Soft Comput (2013) 17:1511–1535
DOI 10.1007/s00500-012-0962-x

METHODOLOGIES AND APPLICATION

Diversity-maintained differential evolution embedded with gradient-based local search

Weicheng Xie · Wei Yu · Xiufen Zou

Published online: 29 December 2012

Abstract Differential evolution (DE) has been used to solve real-parameter optimization problems with nonlinear and multimodal functions for more than a decade of years. However, it is pointed out that this classical DE harbors restricted efficiency and limited local search ability. Inspired by that gradient-based algorithms have powerful local search ability, we propose a new algorithm, which is diversity-maintained DE based on gradient local search (namely, DMGBDE), by incorporating approximate gradient-based algorithms into the DE search while maintaining the diversity of the population. The primary novelties of the proposed DMGBDE are the following: (1) the gradient-based algorithm is embedded into DE in a different manner and (2) a diversity-maintained mutation is introduced to slow down the learning procedure from the searched best individual. We conduct numerical experiments with a number of benchmark problems to measure the performance of the proposed DMGBDE. Simulation results show that the proposed DMGBDE outperforms classical DE and variant without gradient local search or diversity-based mutation. Moreover, comparison with some other recently reported approaches indicates that our proposed DMGBDE is rather competitive.

Keywords Differential evolution · Gradient local search · Diversity-maintained mutation · Ability to continue searching

Communicated by F. Herrera.

W. Xie · W. Yu · X. Zou (✉)
School of Mathematics and Statistics, Wuhan University, Wuhan 430072, China
e-mail: zouxiufen@yahoo.com; xfzou@whu.edu.cn

1 Introduction

For the last few decades, different kinds of stochastic algorithms were proposed to solve real engineering problems in a wide variety of fields. Different from classical gradient-based optimization algorithms, a stochastic algorithm (namely, as meta-heuristic algorithm) adopts one greedy strategy or learns from the behavior patterns of creatures to proceed to global optimum from multiple positions of the searching region. These meta-heuristic algorithms, without utilizing the gradient information of the objective function, are often the real alterative for a variety of non-differentiable and non-convex problems which are difficult for gradient-based algorithms.

On one hand, although more likely to jump out of local optimum, these stochastic algorithms are reported to be more time-consuming than classical gradient-based algorithms when confronting problems with a large number of optimization variables. On the other hand, a wide variety of problems arise in practical applications where part of gradient information with respect to (w.r.t.) optimization variables is easily obtained without difference approximation. Such as in Koh (2007), the derivatives w.r.t. the variables in the second level are often obtained easily when fixing the values of the variables in the first level. Thus, w.r.t. some optimization problems it seems beneficial to add the gradient-based algorithms into the searching because these algorithms harbor powerful local search ability. Lots of work referring to this idea sprang up. The quasi-Newton (QN) method was incorporated into the multi-swarm particle swarm optimizer to improve the local searching ability (Zhao et al. 2008). The information implied in the approximate gradient was utilized to search a local optimal solution in the climb process of the monkey algorithm (Zhao and Tang 2008). Hybrid of particle swarm

optimization (PSO) and gradient-based sequential quadratic programming (SQP) was proposed for optimum structural design (Plevris and Papadrakakis 2011), where the first phase solution found by PSO was set as the initial point for the SQP searching in the second phase. The bias of derivatives approximation of the approximated function was encoded into the cost function of an adaptive PSO algorithm to train network (Han et al. 2010). Newton's method was embedded in the velocity update equation to improve the effect of cognition influence (Zahara et al. 2009). PSO and quasi-Newton method were combined to solve an engineering optimization problem (Ghaffari-Miab et al. 2007).

Among the stochastic algorithms, DE which is proposed by Storn and Price (1995); Storn and Price (1997) has attracted lot of attention as its simple structure, easy to implement, and also relatively good performance on multimodal problems (Rönkkönen et al. 2005). However, DE behaves poorly on some (e.g. some large scale) practical applications. Similar to other stochastic algorithms, other gradient-based algorithms were also incorporated into the DE considering its limited local search ability, although it was showed by Dasgupta et al. (2009) that each individual in DE performed approximately as a gradient-based search agent. A gradient-based mutation was introduced into DE to find feasible point using the gradient of constraints at an infeasible point when considering constraint optimization problems (Takahama and Sakai 2006). An approximate solution obtained with the DE algorithm was used to initialize the gradient algorithms with adjustable control weight by Lopez Cruz et al. (2003). Economic dispatch problem was solved by hybrid of interior point algorithm (IPA) and DE, where the algorithm was divided into two stages: the first stage was to employ IPA to minimize the cost function without considering the valve point effect, based on the obtained solution; the second stage was executed by minimizing the whole cost function with DE (Duvvuru and Swarup 2011). Ten percent of individuals were randomly chosen for line minimization search with classical optimization algorithms as the DE algorithm proceeded (Masters and Land 1997). It was suggested that each offspring generated by DE was first fine-tuned by conjugate gradient algorithm before competing with their parents (Bandurski and Kwedlo 2010). A local search procedure was performed after 200 generations on randomly chosen 5 % individuals of best 50 % individuals (Qin and Suganthan 2005). In Zamuda et al. (2009), the sequential quadratic programming (SQP) method was incorporated into a self-adaptive DE for solving constrained multi-objective optimization.

Gradient algorithms can boost the local search efficiency, it may render the hybrid algorithms stagnating to a local optimum when inappropriate strategy is employed. In this work, we use the local search on the current best individual only if the best individual in current generation has been renewed, which means that it is better than that in the last generation. Meanwhile, some competitive individuals are also local searched when the searched best individual has not been improved for certain generations, in this way, the possibility that the population stagnates to a local optimum can be further reduced. Another aspect that may cause algorithm stagnation is that the trial individual learns from only one mutation individual. In Cai et al. (2011), the k-means clustering which acts as a multi-parent crossover was incorporated into the DE to enhance the performance. Parent-centric crossover which utilizes the information in multi-parents was employed to speed up the convergence of DE by Pant et al. (2008).

In this study, the QN method is embedded into the DE algorithm to boost the ability of local search, in other words, several most competitive individuals are locally searched for some generations by the QN method. The strategy of not only one individual for the local search is introduced to decrease the risk of stagnating to a local optimum. These proposals are inspired by the strategies of local search in Qin and Suganthan (2005), Zhao et al. (2008), Zamuda et al. (2009); however, they differentiate the strategies in the reference mainly on the following aspects: the individuals considered for local search are completely searched by QN method in Qin and Suganthan (2005), Zhao et al. (2008), which is more likely to render the algorithm converging to a local optimum; 10 % randomly chosen individuals are performed the SQP local search in Zamuda et al. (2009), which probably results in that a large amount of function evaluations are wasted on those poor-performance individuals. To further reduce the risk of prematurity, we propose a diversity-maintained mutation operator (DMM) to slow down the learning procedure of the population from the searched best individual, or to maintain the diversity of the population as the algorithm proceeds.

This paper is structured as follows: Sect. 2 gives a description about the canonical DE algorithm and its improved variants; Sect. 3 introduces the basic notations and strategies for our proposed algorithm and a detailed depiction of the algorithm. The numerical results of DMGBDE on benchmark problems are presented in Sect. 4. Finally, discussions and some conclusions are addressed in Sects. 5 and 6, respectively.

2 The classical DE

Without loss of generality, the following minimization problem is considered:

$$\min_{\mathbf{x}} f(\mathbf{x}),$$

where $\mathbf{x} = (x_1, \ldots, x_n)$, and $x_j \in [l_j, u_j]$, $j = 1, \ldots, n$. For meta-heuristic algorithms which are population based, assume population $\mathbf{X} \triangleq \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$, where N is the number of individuals. Let $\mathbf{X}^g \triangleq \{\mathbf{X}_1^g, \ldots, \mathbf{X}_N^g\}$ be the population at generation g.

Unlike other meta-heuristic algorithms, the DE algorithm employs the diversity information implied in the population to generate the mutation individual, and the population is gradually guided into the vicinity of the optimum through learning from the searched best individual. The most frequently used version of DE algorithm is $DE/rand/1/bin$, where *rand* denotes choosing random vectors for mutation, 1 denotes employing one difference term for the mutation procedure, and *bin* denotes generating a trial individual by accepting the parameter values from the mutation individual one at a time. This version is detailed described in Algorithm 1 and our proposed algorithm is devised based on this version.

Algorithm 1 DE algorithm (Storn and Price, 1995)

1: Population initialization: $\mathbf{X}_{i,j}^0 = l_j + rand \cdot (u_j - l_j), i = 1, \cdots, N, j = 1, \cdots, n$, *rand* is an uniform random number in $[0, 1]$ for every $\{i, j\}$. $g = 0$.
2: **while** the termination condition is not satisfied **do**
3: **for** $i = 1, \cdots, N$ **do**
4: Randomly choose mutually different indexes $r_1, r_2, r_3 \triangleq r_1^{(i)}, r_2^{(i)}, r_3^{(i)} \in \{1, \cdots, N\}$, randomly choose rn_i from $\{1, \cdots, n\}$.
5: **Mutation:** The mutation individual $\mathbf{V}_i$ is generated as

$$\mathbf{V}_i = \mathbf{X}^g_{r_1^{(i)}} + F \cdot (\mathbf{X}^g_{r_2^{(i)}} - \mathbf{X}^g_{r_3^{(i)}}) \tag{1}$$

where $F \in [0, 1]$ is the mutation probability.
6: **Crossover:** The trial individual $\mathbf{U}_i$ is created as

$$\mathbf{U}_{i,j} = \begin{cases} \mathbf{V}_{i,j}, \text{ if } (rand \le CR) \text{ or } (j = rn_i), \\ \mathbf{X}^g_{i,j}, \text{ otherwise} \end{cases} \tag{2}$$

where CR records the crossover probability and *rand* is a new generated uniform random number in $[0, 1]$ for every $\{i, j\}$.
7: **Selection:** The i-th individual $\mathbf{X}_i^{g+1}$ of the generation $g + 1$ is chosen as

$$\mathbf{X}_i^{g+1} = \begin{cases} \mathbf{U}_i, \text{ if } f(\mathbf{U}_i) \le f(\mathbf{X}_i^g), \\ \mathbf{X}_i^g, \text{ otherwise} \end{cases} \tag{3}$$

8: $g \leftarrow g + 1$.
9: **end for**
10: **end while**

There are other mutation and crossover strategies which compose other versions of DE (Price et al. 2005). In addition to $rand/1$ mutation strategy, $best/1$ and $rand/2$ are also frequently used, which are listed as follows:

1. $best/1$:

$$\mathbf{V}_i = \mathbf{X}_*^g + F \cdot \left(\mathbf{X}^g_{r_2^{(i)}} - \mathbf{X}^g_{r_3^{(i)}}\right) \tag{4}$$

2. $rand/2$:

$$\mathbf{V}_i = \mathbf{X}^g_{r_1^{(i)}} + F \cdot \left(\mathbf{X}^g_{r_2^{(i)}} - \mathbf{X}^g_{r_3^{(i)}}\right) + F \cdot \left(\mathbf{X}^g_{r_4^{(i)}} - \mathbf{X}^g_{r_5^{(i)}}\right) \tag{5}$$

where $\mathbf{X}_*^g$ denotes the best individual in the generation g. Many recently proposed algorithms (Wang et al. 2011; Qin and Suganthan 2005; Mallipeddi et al. 2011) are based on these mutation strategies, which either adjust these strategies in self-adaptive manner or organizes these strategies in a more efficient form. For the crossover operator, there is another strategy which is frequently used in the literature, namely as *exp* crossover, which adopts some consecutive components from the mutation individual with a predetermined probability.

In Algorithm 1, the influences of parameters F and CR on the performance of the algorithm are often concerned. The impact of several mutation and crossover strategies on the expected mean, variance and trial individual generation was theoretically studied in Zaharie (2008, 2009). It was implied that the canonical DE is sensitive to the control parameter values such as scaling factor F and crossover probability CR or sometimes to the strategies generating trial individuals (Gämperle et al. 2002). Thinking of this aspect, many adopted self-adaptive parameters or strategies to improve the performance of DE. The competitive parameters F and CR were retained for the next generation accompanying with the better individual (Jia et al. 2011). In Das and Konar (2009), the crossover probability CR decreased linearly from CR_{min} to CR_{max} with the number of generation. A self-adaptive DE was proposed (Qin et al. 2009) by adjusting the trial vector generation strategies and the associated control parameters values adaptively from the information of previous experiences.

From another aspect, it is reported that local search ability in canonical DE can be improved to enhance the performance (Yang et al. 2008; Neri and Tirronen 2009). Considering this aspect, many also resorted to promoting the searching ability by modifying the classical searching strategies or adopting different mutation and crossover operators, such as, hill climbing heuristic crossover by Noman and Iba (2008), opposition-based DE by Rahnamayan et al. (2008), Wang et al. (2011), 2-opt-based DE by Chiang et al. (2010), fuzzy DE by Liu and Lampinen (2005), neighborhood-based DE by Das et al. (2009) and some other new operators by Dorronsoro and Bouvry (2011), Fan and Lampinen (2003). The local region of the best individual was refined searched for several times until a better solution was obtained (Jia et al. 2011). Some local search strategies were also adopted by Das and Konar

(2009), Mandal et al. (2011) to speed up the convergence. A survey referring to embedding local search into DE was given by Das and Suganthan (2011).

As illustrated in Sect. 1, gradient-based algorithms are frequently incorporated into DE algorithm to enhance the local search ability; however, how to unite these algorithms in a more efficient manner and how to maintain the diversity of the population as the local searching proceeds need to be further studied.

3 The proposed DMGBDE

3.1 Some preliminary elements for the proposed algorithm

3.1.1 Approximate gradient

The approximate gradient $\mathbf{g_x} = (g_1, \ldots, g_n)$ can be calculated referring to Conn et al. (2000) in Section 8.4.3 as:

$$g_j = \frac{f(\mathbf{x} + \Delta x_j \cdot \mathbf{e_j}) - f(\mathbf{x})}{\Delta x_j}, \quad j = 1, \ldots, n \tag{6}$$

where $\mathbf{e_j}$ is a vector satisfying $\mathbf{e_j}_i = 1$ for $i = j$ and $\mathbf{e_j}_i = 0$ for $i \neq j, \Delta x_j$ is a step size which is chosen as follows:

$$\Delta x_j = sign(x_j) \cdot \sqrt{\epsilon} \cdot \max\{|x_j|, s_j\}, \quad j = 1, \ldots, n \tag{7}$$

where $sign(x_j)$ returns the sign of x_j, ϵ is the machine precision, s_j is a scaling number reflects the magnitude of the problem's variables. The forward difference is adopted instead of centered difference to cut down the number of function evaluations. This difference procedure has been embedded into the function *finitedifferences* in the known MATLAB software.

3.1.2 Gradient-based algorithms

A variety of gradient-based algorithms (Nocedal and Wright 1999) are proposed in the last few decades, which are used to find a local optimum around the given initial solution. For local search of the proposed algorithm, the QN method is employed which is presented in Algorithm 2. The QN method is suggested because it utilizes only first-order derivative for optimization, which saves the number of function evaluations in our proposed algorithm, while at the same time, it is proven that it can converge to a local optimum fast.

The QN algorithm is also embedded into the MATLAB where the approximate gradient of objective function is obtained through the function *finitedifferences*, which greatly facilitates the implementations of the proposed algorithm. The MATLAB subfunction for QN local search is presented in Algorithm 3. Different from other literature such as Qin and Suganthan (2005) where gradient-based algorithms are also employed, in this work the gradient-based algorithms are employed to decrease the objective function for only several generations, which reduces the possibility of obtaining a local optimum.

