5.65E+01	2.72E+01	1.62E+00	5.65E−05	**0.00E+00**
F_{19}	**0.00E+00**	3.59E+02	4.76E+00	**0.00E+00**	**0.00E+00**	1.15E−01	5.03E−02	**0.00E+00**	**0.00E+00**

Table 6. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 50$

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	3.79E+00	3.56E−11	**0.00E+00**	**0.00E+00**	1.09E−12	3.13E+01	4.34E−02	**0.00E+00**	**0.00E+00**
F_2	7.58E+01	8.58E+01	1.51E−10	7.01E+00	4.66E−10	1.26E+02	3.30E+00	4.00E+01	**0.00E+00**
F_3	1.27E+02	4.19E+06	3.88E+00	1.38E+02	2.32E+02	1.11E+05	3.98E+02	**4.90E−01**	9.90E+01
F_4	2.85E+00	2.19E+02	2.50E+02	1.19E−01	1.05E−12	1.58E+01	1.07E−01	1.87E+01	**0.00E+00**
F_5	3.05E−01	3.83E−03	1.58E−03	**0.00E+00**	6.70E−03	3.53E−01	3.92E−02	**0.00E+00**	**0.00E+00**
F_6	4.34E−01	4.10E−07	2.12E+01	6.03E−14	1.20E−12	8.32E−02	2.50E−04	1.44E−13	**0.00E+00**
F_7	**0.00E+00**	1.40E−02	4.22E−04	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.17E−02	**0.00E+00**	**0.00E+00**
F_8	4.74E+02	1.69E+03	**0.00E+00**	3.48E+01	1.43E−03	2.83E+05	2.27E+03	2.32E−03	**0.00E+00**
F_9	3.71E−03	5.86E+02	1.02E+02	5.63E+02	2.20E+02	3.00E−02	2.91E+01	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	3.30E+01	1.66E+01	**0.00E+00**	**0.00E+00**	4.73E−02	2.05E−01	**0.00E+00**	**0.00E+00**
F_{11}	8.58E−04	7.32E+01	1.64E+02	1.09E−01	2.10E+02	3.05E−01	2.60E+01	**0.00E+00**	**0.00E+00**
F_{12}	2.71E+00	1.03E+01	4.17E+02	3.28E−03	3.91E+01	3.79E+01	5.01E+00	5.36E−04	**0.00E+00**
F_{13}	5.87E+01	2.70E+06	4.21E+02	8.35E+01	1.75E+02	3.42E+09	1.40E+02	**8.50E+00**	8.70E+01
F_{14}	2.21E+00	1.66E+02	2.55E+02	**0.00E+00**	2.04E+02	3.92E+03	1.24E+00	1.16E+01	**0.00E+00**
F_{15}	**0.00E+00**	8.13E+00	6.30E−01	**0.00E+00**	**0.00E+00**	3.99E−02	6.56E−02	**0.00E+00**	**0.00E+00**
F_{16}	3.52E+00	2.23E+01	8.59E+02	1.61E−02	1.04E+02	1.96E+01	8.29E+00	**0.00E+00**	**0.00E+00**
F_{17}	1.58E+01	1.47E+05	1.51E+03	9.92E+01	4.17E+02	2.34E+05	1.97E+02	**6.65E−03**	1.20E+01
F_{18}	8.76E−01	7.00E+01	3.07E+02	**0.00E+00**	1.22E+02	3.05E+01	3.34E+00	4.46E−01	**0.00E+00**
F_{19}	**0.00E+00**	5.45E+02	2.02E+01	**0.00E+00**	**0.00E+00**	2.71E−01	1.43E−01	**0.00E+00**	**0.00E+00**

Table 7. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 100$

With the increase of dimension, the mDE-bcM has more obvious advantages, the number of functions on which it can obtain the optimal value is increasing, while the ability of other algorithms to obtain the optimal value is decreasing. Therefore, we can conclude that mDE-bcM is more suitable for dealing with large-scale high-dimension optimization problems.

The average ranking, the number of functions with best results and the nWins value for mDE-bcM and the first group of algorithms are reported in Table 11. Limited to space, we only give the statistical results at dimension 1 000.

