ARTICLE IN PRESS

Swarm and Evolutionary Computation xxx (2018) 1–10

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Information fusion in offspring generation: A case study in DE and EDA

Hui Fang [a], Aimin Zhou [a,*], Hu Zhang [b,**]

[a] Shanghai Key Laboratory of Multidimensional Information Processing, Department of Computer Science and Technology, East China Normal University, Shanghai 200062, China

[b] Beijing Electro-mechanical Engineering Institute, Beijing 100074, China

ARTICLE INFO

Keywords:
Differential evolution
Estimation of distribution algorithm
Hybrid algorithm
Information fusion

ABSTRACT

Both differential evolution (DE) and estimation of distribution algorithm (EDA) are popular and effective evolutionary algorithms (EAs) in solving global optimization problems. The two algorithms utilize different kinds of information for generating offspring solutions. In the former, the mutation and crossover operators use the individual information to create trial solutions, while in the later, a probabilistic model is built for sampling new trial solutions, which extracts the population distribution information. It is therefore natural to make use of both kinds of information for generating solutions. In this paper, we propose an algorithm that hybridizes DE and EDA, named as DE/GM, which utilizes both DE crossover/mutation operators and a Gaussian probabilistic model based operator for offspring generation. The basic idea is to generate some of trial solutions by the EDA operator, and to generate the rest by the DE operator. To validate the performance of DE/GM, a test suite of 13 benchmark functions is employed, and the experimental results suggest that DE/GM is promising.

1. Introduction

In this paper, we consider the following *box-constrained continuous optimization problem*.

$$\begin{cases} \min \ f(x) \\ \text{s.t.} \quad x \in \Omega \end{cases} \tag{1}$$

where $x = (x^1, x^2, \ldots, x^d)^T$ is a decision variable vector, $\Omega = [a^i, b^i]^d$ is the feasible region of the decision space, $a^i < b^i$, $a^i \in R$ and $b^i \in R$ are the lower and upper boundaries of the ith dimension in the decision space, respectively. $f(x) : \Omega \to R$ is a continuous mapping from Ω to the objective space R.

There exists a variety of methods to deal with the global optimization problems. Among them, the *evolutionary algorithms (EAs)* [1] have been attracting much attention partly due to their weak assumptions and global search ability. Different EAs have been proposed, such as *genetic algorithm (GA)* [2], *differential evolution(DE)* [3–5], *particle swarm optimization (PSO)* [6], and *estimation of distribution algorithm (EDA)* [7,8]. DE is a popular EA that creates new offspring solutions by combining several parent solutions through crossover and mutation operators [9,10]. It exhibits remarkable performance in diverse fields of science and engineering, such as cluster analysis [11], robot control [12], controller design [13], and graph theory [14]. However, in practice it has been shown that DE is sensitive to the mutation strategies [15–17] and the control parameters, i.e., the population size, the scaling factor F, and the crossover rate CR [18,19]. EDA is another popular EA paradigm that has a different mechanism from DE for offspring generation. Instead of combining several parent solutions directly, EDAs explicitly extract the distribution information of a set of parent solutions by probabilistic models and sample new solutions from the models [8,20,21]. Compared with traditional EAs, EDAs have their own advantages for dealing with hard problems when there are linkages or dependencies between decision variables [22]. However, EDAs have also been criticized for their high computational cost and low efficiency due to inadequate probabilistic models [21].

Information fusion is a natural way to improve algorithm performance for handling hard optimization problems. In the case of offspring generation, the information fusion strategies can be roughly classified into three levels, i.e., the population, individual and chromosome levels. For the population level, it means that in some generations, the population is generated by one operator while in the other generations, the population is generated by other operators. In Ref. [23], the crossover/mutation operators and a model based operator are called alternatively in different generations to generate trial solutions. For the

* Corresponding author.
** Corresponding author.
E-mail addresses: amzhou@cs.ecnu.edu.cn (A. Zhou), jxzhanghu@126.com (H. Zhang).

https://doi.org/10.1016/j.swevo.2018.02.014
Received 29 January 2017; Received in revised form 29 November 2017; Accepted 23 February 2018
Available online XXX

individual level, it means that some of the solutions in a population are generated by one operator and the others are generated by other operators. Most of hybrid EAs fall into this category. Some hybridize different operators [24,25], and some hybridize global search operators with local search operators [26]. For the chromosome level, it means that some of the elements of a chromosome are generated by one operator and the others by other operators. In Ref. [27], a hybrid algorithm, called DE/EDA, is proposed and some of the elements of the decision vector are from DE and the rest are from EDA. Following this idea in Ref. [28], some of the elements are from EDA and the others are from local search.