Algorithm 2 BFGS scheme of QN algorithm (Nocedal and Wright, 1999)

1: Given starting point $\mathbf{x}_0$, convergence tolerances $TolFun$, $TolX$, inverse Hessian approximation H_0. $k \leftarrow 0$, $\max|\mathbf{s}_0| \leftarrow \inf$.
2: **while** $|Fun(\mathbf{x}_k) - Fun(\mathbf{x}_{k+1})| > TolFun$ and $\max|\mathbf{s}_k| > TolX$ **do**
3: Compute the search direction $\mathbf{d}_k$ as follows
4: $\mathbf{d}_k = -\mathbf{H}_k \mathbf{g}_{\mathbf{x}_k}$, $\mathbf{g}_{\mathbf{x}_k}$ is calculated as in (6).
5: Find α_k satisfying the following two conditions $Fun(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq Fun(\mathbf{x}_k) + c_1 \alpha_k \mathbf{g}_{\mathbf{x}_k}^T \mathbf{d}_k$, $\mathbf{g}_{\mathbf{x}_k + \alpha_k \mathbf{d}_k}^T \mathbf{d}_k \geq c_2 \mathbf{g}_{\mathbf{x}_k}^T \mathbf{d}_k$, where $0 < c_1 < c_2 < 1$.
6: Define $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$, $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k = \mathbf{g}_{\mathbf{x}_{k+1}} - \mathbf{g}_{\mathbf{x}_k}$. Obtain $\mathbf{H}_{k+1}$ as follows
7: $\mathbf{H}_{k+1} = (\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}) \mathbf{H}_k (\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}$.
8: $k \leftarrow k + 1$.
9: **end while**

Algorithm 3 MATLAB code for QN local search: subalgorithm for Algorithm 5

1: function $[Te, Tef] = QNLS(Fun, \mathbf{x}, Option, NumIter)$
2: $Option.MaxIter = NumIter$;
3: $[Te, Tef] = fminunc(Fun, \mathbf{x}, Option)$;

3.1.3 A new mutation operator

Diversity-maintained mutation (DMM) is introduced in the proposed algorithm to slow down the learning speed from the searched best individual. In other words, we modify the form of generating trial individual $\mathbf{U}_i$, where $\mathbf{U}_i$ learns from not only one mutation individual. For every index j, mutually different random variables r_1, r_2, r_3 (denoted as $r_1^{(j)}, r_2^{(j)}, r_3^{(j)}$) are renewed, then the jth component of the mutation individual is generated as in Eq. (8).

$$\mathbf{V}_{i,j}^{(j)} = \mathbf{X}_{r_1^{(j)},j}^{g} + F \cdot \left(\mathbf{X}_{r_2^{(j)},j}^{g} - \mathbf{X}_{r_3^{(j)},j}^{g} \right) \tag{8}$$

The generated mutation individual $\mathbf{V}_i = (\mathbf{V}_{i,1}^{(1)}, \mathbf{V}_{i,2}^{(2)}, \ldots, \mathbf{V}_{i,n}^{(n)})$ can be viewed to be produced by utilizing the information from n mutation individuals $\mathbf{V}_i^{(1)}, \ldots, \mathbf{V}_i^{(n)}$. The DMM operator is a little similar to the coevolutionary strategy (Wiegand 2004), which also adopts components from multi-individuals to generate a trial individual. This

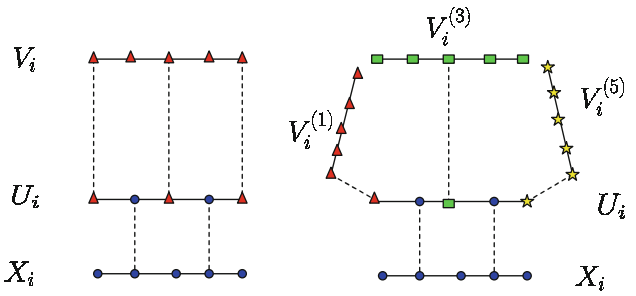

Fig. 1 *Left* generates a trial individual with the DE mutation. *Right* is the generation procedure with the DMM, where $V_i^{(1)}$, $V_i^{(3)}$, $V_i^{(5)}$ are generated mutation individuals for producing 1, 3 and 5 components of the trial individual

DMM operator and the classical DE mutation for generating a trial individual are depicted in Fig. 1. The MATLAB pseudo-code of generating a trial individual where DMM operator is employed with a probability and the procedure of renewing the population are presented in Algorithm 4.

Algorithm 4 Pseudo-code for generating trial individual: sub-algorithm for Algorithm 5

1: function $[\mathbf{X}, \mathbf{Xf}, \mathbf{VF}, \mathbf{VCR}] = TrialInd(\mathbf{Xf}, \mathbf{VF}, \mathbf{VCR}, co)$
2: Generate random numbers $rand_1, rand_2$ in (0,1).

$$\begin{cases} F = 0.1 + 0.8 \cdot rand_1, CR = rand_1 & \text{if } rand_2 \le 0.3, \\ F = \mathbf{VF}_{i,co}, CR = \mathbf{VCR}_{i,co} & \text{otherwise.} \end{cases}$$

3: Generate mutation individual with equation (1) if $co = 1$, or with equation (8) if $co = 2$.
4: Perform crossover operation with equation (2) and generate trial individual $\mathbf{Te}$.
5: **if** $Fun(\mathbf{Te}) < Xf_i$ **then**
6: $\mathbf{X}_i \leftarrow \mathbf{Te}, Xf_i \leftarrow Fun(\mathbf{Te})$,
7: $\mathbf{VF}_{i,co} \leftarrow F, \mathbf{VCR}_{i,co} \leftarrow CR$.
8: **end if**

To illustrate the difference between the mutation operators described in (1) and (8), we present a brief comparison of these two operators in the following. Denote the searched best individual at generation g is $\mathbf{X}_*^g \triangleq \mathbf{X}_{r^g}$ which is assumed to be unique. Because $r_1^{(i)}$ is random number from $\{1, \ldots, N\}$, then

$$p\left\{r_1^{(i)} = r^g\right\} = \frac{1}{N}. \tag{9}$$

A mutation individual $\mathbf{U}_i$ is said to be in the local region of the searched best individual $\mathbf{X}_{r^g}$ if $r_1^{(i)} = r^g$. For the mutation operator in (1), Eq. (9) implies that on average, one of the N mutation individuals falls into the local region of the searched best individual if $F \cdot (\mathbf{X}^g_{r_2^{(i)}} - \mathbf{X}^g_{r_3^{(i)}})$ is a small vector, then the corresponding trial individual accepts approximate $\mathrm{CR} \cdot n$ components from this mutation individual. For the DMM operator described in (8), every trial individual accepts approximate $\frac{\mathrm{CR} \cdot n}{N}$ components from the mutation individuals which fall in the local region of the searched best individual. Moreover, the accepted components for the DMM operator come from different mutation individuals. Thus, the DMM operator learns slowly from the searched best individual, which may facilitate reducing the risk of stagnating to a local optimum.

3.1.4 Adaptive strategy for parameters F and CR

In the proposed algorithm, an adaptive strategy similar to that in Jia et al. (2011), Brest et al. (2006) is employed to accelerate the convergence of algorithm as follows: F and CR of each individual are randomly initialized in intervals [0.1, 0.9] and [0, 1], respectively. Then new uniformly random values of F in [0.1, 0.9] and CR in [0, 1] are generated with probability 0.3 and the parameter values correspond to the better individual are retained to the next generation.

3.1.5 Handling of bounds violation

In our proposed DMGBDE, the following constraint handling is adopted

$$\mathbf{X}_{i,j}^g = \begin{cases} \min\{2 \cdot l_j - \mathbf{X}_{i,j}^g, u_j\} & \text{if } \mathbf{X}_{i,j}^g < l_j, \\ \max\{2 \cdot u_j - \mathbf{X}_{i,j}^g, l_j\} & \text{if } \mathbf{X}_{i,j}^g > u_j, \\ \mathbf{X}_{i,j}^g & \text{otherwise.} \end{cases} \tag{10}$$

3.2 The proposed DMGBDE based on QN algorithm

The overall algorithm which embeds with QN local search algorithm in Algorithm 3 is presented in Algorithm 5. In Algorithm 5, FES is the number of function evaluations which is set to be a global variable, when the considered function is revisited, FES is automatically incremented to FES + 1. *rand* is a MATLAB function which means getting random numbers in (0,1) with uniform distribution. $\lfloor A \rfloor$ means rounding number A to the nearest integer less than or equal to A. For setting the optimization parameters for function *fminunc* in step 4, *LargeScale* $\leftarrow$ *off* indicates adopting the forward difference to approximate the gradient; *HessUpdate* $\leftarrow$ *bfgs* means applying the *bfgs* iterative scheme to update the inverse Hessian matrix; *TolFun*, *TolX* are the convergence tolerances which are the same as those in step 2 of Algorithm 2. A threshold n is added into the step 8 to guarantee that a number of local search are performed from the beginning of running when FES is small.

In order to balance the convergence speed and the diversity of the population, the classical mutation is added

into the proposed algorithm, and pDEc determines the probability of adopting classical DE mutation. In addition to the searched best individual, several competitive individuals are also chosen for local search when the searched best individual is not renewed for several generations.

Please note that the renewed individual is used to generate the next trial individual whenever it is generated in the proposed algorithm, in other words, only one matrix is used to store all the individuals as the algorithm proceeds.

Algorithm 5 The proposed DMGBDE algorithm

Initialize the population $\mathbf{X}$ and compute the corresponding fitness $\mathbf{Xf}$. $MaxFES = n \cdot 10^4$.
Initialize $FES \leftarrow 0$, $FlagRenew \leftarrow 1$, $nNIMPROVE \leftarrow 0$ and objective function Fun.
$\mathbf{VF}, \mathbf{VCR}$ are initialized as $0.1 + 0.8 \cdot rand(N, 2), rand(N, 2)$, respectively.
$Option = optimset('LargeScale','off','HessUpdate','bfgs','TolFun', 1e-30,'TolX', 1e-30)$;
while $FES \leq MaxFES$ **do**
  % **Step 1: Perform local search around the searched best individual.**
  if $FlagRenew = 1$ **then**
    $[\mathbf{Te}, Tef] = QNLS(Fun, \mathbf{X}_i, Option, n + \lfloor \frac{FES}{MaxFES} MaxITER \rfloor)$;
    $\mathbf{X}_* \leftarrow \mathbf{Te}, Xf_* \leftarrow Tef$. Renew FES.
    $FlagRenew = 0$.
  end if
  % **Step 2: Generate trial individuals and renew the population.**
  for $i = 1, \cdots, N$ **do**
    if $rand \leq pDEc$ **then**
      $[\mathbf{X}, \mathbf{Xf}, \mathbf{VF}, \mathbf{VCR}] = TrialInd(\mathbf{Xf}, \mathbf{VF}, \mathbf{VCR}, 1)$.
    else
      $[\mathbf{X}, \mathbf{Xf}, \mathbf{VF}, \mathbf{VCR}] = TrialInd(\mathbf{Xf}, \mathbf{VF}, \mathbf{VCR}, 2)$.
    end if
    $FES \leftarrow FES + 1$. Set $FlagRenew = 1$ if the best individual has been renewed.
  end for
  % **Step 3: Perform local search around other competitive individuals.**
  if the best has not been renewed **then**
    $nNIMPROVE = nNIMPROVE + 1$.
    if $nNIMPROVE > NNIMPROVE$ **then**
      for $i = 1, \cdots, \min\{\max\{\lfloor 0.1 \cdot n \rfloor, 3\}, 8\}$ **do**
        Choose the $(i+1)$-th best individual $\mathbf{X}_{*(i+1)}$.
        $[\mathbf{Te}, Tef] = QNLS(Fun, \mathbf{X}_{*(i+1)}, Option, n + MaxITER)$;
        $\mathbf{X}_{*(i+1)} \leftarrow \mathbf{Te}, Xf_{*(i+1)} \leftarrow Tef$. Renew FES.
      end for
      $nNIMPROVE = 0, FlagRenew = 1$.
    end if
  else
    $nNIMPROVE = 0$.
  end if
end while

4 Numerical experiments on benchmark problems

4.1 Test problems and criteria for comparison

In this section, we conduct experiments on 28 benchmark functions which are presented in Table 1 in detail, where $\mathbf{x}_i^*$ is the ith component of the best solution, $f_{\min}$ is the fitness of the best solution. These functions are frequently used among the research community, the first 14 functions are chosen from CEC2005 contest problems in Suganthan et al. (2005), the other 14 functions are selected from literature Jia et al. (2011), Noman and Iba (2008), Yao et al. (1999).

Two comparisons are presented to compare the performance of the proposed algorithm DMGBDE with that of other variants without local search or DMM operator and some recently reported algorithms. We perform all the algorithms on a PC with a core processor, operating at 2.8 GHz and with 4 GB of RAM. The overall computations about all the considered algorithms are implemented in the popular scientific program MATLAB, version 2009b.

For the comparisons, the following aspects are mainly considered to assess the performance of the algorithms on the test problems: the average best result (AveFit), the standard deviation (SD) of these results, the success rate (SR) of the trials, the mean overall runtime (RunTime), and also the diversity rate (DivRate) which is the ratio of the population diversity before and after the executing the considered algorithm. The diversity of the population $\mathbf{X}$ is simply expressed quantitatively as $\frac{1}{n \cdot N} \sum_{j=1}^{n} \sum_{i=1}^{N} (\frac{\mathbf{X}_{ij} - \bar{\mathbf{X}}_j}{u_j - l_j})^2$, where $\bar{\mathbf{X}}_j = \frac{1}{N} \sum_{i=1}^{N} \mathbf{X}_{i,j}$. The defined diversity depicts the algorithm's ability to continue searching. For quantifying DivRate, a trial is said to be successful if fitness 10^{-4} and 10^{-2} are obtained for f_1–f_5 and f_6–f_{14}, respectively, or fitness 10^{-2} is obtained for noisy function f_{27}, and 10^{-8} for functions f_{15}–f_{28} except f_{27} (Zhang and Sanderson 2009).

Table 1 Descriptions of benchmark functions

Functions and the corresponding descriptions
f_1: Shifted sphere function. Unimodal functions. $f_{\min} = 0$
f_2: Shifted Schwefel's problem 1.2. Unimodal functions. $f_{\min} = 0$
f_3: Shifted rotated high conditioned elliptic function. Unimodal functions. $f_{\min} = 0$
f_4: Shifted Schwefel's problem 1.2 with noise in fitness. Unimodal functions. $f_{\min} = 0$
f_5: Schwefel's problem 2.6 with global optimum on bounds. Unimodal functions. $f_{\min} = 0$
f_6: Shifted Rosenbrock's function. Basic multimodal functions. $f_{\min} = 0$
f_7: Shifted rotated Griewank's function. Basic multimodal functions. $f_{\min} = 0$
f_8: Shifted rotated Ackley's function with global optimum on bounds. Basic multimodal functions. $f_{\min} = 0$
f_9: Shifted Rastrigin's function. Basic multimodal functions. $f_{\min} = 0$
f_{10}: Shifted rotated Rastrigin's function. Basic multimodal functions. $f_{\min} = 0$
f_{11}: Shifted rotated Weierstrass function. Basic multimodal functions. $f_{\min} = 0$
f_{12}: Shifted rotated expanded Scaffer's F6. Basic multimodal functions. $f_{\min} = 0$
f_{13}: Expanded extended Griewank's plus Rosenbrock's function (F8F2). Expanded multimodal functions. $f_{\min} = 0$
f_{14}: Shifted Schwefel's problem 1.2. Expanded multimodal functions. $f_{\min} = 0$
$f_{15}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2,\ x_i \in [-100,\ 100],\ x_i^* = 0, f_{\min} = 0$
$f_{16}(\mathbf{x}) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2),\ x_i \in [-100,\ 100],\ x_i^* = 1, f_{\min} = 0$
$f_{17}(\mathbf{x}) = 20 - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + \exp(1),\ x_i \in [-32, 32],\ x_i^* = 0,\ f_{\min} = 0.$
$f_{18}(\mathbf{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4{,}000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1, x_i \in [-600, 600], x_i^* = 0,\ f_{\min} = 0.$
$f_{19}(\mathbf{x}) = \sum_{i=1}^{n}(\sum_{j=1}^{i})^2,\ x_i \in [-100,\ 100],\ x_i^* = 0, f_{\min} = 0.$
$f_{20}(\mathbf{x}) = \sum_{j=1}^{n}\sum_{i=1}^{n}(\frac{y_{i,j}^2}{4{,}000} - \cos(y_{i,j}) + 1), where y_{i,j} = 100(x_j - x_i^2)^2 + (1 - x_i)^2, x_i \in [-100, 100], x_i^* = 1, f_{\min} = 0.$
$f_{21}(\mathbf{x}) = -\cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2} + 1, \quad x_i \in [-100, 100], \quad x_i^* = 0, \quad f_{\min} = 0.$
$f_{22}(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi\ x_i)),\ x_i \in [-5,\ 5],\ x_i^* = 0, f_{\min} = 0.$
$f_{23}(\mathbf{x}) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$ $x_i \in [-50, 50], x_i^* = -1, f_{\min} = 0$
$f_{24}(\mathbf{x}) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4),$ $x_i \in [-50,\ 50],\ x_i^* = 1, f_{\min} = 0.$
$f_{25}(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert, x_i \in [-10, 10], x_i^* = 0, f_{\min} = 0.$
$f_{26}(\mathbf{x}) = \sum_{i=1}^{n}(\lfloor x_i + 0.5\rfloor)^2, x_i \in [-100, 100], x_i^* \in [-\frac{1}{2}, \frac{1}{2}], f_{\min} = 0.$
$f_{27}(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + rand[0,\ 1),\ x_i \in [-1.28,\ 1.28],\ x_i^* = 0, f_{\min} = 0.$
$f_{28}(\mathbf{x}) = \max\lvert x_i\rvert,\ x_i \in [-100,\ 100],\ x_i^* = 0, f_{\min} = 0.$

4.2 The compared algorithms and parameter settings

To study the influences of all operators or strategies on our proposed DMGBDE, we conduct a comparison on the benchmark functions among seven different algorithms, which are DE/rand/1/bin (DE), mere QN local search of independent N individuals (GB), DMGBDE without QN local search and DMM operator (SPDE, namely as self-adaptive parameter DE), DMGBDE without QN local search (DMDE), DMGBDE without classical mutation (DMGBDE1), DMGBDE without DMM operator (GBDE), and DMGBDE. The comparison is presented in Table 3. Variants SPDE and DMDE differ in only the aspect of DMM operator, so is the variants of GBDE and DMGBDE1. Variants SPDE and GBDE are different in only the aspect of QN local search, so is the variants of DMDE and DMGBDE1. For all comparisons, the number of maximal FES *MaxFES* is set to be $10^4 n$, where n is the dimension of test function.