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	8.55E+00	8.34E−01	**0.00E+00**	**0.00E+00**	2.29E+00	2.03E+01	8.03E−02	**0.00E+00**	**0.00E+00**
F_2	1.05E+02	1.03E+02	1.16E−09	3.36E+01	4.54E−09	1.03E+02	8.03E+00	4.15E+01	**0.00E+00**
F_3	3.32E+05	2.01E+07	**8.91E+01**	2.50E+02	1.69E+02	4.82E+04	2.91E+02	1.35E+02	1.99E+02
F_4	6.98E+00	5.40E+02	6.48E+02	4.43E+00	2.34E−12	6.25E+00	3.52E−01	9.27E−13	**0.00E+00**
F_5	4.05E−01	8.76E−03	**0.00E+00**	**0.00E+00**	5.42E−03	6.43E−02	2.68E−02	**0.00E+00**	**0.00E+00**
F_6	7.14E−01	1.23E+00	2.14E+01	1.19E−13	2.38E−12	2.73E−02	6.22E−01	**0.00E+00**	**0.00E+00**
F_7	**0.00E+00**	2.59E−01	1.17E−01	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.82E−02	**0.00E+00**	**0.00E+00**
F_8	5.76E+03	9.38E+03	**0.00E+00**	7.23E+02	1.42E+01	4.47E+05	1.34E+04	8.71E−01	**0.00E+00**
F_9	8.79E−03	1.19E+03	3.75E+02	1.17E+03	4.27E+02	3.00E−02	6.22E+01	**0.00E+00**	**0.00E+00**
F_{10}	4.19E−02	7.13E+01	4.43E+01	**0.00E+00**	**0.00E+00**	1.59E−02	1.04E+00	**0.00E+00**	**0.00E+00**
F_{11}	5.07E−03	3.85E+02	8.03E+02	3.50E−01	4.28E+02	4.89E−03	5.93E+01	**0.00E+00**	**0.00E+00**
F_{12}	3.61E+00	7.44E+01	9.06E+02	1.75E−02	8.42E+01	4.63E+01	1.00E+01	**0.00E+00**	**0.00E+00**
F_{13}	1.49E+02	5.75E+06	9.43E+02	1.68E+02	2.53E+02	3.16E+09	1.71E+02	**9.45E+01**	1.87E+02
F_{14}	4.75E+00	4.29E+02	6.09E+02	9.76E−01	3.98E+02	4.09E+03	3.75E+00	1.20E+01	**0.00E+00**
F_{15}	**0.00E+00**	2.14E+01	1.75E+00	**0.00E+00**	**0.00E+00**	5.38E−03	3.80E−01	**0.00E+00**	**0.00E+00**
F_{16}	3.70E+00	1.60E+02	1.92E+03	6.02E−02	1.97E+02	9.49E+00	1.74E+01	**0.00E+00**	**0.00E+00**
F_{17}	2.23E+01	1.75E+05	3.36E+03	7.55E+01	6.07E+02	2.36E+05	1.56E+02	8.39E−02	1.20E+01
F_{18}	2.37E+00	2.12E+02	6.89E+02	4.29E−04	2.34E+02	1.69E+01	8.85E+00	8.93E−11	**0.00E+00**
F_{19}	4.19E−02	2.06E+03	7.52E+02	**0.00E+00**	**0.00E+00**	1.00E−01	2.15E+00	**0.00E+00**	**0.00E+00**