To further assess the performance of the DMGBDE, five recently reported algorithms jDE, SaDE, DECLS, EPSDE and CoDE in the literature Brest et al. (2006), Qin et al. (2009), Jia et al. (2011), Mallipeddi et al. (2011) and Wang et al. (2011), respectively. For this comparison, we conduct two suites of experiments with dimension size n be 30 and 50, which are presented in Tables 6 and 7, respectively.

Table 2 Parameter selection and some comments about the algorithms for comparison

Algorithm	Parameter selection and comments
DE	$F = 0.9$, CR = 0.9, $N = n$ (Ronkkonen et al. 2005; Noman and Iba 2008)
jDE	$\{n = 30, N = 100\}$. $\tau_1 = \tau_2 = 0.1$. F and CR are self-adaptive adjusted (Brest et al. 2006) $\{n = 50, N = 150\}$
SaDE	$\{n = 30, N = 50\}$ The number of strategies K is set to be 4. F and CR are self-adaptive. Adjusted based on the previous information (Qin et al. 2009) $\{n = 50, N = 100\}$
EPSDE	$\{n = 30, N = 50\}$ Parameters F and CR choose values from [0.4, 0.9] and [0.1, 0.9], respectively (Jia et al. 2011) $\{n = 50, N = 100\}$
DECLS	$N = n$, $L = n/5$, $m = 1{,}500$. Local search around the searched best individual (Mallipeddi et al. 2011)
CoDE	$N = n$, non-adaptive variant is employed which is reported to be superior to the adaptive variant (Wang et al. 2011b)
DMGBDE	pDEc = 0.2, $N = n$, NNIMPROVE = $30n$, MaxITER = $2n$. About 0.1 individuals are chosen for local search conditionally

Numerical experiments in Noman and Iba (2008) show that large population size is not beneficial to the performance for DE when the maximal FES is fixed to be $10^4 n$, therefore, the population size N of DE is set to be n for the comparison. The parameter settings of all algorithms stay the same as the corresponding literature for 30 dimensional (30D) functions. When the dimension n of the functions changes from 30 to 50, the population size N is increased such that the ratio of the population size and the dimension of the problem is approximate the same as that for 30D problems. All parameter settings and some comments about the algorithms are listed in Table 2.

Notice that the population sizes of the algorithms for comparison are different (such as $N = 100$ for jDE, $N = 50$ for SaDE in Table 2), which may make the comparison unfair because it had been reported that the performance of DE algorithm is sensitive to the selection of the population size (Gämperle et al. 2002) and that small population size probably facilitates the exploitation ability of DE algorithm (Noman and Iba 2008). To check the fairness of the parameter settings in Table 2 for the comparisons, results are first obtained in Table 5 by resetting the population size N to be 30 for all the considered algorithms for 30D problems, then a preliminary comparison of these results (in Table 5) and the results (in Table 6) obtained by the proposed parameter settings of Table 2 is conducted in Sect. 4.3.

It is worthy to illustrate the selection of all the parameter values in the proposed algorithm. The settings of some parameters (0.1, 0.8 and 0.3 in the second step of the Algorithm 4) are similar to the reference (Brest et al. 2006; Jia et al. 2011) with a slight modification, some parameters are assigned their values after several tunings ($pDEc = 0.2$, $NNIMPROVE = 30n$, $MaxITER = 2n$ in Table 2), and the values of some parameters are determined without any tuning ($N = n, 0.1 \cdot n$ in Table 2).

It is worthy to note that a proportional number of function evaluations are used to obtain the approximate gradient through finite difference, thus, our proposed algorithm DMGBDE may occupy less computation time than other algorithms when given the same number of function evaluations. A comparison about the average runtime of the considered algorithms on 28 test problems are presented in Table 8.

To quantitatively compare the performance of the considered algorithms on 28 test functions, we employ Wilcoxon rank sum test method [nonparametric method (Gibbons and Chakraborti 2003)] with significance level 0.05. The results of the comparison are presented in Table 9. The results in Table 9 can not reflect the overall significance of the difference between the control algorithm (DMGBDE) and the other algorithms, thus, further significance comparison of the algorithms are conducted and the results are presented in Table 10. Two statistical tests are employed, the first is Friedman's test which is to obtain the rankings of all the algorithms and the significance of the global difference; the second is Holm's test which is to detect the concrete differences between the control algorithm and the other algorithms (García et al. 2009). For this comparison, the validation of the required conditions for these statistical methods is skipped over and the average fitness values obtained by the corresponding algorithms are used.

The average best fitness w.r.t. the average number of function evaluations are depicted in Figs. 2 and 3. Please note that for our algorithm which is with local search algorithm, the numbers of FES in each generation may differ greatly for every run (The number of function evaluations is actually between $10^4 n$ and $10^4 n + 2(n + MaxITER)n$, however, the results only before $10^4 n$ function evaluations are used.), then all the best fitness and the *FES* are averaged w.r.t. the number of generations. We are concerned with how the algorithm evolves to the best solution, thus, it is reasonable that the performance of the proposed DMGBDE in later generations are only considered.

Table 3 Results of all variants on problems in 30 dimensions

Fun.	DE AveFit ± SD (RS, DivRate)	GB AveFit ± SD (RS, DivRate)	SPDE AveFit ± SD (RS, DivRate)	DMDE AveFit ± SD (RS, DivRate)	DMGBDE1 AveFit ± SD (RS, DivRate)	GBDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_1	4.63E−18 ± 6.85E−18 (**100**, 5.42E−23)	4.98E−12 ± 1.33E−15 (**100**, 7.03E−25)	8.60E−29 ± 1.72E−28 (**100**, 2.25E−31)	**0.00E+00 ± 0.00E+00** (**100**, 5.25E−31)	1.03E−29 ± 4.51E−29 (**100**, 2.32E−31)	7.19E−29 ± 1.19E−28 (**100**, 2.27E−31)	**0.00E+00 ± 0.00E+00** (**100**, 2.34E−31)
f_2	2.54E−02 ± 3.07E−02 (0, 2.35E−07)	4.23E−09 ± 2.86E−10 (**100**, 7.93E−16)	7.02E−13 ± 1.62E−12 (**100**, 3.25E−19)	4.13E−02 ± 3.60E−02 (0, 2.57E−07)	4.55E−09 ± 2.56E−10 (**100**, 5.09E−03)	**3.21E−13 ± 3.54E−13** (**100**, 1.97E−19)	4.69E−11 ± 3.05E−11 (**100**, 9.12E−17)
f_3	6.67E+05 ± 3.09E+05 (0, 1.50E−03)	2.70E+02 ± 2.75E+02 (0, 6.46E−02)	8.97E+04 ± 3.99E+04 (0, 3.82E−06)	5.04E+05 ± 2.91E+05 (0, 7.77E−04)	1.46E+03 ± 1.49E+03 (0, 4.88E−01)	**5.98E−02 ± 2.52E−01** (0, 1.18E−04)	1.22E+01 ± 1.82E+01 (0, 1.02E−04)
f_4	**1.09E+01 ± 8.96E+00** (0, 8.04E−05)	2.34E+05 ± 9.51E+04 (0, 1.00E+00)	1.96E+01 ± 3.03E+01 (0, 9.88E−06)	5.24E+01 ± 5.64E+01 (0, 2.11E−04)	6.49E+03 ± 1.91E+03 (0, 1.84E−01)	1.71E+02 ± 2.12E+02 (0, 8.14E−05)	2.70E+02 ± 2.16E+02 (0, 4.20E−04)
f_5	**1.09E+03 ± 4.31E+02** (0, 1.63E−04)	1.69E+03 ± 4.38E+02 (0, 2.39E−01)	2.29E+03 ± 1.02E+03 (0, 1.36E−05)	1.43E+03 ± 5.03E+02 (0, 4.23E−01)	2.84E+03 ± 5.83E+02 (0, 3.09E−01)	1.85E+03 ± 8.35E+02 (0, 3.26E−05)	1.83E+03 ± 7.44E+02 (0, 1.37E−05)
f_6	2.03E+01 ± 2.30E+01 (0, 5.37E−07)	5.13E−08 ± 9.79E−09 (**100**, 1.46E−02)	2.86E+01 ± 2.67E+01 (5, 3.70E−10)	4.29E+01 ± 2.64E+01 (0, 1.24E−07)	5.70E−08 ± 1.03E−08 (**100**, 2.17E−03)	5.98E−01 ± 1.46E+00 (85, 3.17E−18)	**2.81E−10 ± 1.15E−09** (**100**, 1.95E−06)
f_7	5.30E−03 ± 5.67E−03 (70, 3.69E−14)	**2.54E−10 ± 3.89E−11** (**100**, 1.67E−05)	2.30E−02 ± 1.71E−02 (30, 4.35E−22)	1.97E−02 ± 1.19E−02 (50, 3.73E−18)	3.57E−03 ± 6.16E−03 (95, 5.59E−06)	9.48E−03 ± 1.14E−02 (60, 4.83E−21)	1.08E−02 ± 1.17E−02 (60, 2.18E−20)
f_8	2.10E+01 ± 6.06E−02 (0, 9.54E−01)	**2.00E+01 ± 1.18E−08** (0, 1.00E+00)	2.10E+01 ± 5.28E−02 (0, 8.84E−01)	2.10E+01 ± 6.13E−02 (0, 9.52E−01)	**2.00E+01 ± 4.09E−08** (0, 8.93E−01)	**2.00E+01 ± 4.72E−08** (0, 9.00E−01)	**2.00E+01 ± 4.53E−08** (0, 8.78E−01)
f_9	1.99E+01 ± 5.41E+00 (0, 4.06E−11)	2.83E+02 ± 3.98E+01 (0, 9.96E−01)	6.47E−01 ± 8.71E−01 (55, 5.12E−20)	**0.00E+00 ± 0.00E+00** (**100**, 6.88E−20)	4.97E−02 ± 2.22E−01 (95, 5.53E−20)	4.48E−01 ± 7.55E−01 (70, 5.63E−20)	1.99E−01 ± 5.21E−01 (85, 5.67E−20)
f_{10}	1.41E+02 ± 8.69E+01 (0, 5.62E−02)	5.16E+02 ± 8.65E+01 (0, 9.55E−01)	4.02E+01 ± 8.34E+00 (0, 4.38E−03)	5.42E+01 ± 8.45E+00 (0, 1.09E−01)	3.85E+01 ± 5.08E+00 (0, 1.14E−01)	4.24E+01 ± 1.06E+01 (0, 6.15E−04)	**3.75E+01 ± 7.06E+00** (0, 1.14E−01)
f_{11}	3.62E+01 ± 9.49E+00 (0, 7.97E−01)	5.10E+01 ± 2.08E+00 (0, 1.00E+00)	**1.91E+01 ± 4.81E+00** (0, 4.30E−02)	2.78E+01 ± 1.97E+00 (0, 9.55E−01)	2.80E+01 ± 2.31E+00 (0, 9.20E−01)	2.20E+01 ± 6.20E+00 (0, 3.89E−01)	2.76E+01 ± 1.69E+00 (0, 9.02E−01)
f_{12}	5.30E+03 ± 5.19E+03 (0, 1.07E−05)	**9.11E−09 ± 3.56E−09** (**100**, 1.81E+00)	6.83E+03 ± 9.51E+03 (0, 8.70E−08)	1.79E+04 ± 5.64E+03 (0, 4.69E−01)	7.90E+01 ± 2.62E+02 (55, 5.30E−01)	1.31E+03 ± 1.54E+03 (30, 1.91E−02)	2.18E+02 ± 5.19E+02 (40, 4.66E−01)
f_{13}	3.49E+00 ± 7.25E−01 (0, 1.56E−07)	2.94E+01 ± 1.19E+01 (0, 4.16E−01)	1.27E+00 ± 9.46E−02 (0, 3.67E−02)	1.50E+00 ± 1.10E−01 (0, 4.66E−02)	8.76E−01 ± 1.43E−01 (0, 4.33E−02)	**8.68E−01 ± 9.64E−02** (0, 4.23E−02)	8.84E−01 ± 1.36E−01 (0, 4.66E−02)
f_{14}	1.34E+01 ± 1.55E−01 (0, 8.91E−01)	1.40E+01 ± 2.01E−01 (0, 1.00E+00)	1.27E+01 ± 2.90E−01 (0, 7.49E−01)	1.29E+01 ± 1.84E−01 (0, 8.59E−01)	1.24E+01 ± 2.75E−01 (0, 8.20E−01)	1.23E+01 ± 4.80E−01 (0, 7.45E−01)	**8.98E−01 ± 1.56E−01** (0, 1.67E−05)
f_{15}	1.92E−17 ± 3.71E−17 (**100**, 1.96E−22)	1.60E−15 ± 4.82E−17 (**100**, 2.08E−23)	4.31E−32 ± 6.60E−32 (**100**, 3.95E−35)	**0.00E+00 ± 0.00E+00** (**100**, 8.25E−35)	6.16E−34 ± 2.76E−33 (**100**, 3.75E−35)	3.14E−32 ± 4.89E−32 (**100**, 3.93E−35)	2.03E−32 ± 6.06E−32 (**100**, 3.74E−35)
f_{16}	1.25E+01 ± 1.35E+01 (0, 1.66E−07)	4.08E−11 ± 1.46E−11 (**100**, 1.99E−03)	1.64E+01 ± 2.02E+01 (0, 1.91E−10)	3.38E+01 ± 2.34E+01 (0, 3.06E−08)	3.99E−01 ± 1.23E+00 (90, 1.00E−03)	1.99E−01 ± 8.91E−01 (95, 2.42E−21)	**1.01E−11 ± 2.07E−11** (**100**, 1.84E−19)
f_{17}	1.17E−09 ± 2.72E−09 (95, 1.63E−21)	1.90E+01 ± 1.86E−01 (0, 1.00E+00)	4.66E−02 ± 2.08E−01 (95, 1.86E−21)	**2.66E−15 ± 0.00E+00** (**100**, 4.34E−33)	**2.66E−15 ± 0.00E+00** (**100**, 3.89E−33)	1.62E−01 ± 3.98E−01 (85, 5.66E−21)	**2.66E−15 ± 0.00E+00** (**100**, 3.85E−33)
f_{18}	3.82E−03 ± 6.00E−03 (65, 2.82E−21)	1.22E−16 ± 1.13E−16 (**100**, 1.38E−08)	4.30E−03 ± 8.98E−03 (75, 2.20E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.94E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.41E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.26E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.37E−21)

Table 3 continued

Fun.	DE AveFit ± SD (RS, DivRate)	GB AveFit ± SD (RS, DivRate)	SPDE AveFit ± SD (RS, DivRate)	DMDE AveFit ± SD (RS, DivRate)	DMGBDE1 AveFit ± SD (RS, DivRate)	GBDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_{19}	1.95E−02 ± 2.61E−02 (0, 1.77E−07)	1.56E−16 ± 9.35E−17 (**100**, 1.34E−19)	1.13E−13 ± 1.79E−13 (**100**, 6.61E−20)	3.19E−02 ± 2.14E−02 (0, 1.98E−07)	5.18E−16 ± 5.87E−16 (**100**, 2.99E−02)	**3.71E−22 ± 5.54E−22** (**100**, 2.60E−28)	8.98E−19 ± 1.40E−18 (**100**, 2.67E−24)
f_{20}	3.09E+02 ± 1.06E+02 (0, 7.15E−15)	5.89E+02 ± 5.57E+01 (0, 5.33E−05)	1.32E+02 ± 1.21E+02 (**20**, 1.66E−16)	**1.32E+01 ± 4.03E+00** (0, 9.38E−07)	1.44E+01 ± 2.13E+01 (0, 1.07E−06)	1.67E+02 ± 1.23E+02 (0, 3.54E−22)	1.73E+01 ± 1.29E+01 (0, 2.57E−07)
f_{21}	2.36E−01 ± 4.66E−02 (0, 5.62E−05)	7.75E+00 ± 1.08E+01 (5, 9.43E−01)	2.40E−01 ± 5.98E−02 (0, 2.52E−05)	1.75E−01 ± 4.44E−02 (0, 3.09E−05)	**1.40E−01 ± 7.54E−02** (15, 3.40E−05)	2.35E−01 ± 9.88E−02 (5, 2.42E−05)	1.45E−01 ± 8.87E−02 (**20**, 2.33E−05)
f_{22}	2.90E+01 ± 7.01E+00 (0, 9.18E−16)	1.71E+02 ± 1.52E+01 (0, 1.00E+00)	3.48E−01 ± 5.84E−01 (70, 1.53E−18)	**0.00E+00 ± 0.00E+00** (**100**, 1.86E−18)	1.49E−01 ± 3.65E−01 (85, 1.67E−18)	1.49E−01 ± 3.65E−01 (85, 1.51E−18)	1.99E−01 ± 4.08E−01 (80, 1.53E−18)
f_{23}	2.59E−02 ± 9.44E−02 (90, 1.10E−19)	4.00E−17 ± 9.15E−18 (**100**, 7.94E−03)	1.60E−32 ± 1.15E−33 (**100**, 3.06E−34)	**1.57E−32 ± 2.81E−48** (**100**, 6.05E−34)	**1.57E−32 ± 2.81E−48** (**100**, 3.08E−34)	5.18E−03 ± 2.32E−02 (95, 9.00E−21)	**1.57E−32 ± 2.81E−48** (**100**, 3.00E−34)
f_{24}	1.80E−01 ± 8.04E−01 (90, 3.10E−19)	3.94E−01 ± 4.78E−01 (10, 5.51E−03)	1.10E−03 ± 3.38E−03 (90, 1.08E−22)	**1.35E−32 ± 2.81E−48** (**100**, 3.31E−34)	1.43E−32 ± 2.41E−33 (**100**, 1.50E−34)	5.49E−04 ± 2.46E−03 (95, 7.39E−23)	**1.35E−32 ± 2.81E−48** (**100**, 1.51E−34)
f_{25}	8.54E−11 ± 9.52E−11 (**100**, 1.23E−24)	1.16E+02 ± 8.45E+00 (0, 9.81E−01)	**1.52E−131 ± 4.58E−131** (**100**, 2.08E−265)	3.42E−65 ± 3.35E−65 (**100**, 2.80E−133)	9.73E−27 ± 7.42E−27 (**100**, 1.38E−56)	3.79E−26 ± 4.97E−26 (**100**, 3.77E−55)	1.60E−26 ± 1.75E−26 (**100**, 5.49E−56)
f_{26}	**0.00E+00 ± 0.00E+00** (**100**, 1.45E−05)	6.58E+04 ± 8.09E+03 (0, 1.00E+00)	**0.00E+00 ± 0.00E+00** (**100**, 1.15E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.53E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.29E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.22E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.22E−05)
f_{27}	1.50E−02 ± 5.47E−03 (15, 7.64E−03)	1.29E+02 ± 2.14E+01 (0, 1.00E+00)	2.73E−03 ± 3.15E−03 (95, 2.04E−03)	2.58E−03 ± 5.35E−04 (**100**, 4.48E−03)	2.10E−03 ± 4.72E−04 (**100**, 3.25E−03)	2.88E−03 ± 1.09E−03 (**100**, 2.39E−03)	**2.04E−03 ± 6.74E−04** (**100**, 3.35E−03)
f_{28}	1.89E+00 ± 1.73E+00 (0, 2.97E−04)	6.89E−02 ± 5.28E−02 (0, 7.88E−04)	1.18E+01 ± 6.30E+00 (0, 7.67E−03)	4.46E−06 ± 1.94E−05 (90, 2.69E−14)	**1.82E−17 ± 1.52E−17** (**100**, 3.57E−38)	2.45E−17 ± 1.44E−17 (**100**, 3.83E−38)	1.98E−17 ± 1.17E−17 (**100**, 3.17E−38)

Table 4 Results of variants GBDE and DMGBDE with *MaxFES* be 600000 for some of 30 dimensional problems

MaxFES	Variant	f_2 AveFit (DivRate)	f_3 AveFit (DivRate)	f_4 AveFit (DivRate)	f_{11} AveFit (DivRate)	f_{19} AveFit (DivRate)	f_{12} AveFit (DivRate)	f_{15} AveFit (DivRate)	f_{25} AveFit (DivRate)	f_{28} AveFit (DivRate)
1.0E+04n	GBDE	**3.21E−13**	**5.98E−02**	**1.71E+02**	**2.20E+01**	**3.71E−22**	1.31E+03	3.14E−32	3.79E−26	2.45E−17
		(1.97E−19)	(1.18E−04)	(8.14E−05)	(3.89E−01)	(2.60E−28)	(1.91E−02)	(3.93E−35)	(3.77E−55)	(3.83E−38)
	DMGBDE	4.69E−11	1.22E+01	2.70E+02	2.76E+01	8.98E−19	**2.18E+02**	**2.03E−32**	**1.60E−26**	**1.98E−17**
		(9.12E−17)	(1.02E−04)	(4.20E−04)	(9.02E−01)	(2.67E−24)	(4.66E−01)	(3.74E−35)	(5.49E−56)	(3.17E−38)
2.0E+04n	GBDE	**2.76E−17**	1.11E−03	5.07E+01	**2.06E+01**	**6.67E−27**	1.25E+03	2.34E−32	8.01E−39	1.51E−17
		(1.81E−23)	(5.68E−13)	(5.66E−06)	(8.54E−02)	(6.61E−33)	(2.53E−02)	(3.84E−35)	(1.61E−80)	(1.57E−38)
	DMGBDE	1.36E−15	**1.08E−03**	**1.02E+01**	2.57E+01	4.47E−24	**1.28E+02**	**3.08E−33**	**1.10E−39**	**9.40E−18**
		(1.62E−21)	(2.72E−11)	(1.38E−05)	(8.68E−01)	(7.69E−30)	(2.24E−01)	(3.76E−35)	(6.37E−82)	(1.30E−38)

4.3 Illustration of the results

It can be seen from Table 3 that the algorithm DMGBDE1 achieves significantly larger DivRate than other algorithms on test functions f_2–f_7, f_{16}, f_{19}, which means significantly more powerful capability to continue searching. DMGBDE1 is superior to GBDE (DMGBDE without DM operator) on multimodal functions such as f_6–f_8, f_{12}, while GBDE is more competitive on unimodal functions such as f_2–f_5. Inspired from the results, we add the classical mutation operator into the DMGBDE1 algorithm with some probability, which composes the algorithm DMGBDE to balance the exploration and exploitation abilities. Revealed by Wilcoxon test, DMGBDE obtains significantly better performance on 8 problems than DMGBDE1, and it is not worse than DMGBDE1 in any functions, in other words, DMGBDE greatly enhances the performance of DMGBDE1.

Statistical analysis indicates that DMGBDE significantly outperforms the variant GB (mere QN search) on 24 functions, which illustrates that mere QN local search is far from the effective solver for the considered problems and the strategy related to DE algorithm is essential to the proposed algorithm.

As far as we know, other algorithms using the gradient-based algorithms as the QN local search are based on other stochastic algorithms and applications (Qin and Suganthan 2005; Zhao et al. 2008) or employ other local search strategies (Zhao et al. 2008; Zamuda et al. 2009). Thus, it is not convicing enough to find out the actual function of the QN local search on the proposed algorithm. As a feasible alternative, a comparison of the variants (SPDE and GBDE, DMDE and DMGBDE1) which are only different on the existence of the QN local search is conducted to clarify the effect of the QN local search on the proposed algorithm. Comparison between the results of SPDE (DMGBDE without the DMM operator and the QN search) and GBDE reveals that GBDE is significantly superior to SPDE on 11 functions and inferior to SPDE on 3 functions, which illustrates that the proposed scheme of QN local search is beneficial to the performance of the algorithm. This statement can be also concluded from the comparison of results of DMDE (DMGBDE1 without the QN search) and DMGBDE1.

To see the effect of the DMM operator on the proposed algorithm, comparison of the performances of SPDE and DMDE (DMDE is DMGBDE without QN search), or GBDE and DMGBDE1 is only needed. The population diversity after executing DMDE algorithm is significantly larger than that after executing SPDE algorithm on all 30D problems except functions f_5, f_{17}, f_{24}, f_{28} by the Wilcoxon test, which verifies the analysis in Sect. 3.1.3 that algorithm with DMM learns slower from the searched best individual than that with classical mutation.

Table 5 Results of the considered algorithms with population size N all set to be 30 on problems in 30 dimensions

Fun.	jDE AveFit ± SD (RS, DivRate)	SaDE AveFit ± SD (RS, DivRate)	EPSDE AveFit ± SD (RS, DivRate)	DECLS AveFit ± SD (RS, DivRate)	CoDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_1	7.57E−30 ± 2.47E−29 (**100**, 2.31E−31)	1.77E−29 ± 4.71E−29 (**100**, 2.31E−31)	5.05E−29 ± 8.19E−29 (**100**, 2.28E−31)	1.02E−28 ± 4.06E−28 (**100**, 2.29E−31)	**0.00E+00 ± 0.00E+00** (**100**, 2.33E−31)	**0.00E+00 ± 0.00E+00** (**100**, 2.34E−31)
f_2	1.59E−11 ± 5.34E−11 (**100**, 9.30E−18)	4.86E−04 ± 1.27E−03 (75, 8.93E−12)	**2.38E−21 ± 1.07E−20** (**100**, 1.24E−27)	1.71E−03 ± 3.22E−03 (35, 1.38E−07)	6.72E−15 ± 2.07E−14 (**100**, 1.98E−20)	4.69E−11 ± 3.05E−11 (**100**, 9.12E−17)
f_3	1.16E+05 ± 5.91E+04 (0, 1.72E−04)	5.58E+05 ± 2.20E+05 (0, 1.43E−07)	1.77E+06 ± 4.50E+06 (0, 4.77E−02)	1.61E+06 ± 1.05E+06 (0, 1.27E−01)	1.10E+05 ± 4.48E+04 (0, 1.08E−05)	**1.22E+01 ± 1.82E+01** (0, 1.02E−04)
f_4	3.85E+01 ± 8.91E+01 (0, 1.48E−05)	1.01E+03 ± 9.03E+02 (0, 7.74E−05)	4.51E+02 ± 7.16E+02 (0, 1.02E−05)	9.51E+00 ± 1.50E+01 (0, 2.90E−04)	**9.17E−03 ± 2.51E−02** (**10**, 3.25E−08)	2.70E+02 ± 2.16E+02 (0, 4.20E−04)
f_5	1.86E+03 ± 8.39E+02 (0, 1.83E−05)	4.63E+03 ± 7.82E+02 (0, 8.32E−06)	1.77E+03 ± 6.94E+02 (0, 1.28E−05)	1.33E+03 ± 5.41E+02 (0, 2.04E−04)	**9.49E+02 ± 4.57E+02** (0, 3.96E−06)	1.83E+03 ± 7.44E+02 (0, 1.37E−05)
f_6	5.98E−01 ± 1.46E+00 (85, 1.66E−13)	7.43E+01 ± 6.94E+01 (0, 8.24E−10)	7.97E−01 ± 1.64E+00 (80, 9.91E−24)	3.26E+01 ± 2.75E+01 (0, 1.20E−04)	1.99E−01 ± 8.91E−01 (95, 7.73E−16)	**2.81E−10 ± 1.15E−09** (**100**, 1.95E−06)
f_7	2.18E−02 ± 1.47E−02 (40, 1.95E−22)	2.37E−02 ± 1.60E−02 (40, 2.35E−20)	2.46E−02 ± 1.95E−02 (45, 1.17E−22)	1.08E−02 ± 1.00E−02 (**60**, 3.35E−21)	**1.02E−02 ± 9.21E−03** (55, 8.42E−22)	1.08E−02 ± 1.17E−02 (**60**, 2.18E−20)
f_8	2.11E+01 ± 5.93E−02 (0, 8.69E−01)	2.09E+01 ± 7.67E−02 (0, 7.21E−01)	2.10E+01 ± 5.27E−02 (0, 6.96E−01)	2.10E+01 ± 5.42E−02 (0, 9.36E−01)	2.01E+01 ± 1.50E−01 (0, 2.48E−10)	**2.00E+01 ± 4.53E−08** (0, 8.78E−01)
f_9	2.49E−01 ± 4.42E−01 (75, 6.54E−20)	2.44E+00 ± 1.66E+00 (0, 7.27E−20)	9.95E−02 ± 4.45E−01 (95, 4.78E−20)	1.49E−01 ± 3.65E−01 (85, 7.06E−20)	**0.00E+00 ± 0.00E+00** (**100**, 6.24E−20)	1.99E−01 ± 5.21E−01 (85, 5.67E−20)
f_{10}	4.73E+01 ± 2.84E+01 (0, 2.04E−02)	5.56E+01 ± 1.06E+01 (0, 2.88E−20)	5.05E+01 ± 1.17E+01 (0, 6.44E−02)	5.80E+01 ± 1.55E+01 (0, 1.07E−01)	4.21E+01 ± 1.15E+01 (0, 5.21E−20)	**3.75E+01 ± 7.06E+00** (0, 1.14E−01)
f_{11}	3.06E+01 ± 7.68E+00 (0, 7.47E−01)	1.97E+01 ± 2.59E+00 (0, 8.15E−28)	2.87E+01 ± 3.73E+00 (0, 7.89E−01)	2.80E+01 ± 2.38E+00 (0, 9.35E−01)	**1.16E+01 ± 2.54E+00** (0, 1.34E−27)	2.76E+01 ± 1.69E+00 (0, 9.02E−01)
f_{12}	4.69E+03 ± 4.35E+03 (5, 2.13E−07)	3.28E+03 ± 2.12E+03 (0, 8.87E−08)	1.71E+04 ± 6.69E+03 (0, 4.09E−01)	1.34E+04 ± 9.23E+03 (0, 2.90E−01)	2.53E+03 ± 3.48E+03 (0, 1.21E−09)	**2.18E+02 ± 5.19E+02** (**40**, 4.66E−01)
f_{13}	1.83E+00 ± 8.45E−01 (0, 3.78E−02)	2.52E+00 ± 7.18E−01 (0, 1.89E−02)	1.59E+00 ± 1.26E−01 (0, 4.60E−02)	1.34E+00 ± 1.83E−01 (0, 5.28E−02)	1.60E+00 ± 3.19E−01 (0, 2.34E−04)	**8.84E−01 ± 1.36E−01** (0, 4.66E−02)
f_{14}	1.36E+01 ± 2.50E−01 (0, 8.67E−01)	1.23E+01 ± 5.11E−01 (0, 2.47E−01)	1.29E+01 ± 3.72E−01 (0, 6.21E−01)	1.33E+01 ± 2.87E−01 (0, 8.80E−01)	1.23E+01 ± 6.17E−01 (0, 6.29E−09)	**8.98E−01 ± 1.56E−01** (0, 1.67E−05)
f_{15}	4.44E−32 ± 9.86E−32 (**100**, 3.84E−35)	1.05E−32 ± 3.56E−32 (**100**, 3.78E−35)	9.86E−33 ± 1.42E−32 (**100**, 3.85E−35)	5.61E−32 ± 2.25E−31 (**100**, 3.79E−35)	**0.00E+00 ± 0.00E+00** (**100**, 3.72E−35)	2.03E−32 ± 6.06E−32 (**100**, 3.74E−35)
f_{16}	5.98E−01 ± 1.46E+00 (65, 1.20E−12)	3.78E+01 ± 4.91E+01 (0, 1.03E−09)	5.98E−01 ± 1.46E+00 (85, 4.66E−24)	3.23E+01 ± 2.70E+01 (0, 2.37E−05)	1.99E−01 ± 8.91E−01 (90, 6.91E−17)	**1.01E−11 ± 2.07E−11** (**100**, 1.84E−19)
f_{17}	6.39E−15 ± 5.22E−15 (**100**, 6.34E−33)	8.40E−01 ± 6.00E−01 (30, 8.82E−18)	3.18E−01 ± 5.84E−01 (75, 6.39E−21)	**2.66E−15 ± 0.00E+00** (**100**, 4.32E−33)	5.68E−15 ± 1.30E−15 (**100**, 6.66E−33)	**2.66E−15 ± 0.00E+00** (**100**, 3.85E−33)