Table 8. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 200$

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	2.46E+01	2.84E−12	**0.00E+00**	**0.00E+00**	5.77E−12	1.34E+01	**0.00E+00**	3.92E−13	**0.00E+00**
F_2	1.44E+02	1.29E+02	3.48E−04	7.86E+01	5.34E−06	9.23E+01	2.04E+01	4.56E+01	**0.00E+00**
F_3	1.12E+05	1.14E+06	3.58E+02	6.07E+02	2.20E+02	2.62E+04	5.97E+02	**4.16E+02**	4.99E+02
F_4	1.63E+01	1.91E+03	2.10E+03	1.78E+02	5.62E−12	1.31E+00	1.45E+00	1.91E−11	**0.00E+00**
F_5	4.73E−01	6.98E−03	2.96E−04	**0.00E+00**	4.24E−03	7.48E−03	3.03E−02	1.83E−13	**0.00E+00**
F_6	1.06E+00	5.16E+00	2.15E+01	2.63E−13	6.18E−12	4.63E−01	1.21E+00	3.56E−14	**0.00E+00**
F_7	**0.00E+00**	1.27E−01	7.21E+153	4.69E−14	1.46E−12	**0.00E+00**	8.06E−03	**0.00E+00**	**0.00E+00**
F_8	6.70E+04	7.22E+04	2.36E−06	1.32E+04	6.16E+03	3.21E+05	7.05E+04	5.48E+02	**0.00E+00**
F_9	1.12E−02	3.00E+03	1.74E+03	2.53E+03	1.00E+03	3.00E−02	1.75E+02	**0.00E+00**	**0.00E+00**
F_{10}	2.93E−01	1.86E+02	1.27E+02	2.80E−01	**0.00E+00**	8.41E−03	3.29E+01	**0.00E+00**	**0.00E+00**
F_{11}	2.43E−01	1.81E+03	4.16E+03	4.21E+01	1.00E+03	2.22E−03	1.77E+02	**0.00E+00**	**0.00E+00**
F_{12}	1.16E+01	4.48E+02	2.58E+03	2.55E+01	2.47E+02	4.61E+01	1.73E+01	**0.00E+00**	**0.00E+00**
F_{13}	4.02E+02	3.22E+07	2.87E+03	4.00E+02	5.05E+02	2.97E+09	5.75E+02	**3.23E+02**	4.87E+02
F_{14}	1.16E+01	1.46E+03	1.95E+03	5.65E+01	1.10E+03	3.91E+03	9.00E+00	1.68E+01	**0.00E+00**
F_{15}	4.19E−02	6.01E+01	2.82E+262	5.53E+00	1.08E−12	2.84E−03	2.25E+00	**0.00E+00**	**0.00E+00**
F_{16}	1.32E+01	9.55E+02	5.45E+03	1.08E−01	4.99E+02	5.82E+00	4.87E+01	**0.00E+00**	**0.00E+00**
F_{17}	6.94E+01	8.40E+05	9.59E+03	1.38E+02	7.98E+02	2.38E+05	3.94E+02	6.65E+01	**1.20E+01**
F_{18}	3.87E+00	7.32E+02	2.05E+03	2.41E−03	5.95E+02	9.43E+00	3.28E+01	**0.00E+00**	**0.00E+00**
F_{19}	8.39E−02	1.76E+03	2.44E+06	**0.00E+00**	**0.00E+00**	1.00E−01	5.00E+01	**0.00E+00**	**0.00E+00**

Table 9. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 500$

In Table 11, the algorithm that obtains the best ranking, number of functions with the best results and the nWins value is mDE-bcM, followed by mDE-bES. The average ranking of mDE-bcM is 1.21, the numbers of functions that mDE-bcM is better than that of DE, CHC, MA-SSW, MTS-LS1, SaDE, EvoPR, mDE-bES are 18, 19, 18, 13, 18, 19, 8, respectively. This shows that mDE-bcM has obviously good optimization performance.

The Friedman test reported a p-value = 9.42E−14 for Table 10, which is below the significance level $\alpha = 0.05$, and it means that there are statistical differences

	DE	CHC	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	3.71E+01	1.36E−11	**0.00E+00**	1.15E−11	3.49E+01	4.00E−05	8.24E−13	**0.00E+00**
F_2	1.63E+02	1.44E+02	1.39E+02	2.25E−02	1.43E+02	3.21E+01	5.97E+01	**0.00E+00**
F_3	1.59E+05	8.75E+03	1.22E+03	**2.10E+02**	1.62E+05	1.12E+03	9.00E+02	9.99E+02
F_4	3.47E+01	4.76E+03	1.58E+03	1.15E−11	3.21E+01	4.08E+02	4.03E+01	**0.00E+00**
F_5	7.36E−01	7.02E−03	5.92E−04	3.55E−03	6.33E−01	3.72E−02	**0.00E+00**	**0.00E+00**
F_6	8.70E−01	1.38E+01	1.46E−09	1.24E−11	4.28E−01	1.97E+00	1.28E−12	**8.88E−16**
F_7	**0.00E+00**	3.52E−01	6.23E−13	**0.00E+00**	**0.00E+00**	1.50E−04	**0.00E+00**	**0.00E+00**
F_8	3.15E+05	3.11E+05	7.49E+04	1.23E+05	3.08E+05	2.15E+05	7.98E+03	**0.00E+00**
F_9	6.26E−02	6.11E+03	5.99E+03	1.99E+03	3.00E−02	4.07E+02	**0.00E+00**	**0.00E+00**
F_{10}	1.67E−01	3.83E+02	2.09E−05	**0.00E+00**	1.47E−01	3.86E+02	**0.00E+00**	**0.00E+00**
F_{11}	4.42E−02	4.82E+03	5.27E+01	1.99E+03	4.56E−01	3.96E+02	**0.00E+00**	**0.00E+00**
F_{12}	2.58E+01	1.05E+03	9.48E−02	5.02E+02	3.43E+01	3.23E+01	**0.00E+00**	**0.00E+00**
F_{13}	8.24E+04	6.66E+07	1.02E+03	8.87E+02	3.27E+09	1.13E+03	**6.34E+02**	9.87E+02
F_{14}	2.39E+01	3.62E+03	7.33E+02	2.23E+03	3.71E+03	4.31E+02	2.45E+01	**0.00E+00**
F_{15}	2.11E−01	8.37E+01	1.16E−13	**0.00E+00**	1.11E−01	1.26E+02	**0.00E+00**	**0.00E+00**
F_{16}	1.83E+01	2.32E+03	2.19E+00	1.00E+03	2.37E+01	8.44E+01	**0.00E+00**	**0.00E+00**
F_{17}	1.76E+05	2.04E+07	3.26E+02	1.56E+03	1.62E+05	6.75E+02	1.88E+02	**1.20E+01**
F_{18}	7.55E+00	1.72E+03	2.58E+01	1.21E+03	3.54E+01	1.95E+02	2.49E−01	**0.00E+00**
F_{19}	2.51E−01	4.20E+03	1.56E−12	**0.00E+00**	9.32E−01	2.03E+02	**0.00E+00**	**0.00E+00**