Table 5 continued

Fun.	jDE AveFit ± SD (RS, DivRate)	SaDE AveFit ± SD (RS, DivRate)	EPSDE AveFit ± SD (RS, DivRate)	DECLS AveFit ± SD (RS, DivRate)	CoDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_{18}	3.32E−03 ± 7.29E−03 (80, 2.68E−21)	1.30E−02 ± 2.05E−02 (60, 2.20E−21)	8.26E−03 ± 1.14E−02 (45, 1.57E−21)	7.40E−04 ± 2.28E−03 (90, 2.87E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.48E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.37E−21)
f_{19}	1.50E−12 ± 2.57E−12 (**100**, 2.59E−18)	2.72E−06 ± 4.12E−06 (0, 8.33E−14)	1.14E−14 ± 5.12E−14 (**100**, 1.91E−21)	1.57E−03 ± 4.33E−03 (0, 5.68E−08)	8.98E−17 ± 1.09E−16 (**100**, 2.98E−22)	**8.98E−19 ± 1.40E−18** (**100**, 2.67E−24)
f_{20}	6.06E+01 ± 9.11E+01 (**25**, 6.29E−15)	1.56E+02 ± 8.98E+01 (5, 1.09E−21)	**1.49E+01 ± 1.43E+01** (5, 7.08E−07)	2.51E+01 ± 2.89E+01 (10, 9.71E−07)	8.35E+01 ± 7.63E+01 (0, 2.70E−18)	1.73E+01 ± 1.29E+01 (0, 2.57E−07)
f_{21}	2.45E−01 ± 1.57E−01 (0, 2.52E−05)	4.05E−01 ± 1.32E−01 (0, 3.11E−05)	3.65E−01 ± 1.76E−01 (0, 3.19E−05)	2.00E−01 ± 6.63E−09 (0, 3.23E−05)	2.00E−01 ± 2.38E−11 (0, 2.85E−06)	**1.45E−01 ± 8.87E−02** (**20**, 2.33E−05)
f_{22}	2.49E−01 ± 4.42E−01 (75, 1.43E−18)	8.95E−01 ± 9.63E−01 (35, 1.37E−18)	**4.97E−02 ± 2.22E−01** (**95**, 9.71E−19)	**4.97E−02 ± 2.22E−01** (**95**, 1.89E−18)	**4.97E−02 ± 2.22E−01** (**95**, 1.11E−18)	1.99E−01 ± 4.08E−01 (80, 1.53E−18)
f_{23}	2.07E−02 ± 9.27E−02 (95, 3.38E−20)	1.55E−02 ± 5.07E−02 (90, 3.39E−20)	1.65E−32 ± 1.89E−33 (**100**, 3.40E−34)	**1.57E−32 ± 2.81E−48** (**100**, 3.50E−34)	**1.57E−32 ± 2.81E−48** (**100**, 3.24E−34)	**1.57E−32 ± 2.81E−48** (**100**, 3.00E−34)
f_{24}	1.80E−01 ± 8.04E−01 (95, 4.38E−21)	1.41E−32 ± 1.16E−33 (**100**, 1.51E−34)	1.45E−32 ± 1.36E−33 (**100**, 1.54E−34)	3.70E−32 ± 1.03E−31 (**100**, 1.50E−34)	**1.35E−32 ± 2.81E−48** (**100**, 1.49E−34)	**1.35E−32 ± 2.81E−48** (**100**, 1.51E−34)
f_{25}	7.93E−123 ± 3.15E−122 (**100**, 6.46E−248)	4.95E−106 ± 2.02E−105 (**100**, 2.34E−214)	**9.97E−134 ± 4.44E−133** (**100**, 1.32E−269)	8.95E−87 ± 3.38E−86 (**100**, 1.20E−175)	2.12E−35 ± 1.98E−35 (**100**, 8.28E−74)	1.60E−26 ± 1.75E−26 (**100**, 5.49E−56)
f_{26}	5.00E−02 ± 2.24E−01 (95, 1.52E−05)	**0.00E+00 ± 0.00E+00** (**100**, 8.85E−06)	6.50E−01 ± 9.88E−01 (55, 9.36E−06)	**0.00E+00 ± 0.00E+00** (**100**, 1.58E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.39E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.22E−05)
f_{27}	3.65E−03 ± 1.60E−03 (**100**, 2.23E−03)	5.09E−03 ± 2.29E−03 (**100**, 1.12E−03)	2.91E−03 ± 3.03E−03 (95, 6.04E−04)	2.79E−03 ± 1.04E−03 (**100**, 4.06E−03)	2.63E−03 ± 9.34E−04 (**100**, 4.14E−03)	**2.04E−03 ± 6.74E−04** (**100**, 3.35E−03)
f_{28}	2.02E+01 ± 7.40E+00 (0, 8.39E−03)	2.76E−04 ± 1.08E−03 (40, 5.93E−11)	8.38E+00 ± 2.23E+00 (0, 5.53E−04)	7.05E−11 ± 2.24E−10 (**100**, 4.02E−24)	9.43E−08 ± 2.52E−07 (55, 3.55E−18)	**1.98E−17 ± 1.17E−17** (**100**, 3.17E−38)

Table 6 Results of all algorithms on problems in 30 dimensions

Fun.	jDE AveFit ± SD (RS, DivRate)	SaDE AveFit ± SD (RS, DivRate)	EPSDE AveFit ± SD (RS, DivRate)	DECLS AveFit ± SD (RS, DivRate)	CoDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_1	**0.00E+00 ± 0.00E+00** (**100**, 1.92E−30)	**0.00E+00 ± 0.00E+00** (**100**, 3.27E−31)	2.02E−29 ± 6.22E−29 (**100**, 3.27E−31)	1.02E−28 ± 4.06E−28 (**100**, 2.29E−31)	**0.00E+00 ± 0.00E+00** (**100**, 2.33E−31)	**0.00E+00 ± 0.00E+00** (**100**, 2.34E−31)
f_2	1.25E−06 ± 1.80E−06 (**100**, 6.93E−12)	8.61E−06 ± 1.51E−05 (**100**, 7.76E−13)	1.26E−09 ± 5.61E−09 (**100**, 1.52E−15)	1.71E−03 ± 3.22E−03 (35, 1.38E−07)	**6.72E−15 ± 2.07E−14** (**100**, 1.98E−20)	4.69E−11 ± 3.05E−11 (**100**, 9.12E−17)
f_3	1.46E+05 ± 9.88E+04 (0, 6.16E−04)	4.21E+05 ± 1.86E+05 (0, 2.59E−07)	1.09E+06 ± 3.15E+06 (0, 2.98E−02)	1.61E+06 ± 1.05E+06 (0, 1.27E−01)	1.10E+05 ± 4.48E+04 (0, 1.08E−05)	**1.22E+01 ± 1.82E+01** (0, 1.02E−04)
f_4	5.13E−02 ± 8.17E−02 (0, 1.33E−07)	1.26E+02 ± 1.29E+02 (0, 3.82E−05)	4.03E+02 ± 1.58E+03 (5, 9.08E−03)	9.51E+00 ± 1.50E+01 (0, 2.90E−04)	**9.17E−03 ± 2.51E−02** (**10**, 3.25E−08)	**2.70E+02 ± 2.16E+02** (0, 4.20E−04)
f_5	1.21E+03 ± 4.56E+02 (0, 1.52E−04)	2.95E+03 ± 7.40E+02 (0, 7.06E−07)	1.93E+03 ± 1.68E+03 (0, 2.49E−04)	1.33E+03 ± 5.41E+02 (0, 2.04E−04)	**9.49E+02 ± 4.57E+02** (0, 3.96E−06)	1.83E+03 ± 7.44E+02 (0, 1.37E−05)
f_6	2.62E+01 ± 2.62E+01 (0, 4.68E−07)	5.68E+01 ± 3.52E+01 (0, 1.06E−08)	3.99E−01 ± 1.23E+00 (90, 2.62E−24)	3.26E+01 ± 2.75E+01 (0, 1.20E−04)	1.99E−01 ± 8.91E−01 (95, 7.73E−16)	**2.81E−10 ± 1.15E−09** (**100**, 1.95E−06)
f_7	**9.72E−03 ± 7.99E−03** (75, 4.29E−21)	1.27E−02 ± 9.34E−03 (**80**, 2.19E−17)	1.75E−02 ± 1.29E−02 (40, 2.05E−22)	1.08E−02 ± 1.00E−02 (60, 3.35E−21)	1.02E−02 ± 9.21E−03 (55, 8.42E−22)	1.08E−02 ± 1.17E−02 (60, 2.18E−20)
f_8	2.11E+01 ± 5.95E−02 (0, 9.79E−01)	2.09E+01 ± 6.17E−02 (0, 7.45E−01)	2.09E+01 ± 5.15E−02 (0, 8.02E−01)	2.10E+01 ± 5.42E−02 (0, 9.36E−01)	2.01E+01 ± 1.50E−01 (0, 2.48E−10)	**2.00E+01 ± 4.53E−08** (0, 8.78E−01)
f_9	**0.00E+00 ± 0.00E+00** (**100**, 8.36E−20)	9.95E−02 ± 3.06E−01 (90, 5.01E−20)	**0.00E+00 ± 0.00E+00** (**100**, 5.78E−20)	1.49E−01 ± 3.65E−01 (85, 7.06E−20)	**0.00E+00 ± 0.00E+00** (**100**, 6.24E−20)	1.99E−01 ± 5.21E−01 (85, 5.67E−20)
f_{10}	1.49E+02 ± 5.53E+01 (0, 1.04E−01)	5.18E+01 ± 1.23E+01 (0, 3.35E−20)	5.63E+01 ± 1.31E+01 (0, 6.25E−02)	5.80E+01 ± 1.55E+01 (0, 1.07E−01)	4.21E+01 ± 1.15E+01 (0, 5.21E−20)	**3.75E+01 ± 7.06E+00** (0, 1.14E−01)
f_{11}	3.59E+01 ± 2.83E+00 (0, 9.85E−01)	1.70E+01 ± 2.49E+00 (0, 2.83E−26)	3.01E+01 ± 3.92E+00 (0, 8.08E−01)	2.80E+01 ± 2.38E+00 (0, 9.35E−01)	**1.16E+01 ± 2.54E+00** (0, 1.34E−27)	2.76E+01 ± 1.69E+00 (0, 9.02E−01)
f_{12}	3.51E+04 ± 4.50E+04 (0, 1.94E−01)	2.99E+03 ± 1.81E+03 (0, 3.54E−07)	1.97E+04 ± 5.92E+03 (0, 4.23E−01)	1.34E+04 ± 9.23E+03 (0, 2.90E−01)	2.53E+03 ± 3.48E+03 (0, 1.21E−09)	**2.18E+02 ± 5.19E+02** (**40**, 4.66E−01)
f_{13}	3.85E+00 ± 1.41E+00 (0, 5.21E−02)	4.01E+00 ± 2.89E−01 (0, 4.57E−02)	1.97E+00 ± 1.62E−01 (0, 5.05E−02)	1.34E+00 ± 1.83E−01 (0, 5.28E−02)	1.60E+00 ± 3.19E−01 (0, 2.34E−04)	**8.84E−01 ± 1.36E−01** (0, 4.66E−02)
f_{14}	1.38E+01 ± 2.45E−01 (0, 9.04E−01)	1.27E+01 ± 3.18E−01 (0, 4.26E−01)	1.28E+01 ± 2.33E−01 (0, 6.77E−01)	1.33E+01 ± 2.87E−01 (0, 8.80E−01)	1.23E+01 ± 6.17E−01 (0, 6.29E−09)	**8.98E−01 ± 1.56E−01** (0, 1.67E−05)
f_{15}	**0.00E+00 ± 0.00E+00** (**100**, 1.46E−34)	**0.00E+00 ± 0.00E+00** (**100**, 3.69E−35)	1.85E−33 ± 6.03E−33 (**100**, 3.73E−35)	5.61E−32 ± 2.25E−31 (**100**, 3.79E−35)	**0.00E+00 ± 0.00E+00** (**100**, 3.72E−35)	2.03E−32 ± 6.06E−32 (**100**, 3.74E−35)
f_{16}	1.29E+01 ± 1.34E+01 (0, 3.07E−07)	2.53E+01 ± 2.29E+01 (0, 5.27E−09)	3.99E−01 ± 1.23E+00 (90, 5.89E−24)	3.23E+01 ± 2.70E+01 (0, 2.37E−05)	1.99E−01 ± 8.91E−01 (90, 6.91E−17)	**1.01E−11 ± 2.07E−11** (**100**, 1.84E−19)
f_{17}	3.02E−15 ± 1.09E−15 (**100**, 4.71E−33)	1.51E−01 ± 3.71E−01 (85, 6.87E−21)	3.73E−15 ± 1.67E−15 (**100**, 4.75E−33)	**2.66E−15 ± 0.00E+00** (**100**, 4.32E−33)	5.68E−15 ± 1.30E−15 (**100**, 6.66E−33)	**2.66E−15 ± 0.00E+00** (**100**, 3.85E−33)

Table 6 continued

Fun.	jDE AveFit ± SD (RS, DivRate)	SaDE AveFit ± SD (RS, DivRate)	EPSDE AveFit ± SD (RS, DivRate)	DECLS AveFit ± SD (RS, DivRate)	CoDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_{18}	**0.00E+00 ± 0.00E+00** (**100**, 3.61E−21)	4.55E−03 ± 9.17E−03 (75, 2.21E−21)	9.86E−04 ± 3.14E−03 (90, 2.20E−21)	7.40E−04 ± 2.28E−03 (90, 2.87E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.48E−21)	**0.00E+00 ± 0.00E+00** (**100**, 2.37E−21)
f_{19}	5.37E−07 ± 1.06E−06 (0, 1.92E−12)	1.60E−06 ± 5.34E−06 (10, 4.97E−14)	2.48E−20 ± 1.11E−19 (**100**, 1.60E−26)	1.57E−03 ± 4.33E−03 (0, 5.68E−08)	8.98E−17 ± 1.09E−16 (**100**, 2.98E−22)	8.98E−19 ± 1.40E−18 (**100**, 2.67E−24)
f_{20}	7.43E+01 ± 1.06E+02 (0, 2.24E−06)	2.33E+01 ± 4.02E+01 (**35**, 3.01E−15)	1.77E+01 ± 1.47E+01 (0, 8.60E−07)	2.51E+01 ± 2.89E+01 (10, 9.71E−07)	8.35E+01 ± 7.63E+01 (0, 2.70E−18)	**1.73E+01 ± 1.29E+01** (0, 2.57E−07)
f_{21}	1.90E−01 ± 3.08E−02 (0, 3.27E−05)	2.60E−01 ± 6.81E−02 (0, 2.28E−05)	1.95E−01 ± 5.10E−02 (0, 2.33E−05)	2.00E−01 ± 6.63E−09 (0, 3.23E−05)	2.00E−01 ± 2.38E−11 (0, 2.85E−06)	**1.45E−01 ± 8.87E−02** (**20**, 2.33E−05)
f_{22}	**0.00E+00 ± 0.00E+00** (**100**, 2.03E−18)	**0.00E+00 ± 0.00E+00** (**100**, 1.32E−18)	**0.00E+00 ± 0.00E+00** (**100**, 1.38E−18)	4.97E−02 ± 2.22E−01 (95, 1.89E−18)	4.97E−02 ± 2.22E−01 (95, 1.11E−18)	1.99E−01 ± 4.08E−01 (80, 1.53E−18)
f_{23}	**1.57E−32 ± 2.81E−48** (**100**, 8.30E−34)	5.18E−03 ± 2.32E−02 (95, 1.18E−20)	**1.57E−32 ± 2.81E−48** (**100**, 3.99E−34)	**1.57E−32 ± 2.81E−48** (**100**, 3.50E−34)	**1.57E−32 ± 2.81E−48** (**100**, 3.24E−34)	**1.57E−32 ± 2.81E−48** (**100**, 3.00E−34)
f_{24}	**1.35E−32 ± 2.81E−48** (**100**, 5.85E−34)	2.20E−03 ± 4.51E−03 (80, 3.92E−22)	1.36E−32 ± 2.76E−34 (**100**, 1.46E−34)	3.70E−32 ± 1.03E−31 (**100**, 1.50E−34)	**1.35E−32 ± 2.81E−48** (**100**, 1.49E−34)	1.35E−32 ± 2.81E−48 (**100**, 1.51E−34)
f_{25}	2.24E−36 ± 2.30E−36 (**100**, 5.92E−76)	1.98E−79 ± 4.38E−79 (**100**, 1.50E−161)	3.75E−86 ± 1.34E−85 (**100**, 2.12E−174)	**8.95E−87 ± 3.38E−86** (**100**, 1.20E−175)	2.12E−35 ± 1.98E−35 (**100**, 8.28E−74)	1.60E−26 ± 1.75E−26 (**100**, 5.49E−56)
f_{26}	**0.00E+00 ± 0.00E+00** (**100**, 1.88E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.06E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.19E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.58E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.39E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.22E−05)
f_{27}	8.74E−03 ± 4.90E−03 (80, 5.50E−03)	2.49E−03 ± 1.14E−03 (**100**, 1.70E−03)	**1.11E−03 ± 4.98E−04** (**100**, 1.20E−03)	2.79E−03 ± 1.04E−03 (**100**, 4.06E−03)	2.63E−03 ± 9.34E−04 (**100**, 4.14E−03)	2.04E−03 ± 6.74E−04 (**100**, 3.35E−03)
f_{28}	4.55E−01 ± 6.18E−01 (0, 2.90E−05)	1.75E−09 ± 5.59E−09 (90, 5.70E−21)	2.58E+00 ± 9.02E−01 (0, 9.56E−05)	7.05E−11 ± 2.24E−10 (**100**, 4.02E−24)	9.43E−08 ± 2.52E−07 (55, 3.55E−18)	**1.98E−17 ± 1.17E−17** (**100**, 3.17E−38)

Table 7 Results of all algorithms on problems in 50 dimensions

Fun.	jDE AveFit ± SD (RS, DivRate)	SaDE AveFit ± SD (RS, DivRate)	EPSDE AveFit ± SD (RS, DivRate)	DECLS AveFit ± SD (RS, DivRate)	CoDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_1	**0.00E+00 ± 0.00E+00** (**100**, 4.83E−30)	**0.00E+00 ± 0.00E+00** (**100**, 1.37E−30)	**0.00E+00 ± 0.00E+00** (**100**, 1.78E−30)	2.02E−29 ± 6.22E−29 (**100**, 4.05E−31)	**0.00E+00 ± 0.00E+00** (**100**, 4.06E−31)	**0.00E+00 ± 0.00E+00** (**100**, 4.02E−31)
f_2	3.88E−01 ± 6.66E−01 (0, 8.41E−07)	1.06E−01 ± 7.30E−02 (0, 3.94E−09)	2.67E+03 ± 7.15E+03 (85, 3.59E−02)	2.09E+02 ± 4.24E+02 (0, 7.56E−03)	5.65E−04 ± 5.67E−04 (10, 5.41E−10)	**1.28E−09 ± 4.62E−10** (**100**, 9.22E−16)
f_3	6.04E+05 ± 2.70E+05 (0, 1.75E−03)	7.58E+05 ± 2.38E+05 (0, 3.63E−07)	8.46E+06 ± 1.51E+07 (0, 5.05E−02)	5.65E+06 ± 4.01E+06 (0, 1.25E−01)	4.89E+05 ± 1.91E+05 (0, 2.68E−05)	**3.05E+03 ± 8.05E+03** (0, 2.94E−03)
f_4	**3.81E+02 ± 1.88E+02** (0, 3.50E−04)	3.10E+03 ± 2.06E+03 (0, 3.24E−04)	3.52E+03 ± 1.14E+04 (0, 3.40E−02)	5.37E+03 ± 3.55E+03 (0, 7.48E−02)	3.98E+02 ± 3.46E+02 (0, 3.87E−04)	4.43E+03 ± 1.51E+03 (0, 2.41E−03)
f_5	2.86E+03 ± 6.51E+02 (0, 1.21E−04)	5.67E+03 ± 9.75E+02 (0, 4.23E−05)	2.62E+03 ± 8.30E+02 (0, 2.52E−03)	3.11E+03 ± 4.90E+02 (0, 8.37E−04)	**2.32E+03 ± 5.82E+02** (0, 1.14E−05)	4.05E+03 ± 8.58E+02 (0, 9.18E−05)
f_6	5.36E+01 ± 3.03E+01 (0, 1.56E−07)	9.10E+01 ± 3.11E+01 (0, 1.02E−09)	3.99E−01 ± 1.23E+00 (90, 5.42E−22)	4.95E+01 ± 2.57E+01 (0, 1.36E−04)	3.39E+01 ± 2.46E+01 (0, 3.95E−08)	**4.74E−11 ± 3.53E−11** (**100**, 1.85E−06)
f_7	**1.35E−03 ± 4.34E−03** (95, 3.41E−15)	2.82E−03 ± 1.01E−02 (90, 2.20E−13)	9.59E−03 ± 1.62E−02 (80, 1.26E−22)	3.32E−03 ± 6.84E−03 (90, 8.69E−11)	1.36E−03 ± 3.34E−03 (**100**, 1.75E−18)	1.16E−02 ± 8.61E−03 (45, 2.70E−20)
f_8	2.13E+01 ± 5.07E−02 (0, 9.88E−01)	2.11E+01 ± 4.88E−02 (0, 7.47E−01)	2.11E+01 ± 3.21E−02 (0, 7.92E−01)	2.11E+01 ± 3.11E−02 (0, 9.75E−01)	2.11E+01 ± 5.35E−02 (0, 9.48E−01)	**2.00E+01 ± 3.53E−08** (0, 9.22E−01)
f_9	**0.00E+00 ± 0.00E+00** (**100**, 8.13E−20)	4.97E−02 ± 2.22E−01 (95, 4.65E−20)	**0.00E+00 ± 0.00E+00** (**100**, 5.14E−20)	**0.00E+00 ± 0.00E+00** (**100**, 7.58E−20)	3.24E−01 ± 1.01E+00 (45, 6.83E−05)	**0.00E+00 ± 0.00E+00** (**100**, 5.96E−20)
f_{10}	3.05E+02 ± 8.90E+01 (0, 1.03E−01)	9.98E+01 ± 2.05E+01 (0, 5.32E−20)	1.88E+02 ± 1.96E+01 (0, 5.09E−02)	1.40E+02 ± 2.19E+01 (0, 1.13E−01)	**6.91E+01 ± 1.93E+01** (0, 9.53E−20)	7.20E+01 ± 1.16E+01 (0, 1.22E−01)
f_{11}	6.96E+01 ± 6.91E+00 (0, 9.91E−01)	3.22E+01 ± 3.78E+00 (0, 3.75E−08)	5.86E+01 ± 1.57E+00 (0, 8.28E−01)	5.84E+01 ± 3.36E+00 (0, 9.61E−01)	**2.08E+01 ± 4.73E+00** (0, 3.51E−12)	5.57E+01 ± 2.26E+00 (0, 9.46E−01)
f_{12}	8.13E+04 ± 8.89E+04 (0, 1.43E−01)	2.69E+04 ± 1.76E+04 (0, 9.66E−04)	1.75E+05 ± 3.10E+04 (0, 3.75E−01)	4.93E+04 ± 2.51E+04 (0, 1.84E−01)	1.13E+04 ± 1.44E+04 (0, 3.01E−06)	**3.06E+02 ± 4.72E+02** (**10**, 3.67E−01)
f_{13}	7.80E+00 ± 3.92E+00 (0, 5.64E−02)	1.07E+01 ± 6.18E−01 (0, 4.62E−02)	7.67E+00 ± 4.24E−01 (0, 5.56E−02)	2.87E+00 ± 4.27E−01 (0, 5.99E−02)	1.35E+01 ± 1.61E+00 (0, 6.58E−02)	**1.83E+00 ± 1.72E−01** (0, 4.89E−02)
f_{14}	2.38E+01 ± 2.19E−01 (0, 9.44E−01)	2.24E+01 ± 2.39E−01 (0, 5.04E−01)	2.27E+01 ± 2.04E−01 (0, 7.42E−01)	2.30E+01 ± 3.77E−01 (0, 9.16E−01)	2.23E+01 ± 4.74E−01 (0, 5.62E−01)	**2.21E+01 ± 3.48E−01** (0, 8.71E−01)
f_{15}	**0.00E+00 ± 0.00E+00** (**100**, 1.45E−34)	6.16E−34 ± 2.76E−33 (**100**, 1.46E−34)	**0.00E+00 ± 0.00E+00** (**100**, 1.46E−34)	3.08E−33 ± 1.12E−32 (**100**, 3.70E−35)	**0.00E+00 ± 0.00E+00** (**100**, 3.68E−35)	3.08E−33 ± 1.12E−32 (**100**, 3.70E−35)
f_{16}	4.27E+01 ± 2.57E+01 (0, 1.47E−07)	6.44E+01 ± 3.36E+01 (0, 8.98E−09)	5.98E−01 ± 1.46E+00 (85, 3.29E−24)	4.47E+01 ± 1.89E+01 (0, 3.42E−05)	1.74E+01 ± 1.82E+00 (0, 3.48E−08)	**2.66E−12 ± 1.18E−11** (**100**, 3.01E−21)
f_{17}	6.22E−15 ± 0.00E+00 (**100**, 1.26E−32)	2.47E−01 ± 4.45E−01 (75, 2.09E−20)	6.04E−15 ± 7.94E−16 (**100**, 9.10E−33)	6.04E−15 ± 7.94E−16 (**100**, 1.49E−32)	6.57E−15 ± 1.09E−15 (**100**, 5.94E−33)	**2.66E−15 ± 0.00E+00** (**100**, 4.44E−33)

Table 7 continued

Fun.	jDE AveFit ± SD (RS, DivRate)	SaDE AveFit ± SD (RS, DivRate)	EPSDE AveFit ± SD (RS, DivRate)	DECLS AveFit ± SD (RS, DivRate)	CoDE AveFit ± SD (RS, DivRate)	DMGBDE AveFit ± SD (RS, DivRate)
f_{18}	**0.00E+00 ± 0.00E+00** (**100**, 5.65E−21)	1.48E−03 ± 3.70E−03 (85, 3.32E−21)	7.40E−04 ± 2.28E−03 (90, 3.23E−21)	**0.00E+00 ± 0.00E+00** (**100**, 5.22E−21)	**0.00E+00 ± 0.00E+00** (**100**, 3.83E−21)	**0.00E+00 ± 0.00E+00** (**100**, 4.28E−21)
f_{19}	3.73E−02 ± 3.13E−02 (0, 1.02E−07)	5.60E−03 ± 5.37E−03 (0, 1.85E−10)	6.04E+02 ± 2.30E+03 (75, 1.28E−02)	1.51E+02 ± 2.26E+02 (0, 8.98E−03)	3.90E−05 ± 3.53E−05 (0, 3.84E−11)	**2.88E−17 ± 2.21E−17** (**100**, 2.93E−23)
f_{20}	3.81E+02 ± 2.98E+02 (0, 3.56E−06)	**2.40E+01 ± 3.55E+01** (0, 9.77E−15)	5.73E+02 ± 2.46E+02 (0, 1.87E−06)	1.42E+02 ± 3.36E+02 (**15**, 2.26E−06)	9.07E+02 ± 1.31E+02 (0, 4.52E−06)	5.38E+02 ± 3.65E+02 (0, 2.11E−07)
f_{21}	2.45E−01 ± 5.10E−02 (0, 3.15E−05)	3.60E−01 ± 5.98E−02 (0, 2.46E−05)	2.50E−01 ± 5.13E−02 (0, 1.96E−05)	2.30E−01 ± 4.70E−02 (0, 2.97E−05)	2.00E−01 ± 2.92E−10 (0, 1.81E−05)	**1.80E−01 ± 5.23E−02** (**5**, 1.79E−05)
f_{22}	**0.00E+00 ± 0.00E+00** (**100**, 2.70E−18)	**0.00E+00 ± 0.00E+00** (**100**, 1.74E−18)	**0.00E+00 ± 0.00E+00** (**100**, 1.56E−18)	**0.00E+00 ± 0.00E+00** (**100**, 2.82E−18)	1.39E+01 ± 5.61E+00 (0, 1.79E−02)	**0.00E+00 ± 0.00E+00** (**100**, 2.35E−18)
f_{23}	**9.42E−33 ± 2.81E−48** (**100**, 8.24E−34)	3.11E−03 ± 1.39E−02 (95, 1.28E−20)	**9.42E−33 ± 2.81E−48** (**100**, 8.04E−34)	**9.42E−33 ± 2.81E−48** (**100**, 3.97E−34)	**9.42E−33 ± 2.81E−48** (**100**, 3.73E−34)	**9.42E−33 ± 2.81E−48** (**100**, 3.61E−34)
f_{24}	**1.35E−32 ± 2.81E−48** (**100**, 5.82E−34)	5.49E−04 ± 2.46E−03 (95, 9.60E−23)	**1.35E−32 ± 2.81E−48** (**100**, 5.85E−34)	2.37E−32 ± 4.17E−32 (**100**, 1.48E−34)	**1.35E−32 ± 2.81E−48** (**100**, 1.47E−34)	1.36E−32 ± 2.76E−34 (**100**, 1.48E−34)
f_{25}	1.32E−28 ± 7.43E−29 (**100**, 4.89E−61)	2.52E−52 ± 2.50E−52 (**100**, 1.93E−108)	5.80E−57 ± 1.93E−56 (**100**, 1.12E−116)	**2.88E−63 ± 1.17E−62** (**100**, 4.96E−129)	3.39E−18 ± 2.00E−18 (**100**, 7.27E−40)	5.47E−23 ± 3.04E−23 (**100**, 1.31E−49)
f_{26}	**0.00E+00 ± 0.00E+00** (**100**, 1.82E−05)	**0.00E+00 ± 0.00E+00** (**100**, 9.75E−06)	**0.00E+00 ± 0.00E+00** (**100**, 1.06E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.69E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.28E−05)	**0.00E+00 ± 0.00E+00** (**100**, 1.33E−05)
f_{27}	1.62E−02 ± 6.94E−03 (5, 4.32E−03)	5.35E−03 ± 1.91E−03 (95, 1.13E−03)	**1.36E−03 ± 5.92E−04** (**100**, 1.03E−03)	4.29E−03 ± 9.93E−04 (**100**, 3.32E−03)	5.16E−03 ± 1.63E−03 (**100**, 3.59E−03)	4.13E−03 ± 7.03E−04 (**100**, 2.91E−03)
f_{28}	2.36E+00 ± 1.60E+00 (0, 3.40E−04)	2.25E−02 ± 6.90E−02 (5, 3.06E−07)	3.79E+00 ± 1.20E+00 (0, 2.03E−04)	1.45E−06 ± 2.35E−06 (0, 6.56E−16)	5.45E−04 ± 4.48E−04 (0, 2.01E−11)	**1.29E−14 ± 2.22E−14** (**100**, 6.65E−31)

Table 8 A comparison of the RunTime (in s) of all algorithms running on the 28 test problems once

Dim.	DE	jDE	SaDE	EPSDE	DECLS	CoDE	DMGBDE
30D	1,051	586	1,309	1,700	1,015	844	807
50D	2,618	1,348	2,529	3,340	2,366	1,867	1,853

The DMM operator facilitates the proposed algorithm when embedding with gradient local search, which can be preliminarily observed from that, DMGBDE1 obtains significantly smaller fitness than GBDE on 10 problems and larger fitness than GBDE on 7 functions, DMGBDE is superior to GBDE on 8 problems and inferior to GBDE on

Table 9 Wilcoxon rank-sum test of the compared algorithms on 28 test problems with 30 and 50 dimensions

Dimension	Algorithm	DMGBDE (comparison results on all test functions)
30D	jDE	$+$: f_4, f_5, f_{22}, f_{25},
		$\approx$: $f_1, f_7, f_9, f_{15}, f_{17}, f_{18}, f_{23}, f_{24}, f_{26}$,
		$-$: $f_2, f_3, f_6, f_8, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{16}, f_{19}, f_{20}, f_{21}, f_{27}, f_{28}$.
	SaDE	$+$: $f_4, f_{11}, f_{22}, f_{25}$,
		$\approx$: $f_1, f_7, f_9, f_{15}, f_{17}, f_{20}, f_{23}, f_{26}, f_{27}$,
		$-$: $f_2, f_3, f_5, f_6, f_8, f_{10}, f_{12}, f_{13}, f_{14}, f_{16}, f_{18}, f_{19}, f_{21}, f_{24}, f_{28}$.
	EPSDE	$+$: $f_{19}, f_{22}, f_{25}, f_{27}$,
		$\approx$: $f_1, f_5, f_7, f_9, f_{15}, f_{18}, f_{20}, f_{23}, f_{24}, f_{26}$,
		$-$: $f_2, f_3, f_4, f_6, f_8, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{21}, f_{28}$.
	DECLS	$+$: f_4, f_5, f_{25},
		$\approx$: $f_1, f_7, f_9, f_{11}, f_{15}, f_{17}, f_{18}, f_{20}, f_{22}, f_{23}, f_{26}$,
		$-$: $f_2, f_3, f_6, f_8, f_{10}, f_{12}, f_{13}, f_{14}, f_{16}, f_{19}, f_{21}, f_{24}, f_{27}, f_{28}$.
	CoDE	$+$: $f_2, f_4, f_5, f_{11}, f_{25}$,
		$\approx$: $f_1, f_6, f_7, f_9, f_{10}, f_{15}, f_{18}, f_{21}, f_{22}, f_{23}, f_{24}, f_{26}$,
		$-$: $f_3, f_8, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{19}, f_{20}, f_{27}, f_{28}$.
50D	jDE	$+$: f_4, f_5, f_7, f_{25},
		$\approx$: $f_1, f_9, f_{15}, f_{18}, f_{20}, f_{22}, f_{23}, f_{24}, f_{26}$,
		$-$: $f_2, f_3, f_6, f_8, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{19}, f_{21}, f_{27}, f_{28}$.
	SaDE	$+$: $f_4, f_7, f_{11}, f_{20}, f_{25}$,
		$\approx$: $f_1, f_9, f_{15}, f_{18}, f_{22}, f_{23}, f_{24}, f_{26}$,
		$-$: $f_2, f_3, f_5, f_6, f_8, f_{10}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{19}, f_{21}, f_{27}, f_{28}$.
	EPSDE	$+$: $f_4, f_5, f_7, f_{25}, f_{27}$,
		$\approx$: $f_1, f_9, f_{15}, f_{18}, f_{19}, f_{20}, f_{22}, f_{23}, f_{24}, f_{26}$,
		$-$: $f_2, f_3, f_6, f_8, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{21}, f_{28}$.
	DECLS	$+$: f_5, f_7, f_{20}, f_{25},
		$\approx$: $f_1, f_4, f_9, f_{15}, f_{18}, f_{22}, f_{23}, f_{24}, f_{26}, f_{27}$,
		$-$: $f_2, f_3, f_6, f_8, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{19}, f_{21}, f_{28}$.
	CoDE	$+$: f_4, f_5, f_7, f_{11},
		$\approx$: $f_1, f_{10}, f_{15}, f_{18}, f_{23}, f_{24}, f_{26}$,
		$-$: $f_2, f_3, f_6, f_8, f_9, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{19}, f_{20}, f_{21}, f_{22}, f_{25}, f_{27}, f_{28}$.