Table 10. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 1\,000$

	DE	CHC	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
Ranking	4.58	6.37	3.42	3.42	4.79	4.79	1.79	1.21
# Best	1	0	1	5	1	0	10	17
nWins-better	18	19	18	13	18	19	8	–
nWins-similar	1	0	1	4	1	0	9	–
nWins-worse	0	0	0	2	0	0	2	–
p-value	2.21E−05	1.31E−05	2.21E−05	4.50E−03	2.21E−05	1.31E−05	1.14E−02	–

Table 11. Average ranking, number of functions with the best results, nWins value and the Friedman test for mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 1\,000$

as the negation of the null hypothesis. Used mDE-bcM as the control algorithm, further the Friedman tests show that mDE-bcM have been significant differences with other 7 algorithms.

The comparison results of mDE-bcM and the second group of algorithms are in Table 12, we run for 25 times on every function $F_1 - F_{19}$, where $D = 1\,000$, except for our mDE-bcM, FEs is 5×10^6.

The average ranking, the number of functions with best results and the nWins value for mDE-bcM and the second group of algorithms are reported in Table 13.

In Table 13, the algorithms that obtain the best ranking are mDE-bcM and MOS-SOCO2011, number of functions with the best results is mDE-bcM, followed by MOS-SOCO2011, and the numbers of functions that mDE-bcM is better than that of MOS-CEC2013, MOS-SOCO2011, jDElscop, GaDE, DECC-G, CCJADE, L-SHADE and LM-CMA-ES are 14, 3, 9, 15, 15, 12, 18 and 18, respectively. This shows that mDE-bcM and MOS-SOCO2011 have the same performance, but they have significant advantages over other algorithms.

	MOS-CEC2013	MOS-SOCO2011	jDElscop	GaDE	DECC-G	CCJADE	L-SHADE	LM-CMA-ES	mDE-bcM
F_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.26E−06	1.80E−15	7.82E−04	2.16E−13	**0.00E+00**
F_2	1.10E+02	5.88E−01	2.46E+01	5.46E+01	1.31E+03	1.38E+02	4.55E+01	1.49E+02	**0.00E+00**
F_3	7.39E+00	7.09E+01	8.51E+02	9.47E+02	**1.09E+00**	3.74E+02	1.64E+03	6.04E+02	9.99E+02
F_4	**0.00E+00**	**0.00E+00**	2.39E−01	3.79E−02	2.16E+11	8.62E−01	1.71E+03	1.62E+04	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	0.00E+00	5.91E−04	8.30E+06	4.85E−04	6.07E−03	1.13E−13	**0.00E+00**
F_6	**0.00E+00**	**0.00E+00**	1.16E−12	2.55E−14	9.63E−01	3.28E−13	1.92E+00	1.99E+01	8.88E−16
F_7	2.56E−12	**0.00E+00**	**0.00E+00**	**0.00E+00**	Inf	**0.00E+00**	1.34E+00	4.22E+569	**0.00E+00**
F_8	5.98E+03	1.66E+05	3.17E+04	1.55E+04	1.11E+05	1.92E+06	5.76E+04	1.66E−06	**0.00E+00**
F_9	2.51E+03	0.00E+00	9.21E−08	4.29E−04	1.78E+01	9.47E−01	1.84E+03	9.17E+03	**0.00E+00**
F_{10}	1.58E+00	0.00E+00	0.00E+00	3.36E−01	1.94E+02	**0.00E+00**	3.41E+02	5.63E+02	**0.00E+00**
F_{11}	2.54E+03	**0.00E+00**	4.98E−08	8.58E−04	1.76E+01	8.68E−01	1.89E+03	9.22E+03	**0.00E+00**
F_{12}	9.99E+02	**0.00E+00**	**0.00E+00**	2.90E−12	**0.00E+00**	4.46E+00	1.27E+03	2.69E+03	**0.00E+00**
F_{13}	1.23E+03	1.69E+02	6.67E+02	7.19E+02	3.86E+03	**9.82E+01**	2.40E+03	3.17E+03	9.87E+02
F_{14}	3.37E+03	**0.00E+00**	4.03E−01	7.72E−03	1.59E+02	3.46E−01	2.65E+03	1.28E+04	**0.00E+00**
F_{15}	1.93E−12	**0.00E+00**	**0.00E+00**	8.40E−02	1.84E+01	**0.00E+00**	2.11E+01	3.45E+418	**0.00E+00**
F_{16}	8.02E+03	**0.00E+00**	**0.00E+00**	1.67E−12	0.00E+00	3.92E+00	2.23E+03	5.38E+03	**0.00E+00**
F_{17}	3.55E+11	6.71E+01	1.71E+02	2.18E+02	1.98E+02	**7.83E+00**	2.80E+03	7.60E+03	1.20E+01
F_{18}	2.03E+03	**0.00E+00**	3.28E−12	1.31E−07	8.43E+00	5.89E−01	9.12E+02	5.68E+03	**0.00E+00**
F_{19}	2.05E+03	**0.00E+00**	**0.00E+00**	2.10E−01	1.12E+02	**0.00E+00**	2.57E+02	9.65E+04	**0.00E+00**