The notations $+(-)$ mean the algorithm for comparison is significantly superior to (inferior to) DMGBDE with significance level 0.05, $\approx$ means the two compared algorithms are not significantly different on the performance

Table 10 The significance test of the considered algorithms on 28 test problems

n	Test	jDE	SaDE	EPSDE	DECLS	CoDE	DMGBDE	p_g value
30	Fridman (ranking)	3.77E+00	4.20E+00	3.79E+00	4.25E+00	2.77E+00	2.23E+00	6.36E−05
	Holm (p value)	2.72E−04	3.42E−04	5.67E−03	5.67E−03	2.84E−01	–	–
50	Fridman (ranking)	3.84E+00	4.11E+00	3.79E+00	3.66E+00	3.09E+00	2.52E+00	1.70E−02
	Holm (p value)	7.40E−03	3.29E−02	3.37E−02	4.45E−02	2.53E−01	–	–

Test denotes the statistical method for comparison. The value p_g reflects the significance of the global difference

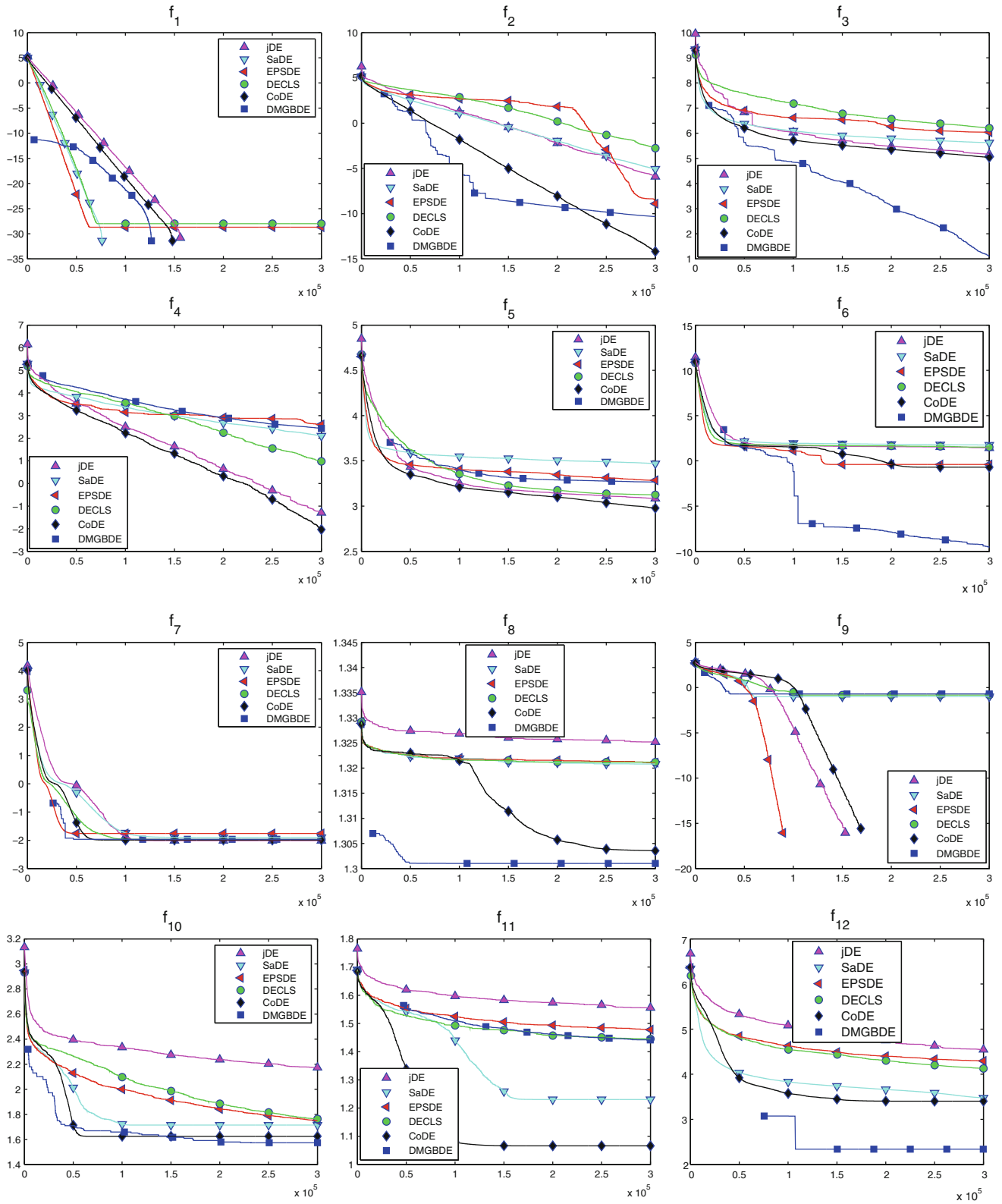

Fig. 2 The average best fitness (y-axis, which has been scaled by taking the log_{10} algorithm) w.r.t. the FES (x-axis) for functions f_1–f_{18}

5 problems at 1.0E+04n generations. By running GBDE and DMGBDE further on 30D benchmark functions, which is to enlarge the maximal FES *MaxFES* to 2.0E+04n, DMGBDE is superior (inferior) to GBDE on 8 (3) problems. Moreover, on the problems that DMGBDE performs worse, the population diversity of DMGBDE is

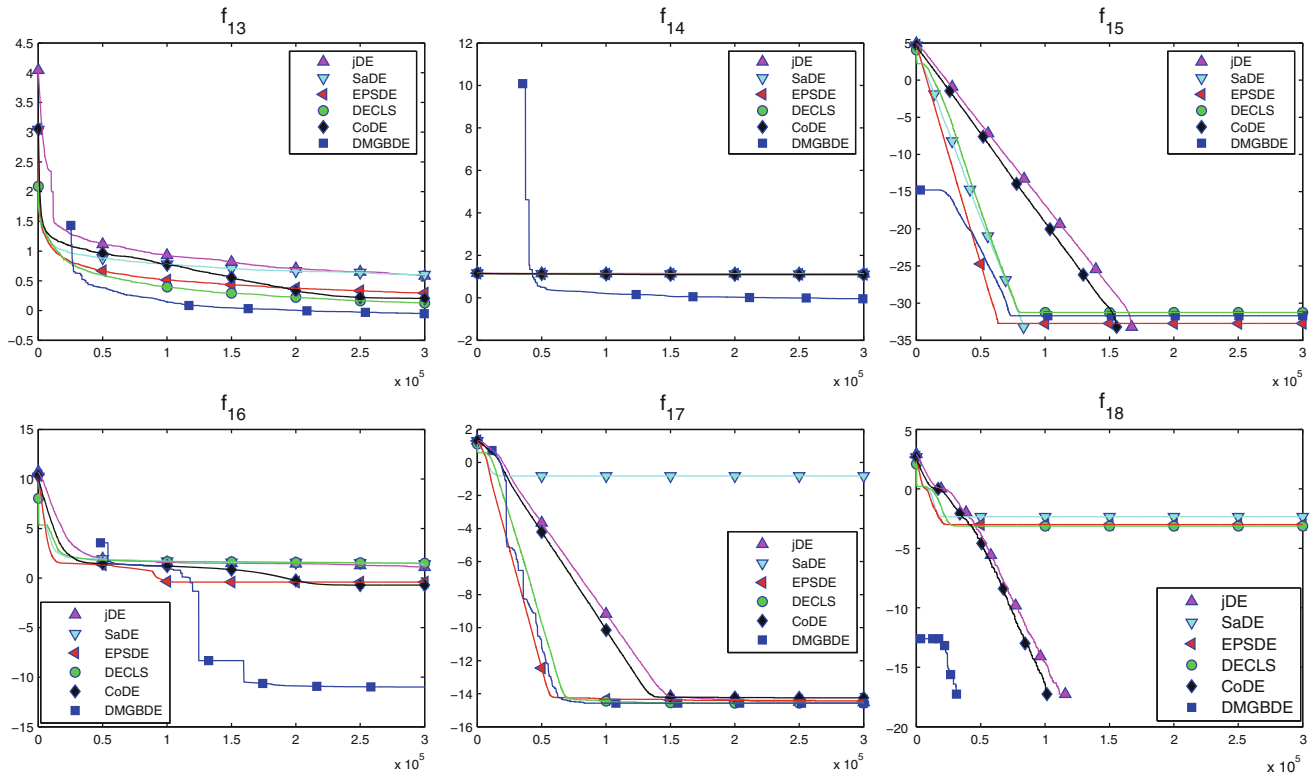

Fig. 2 continued

significantly larger that of GBDE, which implies the possibility that DMGBDE outperforms GBDE on these problems in a long run. Results of GBDE and DMGBDE on nine problems for 2.0E+04n generations are presented in Table 4. Therefore, the DMM operator makes a difference in the aspect of population diversity which may facilitate the algorithm to achieve a better solution in further generations.

To check whether the parameter settings in Table 2 are reasonable for the subsequent comparisons, a preliminary comparison is conducted by statistically comparing the results in Tables 5 and 6. Wilcoxon's test shows that the algorithm jDE with $N = 100$ achieves better (worse) performance on 11 (10) functions than that with $N = 30$; moreover, jDE with $N = 30$ obtains bad solutions on some relatively simple functions such as $f_{18}, f_{23}, f_{26}, f_{28}$, thus, the decrease of the population size for this algorithm largely increases the possibility of converging to a local optimum. For the algorithm SaDE, its performance with $N = 50$ are significantly better on 14 functions and worse on 3 functions than the performance with $N = 30$. For the algorithm EPSDE, its performance with $N = 50$ are significantly better on 11 functions and worse on 3 functions than the performance with $N = 30$. Thus, we can conclude that the proposed settings of the population size in Table 2 are better than the uniform setting $N = 30$ for 30D problems. The following comparisons are based on the results of the considered algorithms with the proposed parameter settings in Table 2.

Tables 6 and 7 demonstrate that the proposed DMGBDE achieves the best AveFit and SR on a large number of functions, where all the best AveFit and SR are labeled in boldface. It is also demonstrated that the diversity rate of DMGBDE algorithm is larger than that of the algorithms achieving the best mean fitness on most functions, which illustrates the possibility that our algorithm obtains the best mean fitness in further generations still exists. The large diversity rate maintained also illustrate the effectiveness of the proposed DMM operator.

Table 8 reveals that the proposed algorithm DMGBDE requires the least but one runtime for the overall running on all test problems, which is not surprising because a number of FES are utilized to obtain the approximate derivatives.

Table 9 which employs Wilcoxon test reveals more obvious evidence for the comparison of the considered algorithms on all test problems. For unimodal functions f_1–f_5, DMGBDE and CoDE significantly outperform the other four algorithms, DMGBDE is especially competitive on f_2, f_3 w.r.t. 50D problems, and CoDE is competitive on f_4, f_5. For these functions, the local search ability is deterministic, CoDE also achieves good performance because it can swiftly shift to the variant which harbors powerful

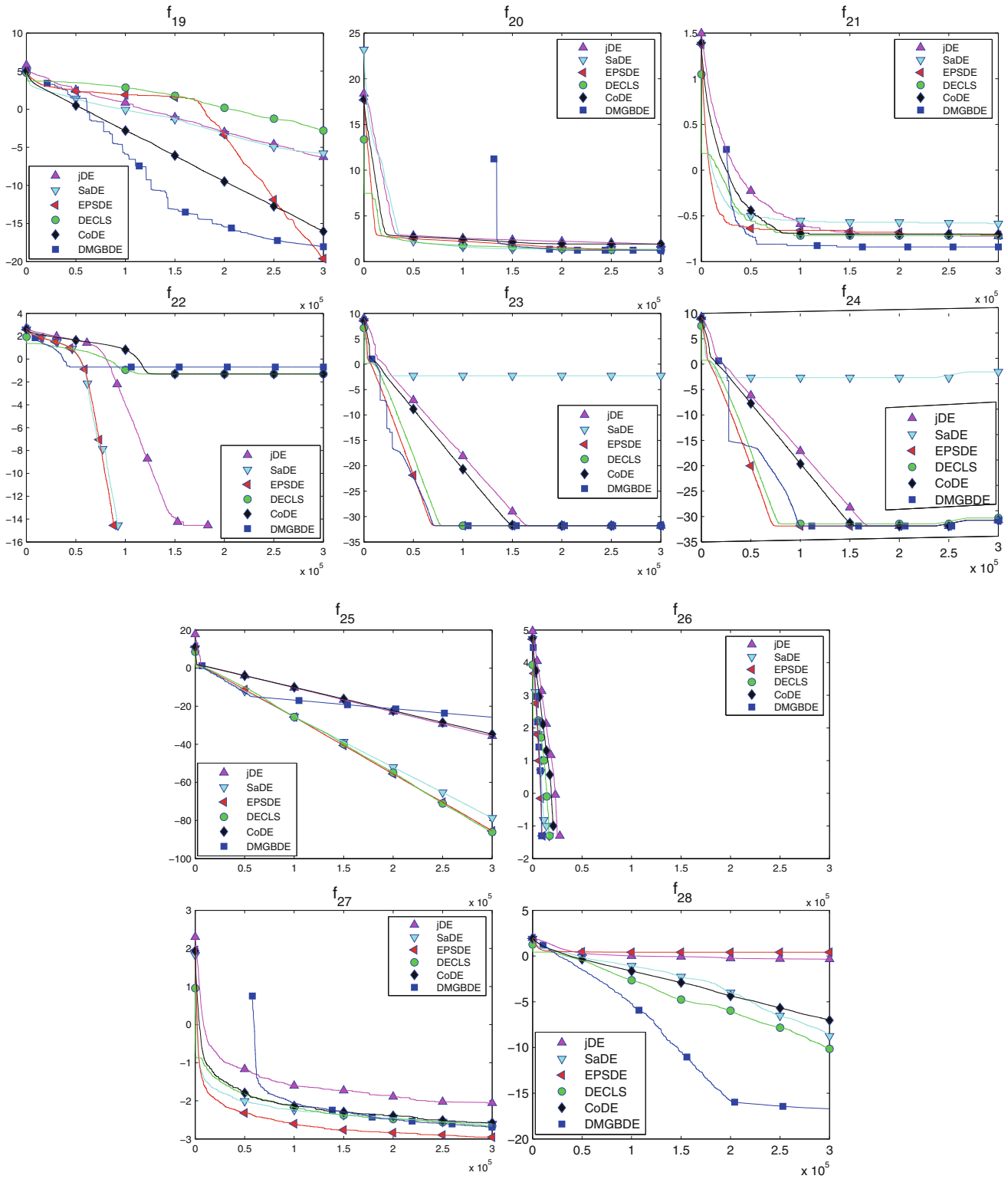

Fig. 3 The average best fitness (y-axis, which has been scaled by taking the log_{10} algorithm) w.r.t. the FES (x-axis) for functions f_{19}–f_{28}

exploitation ability. For shift multimodal functions f_6, f_9, DMGBDE performs well on f_6, while it may stagnate to a local optimum several times for f_9 in 30D, the DMM operator should be added with larger probability to decrease the risk. For the shift rotated functions f_7, f_8, f_{10}–f_{12}, DMGBDE achieves the best fitness on functions f_8, f_{10}, f_{12} w.r.t. 30D problems, and f_8, f_{12} w.r.t. 50D problems. For these functions, the performance can be further