Table 12. Comparison of mDE-bcM and the second group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 1\,000$

	MOS-CEC2013	MOS-SOCO2011	jDElscop	GaDE	DECC-G	CCJADE	L-SHADE	LM-CMA-ES	mDE-bcM
Ranking	4.11	1.68	2.37	3.00	4.63	3.16	5.53	6.05	1.68
# Best	4	14	8	2	4	6	0	0	15
nWins-better	14	3	9	15	15	12	18	18	–
nWins-similar	3	13	8	2	3	4	0	0	–
nWins-worse	2	3	2	2	1	3	1	1	–
p-value	2.70E−03	1	3.48E−02	1.60E−04	3.00E−03	2.01E−02	1.31E−05	9.62E−05	–

Table 13. Average ranking, number of functions with the best results, nWins value and the Friedman test for mDE-bcM and the second group of algorithms for 25 runs on functions $F_1 - F_{19}$, where $D = 1\,000$

The Friedman test reported a p-value = 6.51E−14 for Table 12, which is below the significance level $\alpha = 0.05$, and means that there are statistical differences as the negation of the null hypothesis. Used mDE-bcM as the control algorithm, further the Friedman tests show that mDE-bcM has been significant differences with MOS-CEC2013, GaDE, CCJADE, L-SHADE and LM-CMA-ES, for jDElscop and DECC-G, the p-values are nonetheless very close to the significance level, but for MOS-SOCO2011, the p-value = 1 shows that there are not significant differences between mDE-bcM and MOS-SOCO2011, MOS-SOCO2011 is indeed a competitive algorithm.

It is worth noting that MOS-CEC2013 performs very well on the CEC 2013 benchmark suite, but generally on the SOCO 2011 benchmark suite, which proves once again the limitation of the algorithm in solving the problem.

7 CONCLUSIONS

This research proposes a new algorithm called mDE-bcM for solving large-scale global optimization problems. The mDE-bcM divides the population into three subpopulations with the fitness values, the second and the third of which employs

a modified mutation strategy called best-and-current mutation strategy, three subpopulations evolved independently and then fused after one evolution. The mDE-bcM was tested on a set of benchmark functions provided for the Soft Computing special issue on scalability of evolutionary algorithms for large-scale continuous optimization problems. After comparing with other 16 state-of-the-art algorithms in use, it shows a very competitive performance on the SOCO 2011 benchmark suite.

For future work, we intend to use ensemble mutation strategies and success rate of mutation strategies for difficult problems, and test our algorithm on extra sets of competitive LSGO benchmarks, such as presented in the CEC'11 [47] and CEC'13 [32] special sessions.

Acknowledgement

This work is supported by the NSFC (National Natural Science Foundation of China) project (grants No. 62066041, 41861047) and the Northwest Normal University young teachers' scientific research capability upgrading program (NWNU-LKQN-17-6).

REFERENCES

[1] Storn, R.—Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, Vol. 11, 1997, No. 4, pp. 341–359, doi: 10.1023/A:1008202821328.