Table 11 Results of parameter perturbations on problems in 30D

Fun.	*NNIMPROVE* = 10*n* AveFit ± SD	*MaxITER* = 3*n* AveFit ± SD	*N* = 2*n* AveFit ± SD	*pDEc* = 0.1 AveFit ± SD	*pDEc* = 0.5 AveFit ± SD	Proposed AveFit ± SD
f_1	2.08E−29 ± 6.20E−29	2.52E−30 ± 1.13E−29	**0.00E+00 ± 0.00E+00**	5.11E−29 ± 8.94E−29	1.33E−29 ± 4.59E−29	**0.00E+00 ± 0.00E+00**
f_2	9.03E−12 ± 1.90E−11	9.46E−12 ± 1.86E−11	1.14E−10 ± 6.99E−11	6.17E−11 ± 6.01E−11	**1.35E−12 ± 1.76E−12**	4.69E−11 ± 3.05E−11
f_3	9.44E−02 ± 1.39E−01	4.58E−02 ± 6.50E−02	8.71E+01 ± 1.19E+02	8.05E+00 ± 8.98E+00	**1.76E−02 ± 6.07E−02**	1.22E+01 ± 1.82E+01
f_4	1.54E+02 ± 2.91E+02	1.20E+02 ± 1.81E+02	1.41E+02 ± 1.14E+02	3.09E+02 ± 4.06E+02	**1.01E+02 ± 1.41E+02**	2.70E+02 ± 2.16E+02
f_5	2.14E+03 ± 7.35E+02	1.78E+03 ± 5.89E+02	**1.38E+03 ± 4.61E+02**	1.65E+03 ± 5.94E+02	1.87E+03 ± 6.71E+02	1.83E+03 ± 7.44E+02
f_6	1.99E−01 ± 8.91E−01	2.88E−09 ± 1.29E−08	**2.33E−10 ± 8.39E−10**	1.99E−01 ± 8.91E−01	3.99E−01 ± 1.23E+00	2.81E−10 ± 1.15E−09
f_7	**3.08E−03 ± 6.68E−03**	9.73E−03 ± 1.16E−02	3.57E−03 ± 6.21E−03	5.79E−03 ± 7.63E−03	8.74E−03 ± 9.12E−03	1.08E−02 ± 1.17E−02
f_8	**2.00E+01 ± 2.59E−08**	**2.00E+01 ± 3.28E−08**	**2.00E+01 ± 9.23E−08**	**2.00E+01 ± 3.76E−08**	**2.00E+01 ± 4.73E−08**	**2.00E+01 ± 4.53E−08**
f_9	4.97E−02 ± 2.22E−01	2.49E−01 ± 4.42E−01	**0.00E+00 ± 0.00E+00**	2.98E−01 ± 5.68E−01	9.95E−02 ± 3.06E−01	1.99E−01 ± 5.21E−01
f_{10}	4.02E+01 ± 9.32E+00	3.65E+01 ± 8.47E+00	3.78E+01 ± 6.45E+00	3.90E+01 ± 6.25E+00	**3.64E+01 ± 9.76E+00**	3.75E+01 ± 7.06E+00
f_{11}	2.90E+01 ± 1.64E+00	2.83E+01 ± 1.45E+00	2.92E+01 ± 1.15E+00	2.81E+01 ± 1.63E+00	**2.76E+01 ± 2.37E+00**	**2.76E+01 ± 1.69E+00**
f_{12}	6.97E+01 ± 1.51E+02	1.94E+02 ± 4.88E+02	1.42E+02 ± 4.62E+02	**5.94E+01 ± 1.62E+02**	8.19E+02 ± 1.87E+03	2.18E+02 ± 5.19E+02
f_{13}	1.00E+00 ± 1.25E−01	9.05E−01 ± 1.13E−01	1.08E+00 ± 1.61E−01	8.88E−01 ± 1.37E−01	9.32E−01 ± 1.22E−01	**8.84E−01 ± 1.36E−01**
f_{14}	1.27E+01 ± 2.60E−01	1.25E+01 ± 3.80E−01	1.23E+01 ± 3.77E−01	1.24E+01 ± 3.27E−01	1.24E+01 ± 3.54E−01	**8.98E−01 ± 1.56E−01**
f_{15}	**0.00E+00 ± 0.00E+00**	4.93E−33 ± 1.16E−32	**0.00E+00 ± 0.00E+00**	3.08E−33 ± 1.12E−32	8.63E−33 ± 1.79E−32	2.03E−32 ± 6.06E−32
f_{16}	3.99E−01 ± 1.23E+00	**6.13E−12 ± 1.52E−11**	**6.13E−12 ± 1.58E−11**	9.24E−12 ± 1.91E−11	3.99E−01 ± 1.23E+00	1.01E−11 ± 2.07E−11
f_{17}	**2.66E−15 ± 0.00E+00**	**2.66E−15 ± 0.00E+00**	**2.66E−15 ± 0.00E+00**	**2.66E−15 ± 0.00E+00**	**2.66E−15 ± 0.00E+00**	**2.66E−15 ± 0.00E+00**
f_{18}	**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**
f_{19}	8.51E−21 ± 1.62E−20	7.15E−20 ± 1.17E−19	5.21E−17 ± 8.44E−17	5.77E−18 ± 1.97E−17	**1.04E−21 ± 1.35E−21**	8.98E−19 ± 1.40E−18
f_{20}	6.53E+01 ± 9.28E+01	4.84E+01 ± 7.80E+01	**1.47E+01 ± 1.71E+01**	4.28E+01 ± 6.15E+01	1.68E+02 ± 1.24E+02	1.73E+01 ± 1.29E+01
f_{21}	1.55E−01 ± 6.05E−02	1.30E−01 ± 8.64E−02	**9.99E−02 ± 7.94E−02**	1.45E−01 ± 6.86E−02	1.60E−01 ± 7.54E−02	1.45E−01 ± 8.87E−02
f_{22}	4.97E−02 ± 2.22E−01	9.95E−02 ± 3.06E−01	**0.00E+00 ± 0.00E+00**	2.98E−01 ± 4.68E−01	4.97E−02 ± 2.22E−01	1.99E−01 ± 4.08E−01
f_{23}	**1.57E−32 ± 2.81E−48**	5.18E−03 ± 2.32E−02	**1.57E−32 ± 2.81E−48**	**1.57E−32 ± 2.81E−48**	**1.57E−32 ± 2.81E−48**	**1.57E−32 ± 2.81E−48**
f_{24}	1.40E−32 ± 1.23E−33	2.31E−32 ± 4.25E−32	**1.35E−32 ± 2.81E−48**	1.37E−32 ± 4.52E−34	1.41E−32 ± 1.41E−33	**1.35E−32 ± 2.81E−48**
f_{25}	2.22E−26 ± 1.82E−26	1.61E−26 ± 2.37E−26	6.42E−24 ± 5.19E−24	**1.46E−26 ± 1.15E−26**	2.28E−26 ± 3.96E−26	1.60E−26 ± 1.75E−26
f_{26}	**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**
f_{27}	2.49E−03 ± 9.85E−04	**1.78E−03 ± 5.57E−04**	2.86E−03 ± 6.10E−04	2.21E−03 ± 7.57E−04	2.34E−03 ± 7.53E−04	2.04E−03 ± 6.74E−04
f_{28}	1.61E−17 ± 1.30E−17	1.75E−17 ± 1.25E−17	5.45E−14 ± 1.06E−13	**1.36E−17 ± 8.38E−18**	1.44E−17 ± 1.38E−17	1.98E−17 ± 1.17E−17

improved by better balancing the abilities of exploration (corresponding to the DMM operator) and exploitation (corresponding to local search). Incorporation of the DMM operator and the QN local search reveals its effectiveness on difficult functions-expanded multimodal functions f_{13}, f_{14} which are shifted, rotated, non-separable and multimodal. Although relatively worse performance on f_{22} in 30D which may be improved by increasing the probability of the DMM operator, for functions f_{15}–f_{28}, DMGBDE achieves the best fitness on a majority of functions, which are all the functions except f_{15}, f_{19}, f_{22}, f_{25}, f_{27} w.r.t. 30D problems and all the functions except f_{15}, f_{20}, f_{24}, f_{27} w.r.t. 50D problems. The good performance on these functions should be attributed to the powerful local search ability of DMGBDE.

Different from the results in Table 9 which statistically analyzes the performance of algorithms on each problem, the results in Table 10 reveal the significance comparison of the overall performance on 28 test problems between DMGBDE and the other algorithms. The p_g (6.36E−05) value obtained by Friedman's test for 30D problems illustrates significant global difference exists among all mean fitness values when given a significance level $\alpha = 0.05$. The proposed algorithm DMGBDE is the best performing algorithm because it achieves the lowest ranking value 2.23E+00. The p values obtained by Holm's test for 30D problems reveal that the control algorithm (DMGBDE) outperforms the algorithms jDE, SaDE, EPSDE and DECLS with the significance level $\alpha = 0.05$, while DMGBDE is not significantly better than CoDE when given this significance level where the p value (2.84E−01) is larger than α. Similar conclusions can be also obtained from the statistical analysis for 50D problems. In general, the results in Tables 6, 7, 8, 9, 10 illustrate that DMGBDE is better or at least competitive compared with the other five algorithms in overall.

Figures 2 and 3 present the variations of objective fitness of all the considered algorithms w.r.t. the number of function evaluations. It can be seen that the objective fitness by the algorithm DMGBDE is still greatly descending on a large number of functions such as f_3, f_6, f_{19}, f_{27}, which implies possibility of large improvement on these functions in further generations.

4.4 Results of other parameter selections

It seems that more control parameters *pDEc*, *MaxITER*, *NNIMPROVE* are introduced in the proposed DMGBDE than DE algorithm; however, these parameters are fixed when running on all the 28 benchmark problems. We conduct some other experiments on 30D problems to study the influence of these parameters on the performance of DMGBDE, some of which are presented in Table 11, where the values of the parameters are perturbed a little from the proposed parameter setting. Five independent experiments are conducted, where the values of parameters *NNIMPROVE*, *MaxITER*, *N*, *pDEc*, *pDEc* are reset from $30n$, $2n$, n, 0.2,0.2 to $10n$, $3n$, $2n$, 0.1,0.5, respectively. The other parameter values are fixed when one parameter is perturbed for the experiment.

Results in Table 11 show that the proposed parameter setting is not always the best selection, for instance, DMGBDE with *NNIMPROVE* $= 30n$ achieves better performance on functions f_3, f_7, f_9 than that with the proposed parameter setting. However, the results with parameter perturbation are similar to that with proposed parameter setting, which is verified by Friedman's test (García et al. 2009) as follows. The ranking values of the algorithms with perturbed parameters and the proposed parameter setting are 3.37E+00, 3.55E+00, 3.64E+00, 3.66E+00, 3.32E+00, 3.45E+00 (the smallest ranking value corresponds to the best parameter setting), respectively, and the significance p value reflects the global difference is 9.77E−01 which is much larger than a significance level $\alpha = 0.05$ or $\alpha = 0.1$. The above statistical analysis illustrates that our algorithm is not sensitive to small perturbations of the control parameters. It is worthy to note that DMGBDE with population size N being n is significantly superior to DMGBDE with $N = 2n$ on 4 functions and inferior to DMGBDE with $N = 2n$ on 2 functions, which implies that enlargement of population size is not beneficial to DMGBDE when given function evaluations $10^4 n$. For the control parameters, *pDEc*, *MaxITER* are suggested to choose values from the interval [0.1,0.5], $[n, 4n]$, respectively, and they gradually increase as the algorithm proceeds, *NNIMPROVE* is proposed to choose values from the interval $[10n, 50n]$, and it gradually decreases as the algorithm proceeds.

5 Discussion

Although it is illustrated in Sect. 4.4 that the proposed algorithm is not sensitive to small perturbations of parameter *pDEc* which is the probability of employing classical DE mutation, *pDEc* is suggested to be self-adaptively adjusted to better balance between the convergence speed and the diversity of the population to improve the overall performance of the proposed algorithm. And other variants of DMGBDE can be set as a pool of strategies, then the ultimate DMGBDE selects one of the strategies or variants in a self-adaptive manner.

Other strategies can be adopted to further enhance the performance of the proposed algorithm. From one aspect, the DE variant DE/rand/1/bin and the corresponding parameter setting can be substituted for other variants and some reported more competitive parameter setting to devise more efficient algorithms. From another aspect, the

QN method can be replaced with other local search algorithms to boost the effectiveness of the proposed algorithm, or expand the applicability of the proposed algorithm.

6 Conclusion and future work

In this work, we propose an algorithm to solve single-objective minimization problems. In this algorithm, a gradient-based algorithm-QN method is embedded into a diversity-maintained DE variant. To minimize the risk of stagnating to a local optimum, different strategies (such as, restarting the local search only if the best is renewed, several competitive individuals in addition to the best individual are added to the local search if there is no improvement for several consecutive generations) are employed and a diversity-maintained mutation operator is introduced. We conduct our algorithm on a number of popular benchmark problems and compare that with several recently reported algorithms. Numerical results illustrate the proposed algorithm is superior to or at least competitive to the algorithms for comparison on most of the test problems.

Our algorithm is very general, by replacing the QN method with other algorithms for constrained problems, it can be expanded to solve test problems with more complicated constraints, which is our future work. The proposed algorithm needs to be further applied into practical problems to test and verify its effectiveness.

Acknowledgments The authors thank the anonymous reviewers for their helpful comments and suggestions, and also would like to thank Dr. Ponnuthurai Nagaratnam Suganthan in Nanyang Technological University, Singapore, Dr. Yong Wang in Central South University, China, Dr. Janez Brest in University of Maribor, Slovenia, and Dr. Rammohan Mallipeddi in Nanyang Technological University, Singapore, for providing the source codes of their algorithms SaDE, CoDE, jDE, and EPSDE, respectively, on the websites. This work was supported by the Chinese National Natural Science Foundation under Grant 61173060 and the Major Research Plan of National Natural Science Foundation of China under Grant 91230118.

References

Bandurski K, Kwedlo W (2010) A Lamarckian hybrid of differential evolution and conjugate gradients for neural network training. Neural Process Lett 32(1):31–44

Brest J, Greiner S, Bošković B, Mernik M, Žumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657

Cai Z, Gong W, Ling C, Zhang H (2011) A clustering-based differential evolution for global optimization. Appl Soft Comput 11(1):1363–1379

Chiang C, Lee W, Heh J (2010) A 2-opt based differential evolution for global optimization. Appl Soft Comput 10(4):1200–1207

Conn A, Gould N, Toint P (2000) Trust-region methods, vol 1. Society for Industrial Mathematics, Philadelphia, PA, USA

Das S, Konar A (2009) Automatic image pixel clustering with an improved differential evolution. Appl Soft Comput 9(1):226–236

Das S, Suganthan P (2011) Differential evolution: A survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31

Das S, Abraham A, Chakraborty U, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evol Comput 13(3):526–553

Dasgupta S, Das S, Biswas A, Abraham A (2009) On stability and convergence of the population-dynamics in differential evolution. AI Commun 22(1):1–20

Dorronsoro B, Bouvry P (2011) Improving classical and decentralized differential evolution with new mutation operator and population topologies. IEEE Trans Evol Comput 15(1):67–98

Duvvuru N, Swarup K (2011) A hybrid interior point assisted differential evolution algorithm for economic dispatch. IEEE Trans Power Syst 26(2):541–549

Fan H, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Glob Optim 27(1):105–129

Gämperle R, Müller S, Koumoutsakos A (2002) A parameter study for differential evolution. In: International conference on advances in intelligence systems, fuzzy systems, evolutionary computation, Citeseer, vol 10, pp 293–298

García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization. J Heuristics 15(6):617–644

Ghaffari-Miab M, Farmahini-Farahani A, Faraji-Dana R, Lucas C (2007) An efficient hybrid swarm intelligence-gradient optimization method for complex time Green's functions of multilayer media. Prog Electromagn Res 77:181–192

Gibbons J, Chakraborti S (2003) Nonparametric statistical inference, vol 168. Marcel Dekker, New York

Han F, Ling Q, Huang D (2010) An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks. Neural Comput Appl 19(2):255–261

Jia D, Zheng G, Khurram Khan M (2011) An effective memetic differential evolution algorithm based on chaotic local search. Inf Sci 181(15):3175–3187

Koh A (2007) Solving transportation bi-level programs with differential evolution. In: IEEE congress on evolutionary computation, IEEE, Singapore, pp 2243–2250

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

Lopez Cruz I, Van Willigenburg L, Van Straten G (2003) Optimal control of nitrate in lettuce by a hybrid approach: differential evolution and adjustable control weight gradient algorithms. Comput Electron Agric 40(1-3):179–197

Mallipeddi R, Suganthan P, Pan Q, Tasgetiren M (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11(2):1679–1696

Mandal A, Das A, Mukherjee P, Das S, Suganthan P (2011) Modified differential evolution with local search algorithm for real world optimization. In: IEEE congress on evolutionary computation, IEEE, New Orleans, LA, pp 1565–1572

Masters T, Land W (1997) A new training algorithm for the general regression neural network. In: IEEE international conference on systems, man, and cybernetics, IEEE, Orlando, FL, vol 3, pp 1990–1994

Neri F, Tirronen V (2009) Scale factor local search in differential evolution. Memet Comput 1(2):153–171

Nocedal J, Wright S (1999) Numerical optimization. Springer verlag, New York, USA

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125

Pant M, Ali M, Singh V (2008) Differential evolution with parent centric crossover. In: Second UKSIM European symposium on computer modeling and simulation, IEEE, Liverpool, pp 141–146

Plevris V, Papadrakakis M (2011) A hybrid particle swarmgradient algorithm for global structural optimization. Comput-Aided Civ Inf 26(1):48–68

Price K, Storn R, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer, New York

Qin A, Suganthan P (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE congress on evolutionary computation, IEEE 2, pp 1785–1791

Qin A, Huang V, Suganthan P (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417

Rahnamayan S, Tizhoosh H, Salama M (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79

Rönkkönen J, Kukkonen S, Price K (2005) Real-parameter optimization with differential evolution. In: The 2005 IEEE congress on evolutionary computation, IEEE, Edinburgh, Scotland, vol 1, pp 506–513

Storn R, Price K (1995) Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. International computer science institute, Berkely, CA, USA, Tech Rep TR-95-012

Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

Suganthan P, Hansen N, Liang J, Deb K, Chen Y, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Nanyang Technological University, Singapore, Tech Rep

Takahama T, Sakai S (2006) Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In: IEEE congress on evolutionary computation, IEEE, Vancouver, BC, pp 1–8

Wang H, Wu Z, Rahnamayan S (2011) Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. Soft Comput 15(11):2127–2140

Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66

Wiegand R (2004) An analysis of cooperative coevolutionary algorithms. PhD thesis, Fairfax, VA, USA

Yang Z, Yao X, He J (2008) Making a difference to differential evolution. In: Siarry P, Michalewicz Z (eds) Advances in metaheuristics for hard optimization, Springer, Berlin, pp 397–414

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

Zahara E, Kao Y, Su J (2009) Enhancing particle swarm optimization with gradient information. In: Fifth international conference on natural computation, IEEE, Tianjin, China, vol 3, pp 251–254

Zaharie D (2008) Statistical properties of differential evolution and related random search algorithms. COMPSTAT 2008, pp 473–485

Zaharie D (2009) Influence of crossover on the behavior of differential evolution algorithms. Appl Soft Comput 9(3):1126–1138

Zamuda A, Brest J, Bošković B, Žumer V (2009) Differential evolution with self-adaptation and local search for constrained multiobjective optimization. In: IEEE congress on evolutionary computation, IEEE, Trondheim, pp 195–202

Zhang J, Sanderson A (2009) Jade: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958

Zhao R, Tang W (2008) Monkey algorithm for global numerical optimization. J Uncertain Syst 2(3):165–176

Zhao S, Liang J, Suganthan P, Tasgetiren M (2008) Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: IEEE congress on evolutionary computation, IEEE, Hong Kong, pp 3845–3852