[2] Price, K.—Storn, R. M.—Lampinen, J. A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin, Heidelberg, Natural Computing Series, 2005, doi: 10.1007/3-540-31306-0.

[3] Das, S.—Suganthan, P. N.: Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation, Vol. 15, 2011, No. 1, pp. 4–31, doi: 10.1109/TEVC.2010.2059031.

[4] Das, S.—Mullick, S. S.—Suganthan, P. N.: Recent Advances in Differential Evolution – An Updated Survey. Swarm and Evolutionary Computation, Vol. 27, 2016, pp. 1–30, doi: 10.1016/j.swevo.2016.01.004.

[5] Floudas, C. A.—Gounaris, C. E.: A Review of Recent Advances in Global Optimization. Journal of Global Optimization, Vol. 45, 2009, No. 1, pp. 3–38, doi: 10.1007/s10898-008-9332-8.

[6] Maučec, M. S.—Brest, J.: A Review of the Recent Use of Differential Evolution for Large-Scale Global Optimization: An Analysis of Selected Algorithms on the CEC 2013 LSGO Benchmark Suite. Swarm and Evolutionary Computation, Vol. 50, 2019, Art. No. 100428, doi: 10.1016/j.swevo.2018.08.005.

[7] Qin, A. K.—Huang, V. L.—Suganthan, P. N.: Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. IEEE Trans-

actions on Evolutionary Computation, Vol. 13, 2009, No. 2, pp. 398–417, doi: 10.1109/TEVC.2008.927706.

[8] Mallipeddi, R.—Lee, M.: An Evolving Surrogate Model-Based Differential Evolution Algorithm. Applied Soft Computing, Vol. 34, 2015, pp. 770–787, doi: 10.1016/j.asoc.2015.06.010.

[9] Neri, F.—Tirronen, V.: Recent Advances in Differential Evolution: A Survey and Experimental Analysis. Artificial Intelligence Review, Vol. 33, 2010, No. 1-2, pp. 61–106, doi: 10.1007/s10462-009-9137-2.

[10] Weber, M.—Neri, F.—Tirronen, V.: Distributed Differential Evolution with Explorative-Exploitative Population Families. Genetic Programming and Evolvable Machines, Vol. 10, 2009, No. 4, pp. 343–371, doi: 10.1007/s10710-009-9089-y.

[11] Ali, M. Z.—Awad, N. H.—Suganthan, P. N.: Multi-Population Differential Evolution with Balanced Ensemble of Mutation Strategies for Large-Scale Global Optimization. Applied Soft Computing, Vol. 33, 2015, pp. 304–327, doi: 10.1016/j.asoc.2015.04.019.

[12] Lampinen, J.: Differential Evolution – New Naturally Parallel Approach for Engineering Design Optimization. In: Topping, B. H. V. (Ed.): Developments in Computational Mechanics with High Performance Computing. Civil-Comp Press, Ithaca, UK, 1999, pp. 217–228.

[13] Zaharie, D.: Control of Population Diversity and Adaptation in Differential Evolution Algorithms. In: Matousek, D., Osmera, P. (Eds.): Proceedings of MENDEL International Conference on Software Computing, 2003, pp. 41–46.

[14] Weber, M.—Neri, F.—Tirronen, V.: Shuffle or Update Parallel Differential Evolution for Large-Scale Optimization. Soft Computing, Vol. 15, 2011, No. 11, pp. 2089–2107, doi: 10.1007/s00500-010-0640-9.

[15] Chen, Y.—Lin, Y.—Hu, X.: Parallel Differential Evolution with Multi-Population and Multi-Strategy. Journal of Frontiers of Computer Science and Technology, Vol. 8, 2014, No. 12, pp. 1502–1510.

[16] Wu, G.—Mallipeddi, R.—Suganthan, P. N.—Wang, R.—Chen, H.: Differential Evolution with Multi-Population Based Ensemble of Mutation Strategies. Information Sciences, Vol. 329, 2016, pp. 329–345, doi: 10.1016/j.ins.2015.09.009.

[17] Zhang, J.—Sanderson, A. C.: JADE: Adaptive Differential Evolution with Optional External Archive. IEEE Transactions on Evolutionary Computation, Vol. 13, 2009, No. 5, pp. 945–958, doi: 10.1109/TEVC.2009.2014613.

[18] Kong, X.—Gao, L.—Ouyang, H.—Shao, Y.: Adaptive Differential Evolution Algorithm with Bidirectional Randomly Multi-Mutation Strategy. Computer Integrated Manufacturing Systems, Vol. 20, 2014, No. 8, pp. 1948–1958, doi: 10.13196/j.cims.2014.08.kongxiangyong.1948.11.20140817 (in Chinese).

[19] Zhang, X.—Yuen, S. Y.: A Directional Mutation Operator for Differential Evolution Algorithms. Applied Soft Computing, Vol. 30, 2015, pp. 529–548, doi: 10.1016/j.asoc.2015.02.005.

[20] Qiu, X.—Jiang, Y.—Li, B.: Fractal Mutation Factor Correcting Differential Evolution Algorithm. Pattern Recognition and Artificial Intelligence, Vol. 28, 2015, No. 2, pp. 132–138, doi: 10.16451/j.cnki.issn1003-6059.201502005 (in Chinese).

[21] Parouha, R. P.—Das, K. N.: A Memory Based Differential Evolution Algorithm for Unconstrained Optimization. Applied Soft Computing, Vol. 38, 2016, pp. 501–517, doi: 10.1016/j.asoc.2015.10.022.

[22] Elsayed, S. M.—Sarker, R. A.—Essam, D. L.: Differential Evolution with Multiple Strategies for Solving CEC2011 Real-World Numerical Optimization Problems. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2011), New Orleans, LA, USA, IEEE, 2011, pp. 1041–1048, doi: 10.1109/CEC.2011.5949732.

[23] Tong, L.—Dong, M.—Jing, C.: An Improved Multi-Population Ensemble Differential Evolution. Neurocomputing, Vol. 290, 2018, pp. 130–147, doi: 10.1016/j.neucom.2018.02.038.

[24] Xu, B.—Tao, L.—Chen, X.—Cheng, W.: Adaptive Differential Evolution with Multi-Population-Based Mutation Operators for Constrained Optimization. Soft Computing, Vol. 23, 2019, pp. 3423–3447, doi: 10.1007/s00500-017-3001-0.

[25] Herrera, F.—Lozano, M.—Molina, D.: Test Suite for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and Other Metaheuristics for Large Scale Continuous Optimization Problems. Technical Report, SCI2S, University of Granada, Spain, 2010.

[26] Lozano, M.—Molina, D.—Herrera, F.: Editorial Scalability of Evolutionary Algorithms and Other Metaheuristics for Large-Scale Continuous Optimization Problems. Soft Computing, Vol. 15, 2011, No. 11, pp. 2085–2087, doi: 10.1007/s00500-010-0639-2.

[27] Zielinski, K.—Peters, D.—Laur, R.: Run Time Analysis Regarding Stopping Criteria for Differential Evolution and Particle Swarm Optimization. Proceedings of the 1st International Conference on Experiments/Process/System Modelling/Simulation/Optimization, 2005, pp. 1–8.

[28] Opara, K.—Arabas, J.: Differential Evolution: A Survey of Theoretical Analyses. Swarm and Evolutionary Computation, Vol. 44, 2019, pp. 546–558, doi: 10.1016/j.swevo.2018.06.010.

[29] LaTorre, A.—Muelas, S.—Peña, J.-M.: A Comprehensive Comparison of Large Scale Global Optimizers. Information Sciences, Vol. 316, 2015, pp. 517–549, doi: 10.1016/j.ins.2014.09.031.

[30] LaTorre, A.—Muelas, S.—Peña, J.-M.: Large Scale Global Optimization: Experimental Results with MOS-Based Hybrid Algorithms. 2013 IEEE Congress on Evolutionary Computation (CEC 2013), Cancún, Mexico, 2013, pp. 2742–2749, doi: 10.1109/CEC.2013.6557901.

[31] Tang, K.—Li, X.—Suganthan, P. N.—Yang, Z.—Weise, T.: Benchmark Functions for the CEC 2010 Special Session and Competition on Large Scale Global Optimization. Technical Report, 2010.

[32] Li, X.—Tang, K.—Omidvar, M. N.—Yang, Z.—Qin, K.: Benchmark Functions for the CEC 2013 Special Session and Competition on Large Scale Global Optimization. Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

[33] LaTorre, A.—Muelas, S.—Peña, J.: A MOS-Based Dynamic Memetic Differential Evolution Algorithm for Continuous Optimization: A Scalability Test. Soft Computing, Vol. 15, 2011, No. 11, pp. 2187–2199, doi: 10.1007/s00500-010-0646-3.

[34] Falco, I.—Cioppa, A.—Trunfio, G.: Investigating Surrogate-Assisted Cooperative Coevolution for Large-Scale Global Optimization. Information Sciences, Vol. 482, 2019, pp. 1–26, doi: 10.1016/j.ins.2019.01.009.

[35] Derrac, J.—García, S.—Molina, D.—Herrera, F.: A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. Swarm and Evolutionary Computation, Vol. 1, 2011, No. 1, pp. 3–18, doi: 10.1016/j.swevo.2011.02.002.

[36] Eshelman, L. J.—Schaffer, J. D.: Real-Coded Genetic Algorithms and Interval-Schemata. Foundations of Genetic Algorithms, Vol. 2, 1993, pp. 187–202, doi: 10.1016/B978-0-08-094832-4.50018-0.

[37] Auger, A.—Hansen, N.: A Restart CMA Evolution Strategy with Increasing Population Size. 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, UK, 2005, Vol. 2, pp. 1769–1776, doi: 10.1109/CEC.2005.1554902.

[38] Molina, D.—Lozano, M.—Sanchez, A. M.—Herrera, F.: Memetic Algorithms Based on Local Search Chains for Large Scale Continuous Optimisation Problems: MA-SSW-Chains. Soft Computing, Vol. 15, 2011, No. 11, pp. 2201–2220, doi: 10.1007/s00500-010-0647-2.

[39] Tseng, L.—Chen, C.: Multiple Trajectory Search for Large Scale Global Optimization. 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, China, 2008, pp. 3052–3059, doi: 10.1109/CEC.2008.4631210.

[40] Liang, J. J.—Suganthan, P. N.—Deb, K.: Novel Composition Test Functions for Numerical Global Optimization. Proceedings 2005 IEEE Swarm Intelligence Symposium (SIS 2005), 2005, pp. 68–75, doi: 10.1109/SIS.2005.1501604.

[41] Duarte, A.—Marti, R.—Gortazar, F.: Path Relinking for Large-Scale Global Optimization. Soft Computing, Vol. 15, 2011, No. 11, pp. 2257–2273, doi: 10.1007/s00500-010-0650-7.

[42] Brest, J.—Maučec, M. S.: Self-Adaptive Differential Evolution Algorithm Using Population Size Reduction and Three Strategies. Soft Computing, Vol. 15, 2011, No. 11, pp. 2157–2174, doi: 10.1007/s00500-010-0644-5.

[43] Yang, Z.—Tang, K.—Yao, X.: Scalability of Generalized Adaptive Differential Evolution for Large-Scale Continuous Optimization. Soft Computing, Vol. 15, 2011, No. 11, pp. 2141–2155, doi: 10.1007/s00500-010-0643-6.

[44] Yang, Z.—Tang, K.—Yao, X.: Large Scale Evolutionary Optimization Using Cooperative Coevolution. Information Sciences, Vol. 178, 2008, No. 15, pp. 2985–2999, doi: 10.1016/j.ins.2008.02.017.

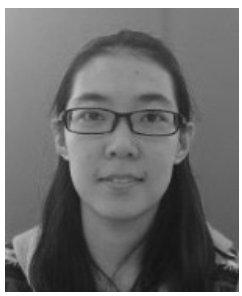
[45] Tanabe, R.—Fukunaga, A. S.: Improving the Search Performance of SHADE Using Linear Population Size Reduction. 2014 IEEE Congress on Evolutionary Computation (CEC 2014), 2014, pp. 1658–1665, doi: 10.1109/CEC.2014.6900380.

[46] Loshchilov, I.: A Computationally Efficient Limited Memory CMA-ES for Large Scale Optimization. Proceedings of the 2014 Annual Conference on Ge-

netic and Evolutionary Computation (GECCO '14), 2014, pp. 397–404, doi: 10.1145/2576768.2598294.

[47] Li, X.—Tang, K.—Omidvar, M.—Yang, Z.—Qin, K.: Benchmark Functions for the CEC '2011 Special Session and Competition on Large Scale Global Optimization. Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

Yongjie Ma graduated from the University of Electronic Science and Technology of China with a Master's degree in electromagnetic measurement technology and instruments in 1998. He received his Ph.D. in traffic information and control from Lanzhou Jiaotong University in 2011. He is currently Professor in the School of Physics and Electronic Engineering of Northwest Normal University. His research interests include evolutionary algorithms and their engineering application, intelligent transportation image processing. He published more than 70 technical papers in journals.



Lin Zhu graduated from Northwest Normal University in 2017 with a Master's degree. Her main research interest are evolutionary algorithms, and she has published 3 research papers in journals.

Yulong Bai received his Ph.D. in natural science (cartography and geography information systems) from Graduate University of Chinese Academy of Science, China in January 2012. He is currently Professor at College of Physics and Electrical Engineering, Northwest Normal University, China. His research interests include control theory and application, data assimilation methods and chaos. He published more than 30 papers in peer-reviewed journals.