Algorithm 1 DE/GM Framework.

// initialize population
1 Initialize the population $pop = \{x_1, \cdots, x_N\}$ and evaluate them.
// terminate condition
2 **while** $fe < FE$ **do**
// sort the population
3 Sort pop by an ascending order of the objective values.
// local search to x_1 by mean shift
4 $y' \leftarrow MeanShift(pop)$.
// population partition
5 Partition pop into K classes$\{C_1, C_2, ..., C_K\}$.
6 **foreach** $k \in \{1, \cdots, K\}$ **do**
// EDA model building and sampling
7 $y'' \leftarrow GaussianModel(C_k)$;
// chromosome level fusion
8 Set $y_k^j = \begin{cases} y'^j & if\ rand(0,1) < P_c \\ y''^j & \text{otherwise} \end{cases}$, for $j = 1 \cdots d$.
// offspring repair
9 Repair y_k.
// environmental selection
10 Set $x_{N-k+1} = y_k$ if $f(y_k) < f(x_{N-k})$.
11 **end**
12 Set $pop' = \{x_1, \cdots, x_{N-K}\}$.
13 **foreach** $i \in \{1, \cdots, N\text{-}K\}$ **do**
// DE based offspring generation
14 $y_i \leftarrow DE(x_i, pop')$.
// offspring repair
15 Repair y_i.
// environmental selection
16 Set $x_i = y_i$ if $f(y_i) < f(x_i)$.
17 **end**
18 **end**

It is clear that DE and EDA represent two different offspring generation mechanisms. The former uses the individual location information while the latter uses the global population distribution information. To use more information in offspring generation, it is natural to combine the two search strategies as it has been done in DE/EDA [27] which falls into the chromosome level category. Following the idea of DE/EDA and our previous work [29], which uses the mean-shift method as a local search to improve the performance of DE, we propose a new algorithm in this paper. The proposed approach, called DE/GM, utilizes both the DE operator and a Gaussian probabilistic model based EDA operator for generating offspring solutions. In each generation, some of trial solutions are generated by the DE operator and the rest are generated by the EDA operator. The major differences between DE/GM and DE/EDA are as follows:

- DE/GM is based on both the individual information fusion and the chromosome information fusion while DE/EDA is only based on the chromosome information fusion.
- In DE/GM, the EDA operator is based on the Gaussian probabilistic model and a mean-shift based local search while in DE/EDA, the EDA operator is based on a univariate marginal distribution model.

The rest of this paper is organized as follows. In Section 2, the DE/GM algorithm framework is introduced, and the details of the EDA operator are presented. The experimental results and analysis are given in Section 3. Finally, this paper is concluded in Section 4.

2. Proposed algorithm

2.1. DE/GM framework

As mentioned above, the basic idea of our approach is hybridizing DE and EDA for offspring generation in both the individual level and the chromosome level. In each generation, the population is sorted and the DE operator is applied to the best individuals. The EDA part works as follows: firstly, the population is partitioned into several clusters, then for each cluster a multivariate Gaussian probabilistic model is built and a trial solution is sampled, after that the trial solutions are combined with a solution, which is an improved one of the best solution by a mean-shift based local search [29], to form offspring solutions, finally the offspring solutions will replace the worst solutions in the current population.

In each generation, DE/GM maintains

- a set of N solutions $\{x_1, x_2, \dots, x_N\}$,
- their objective values $\{f(x_1), f(x_2), \dots, f(x_N)\}$.

The algorithm framework of the proposed DE/GM is shown in Algorithm 1. We would like to make some comments on the algorithm as follows.

- *Population Initialization:* In *Line 1*, the initial population is uniformly and randomly sampled from the feasible region Ω.
- *Termination Condition:* In *Line 2*, the algorithm stops when the number of function evaluations *fe* reaches the preset maximum number *FE*.
- *Offspring Generation:* A hybrid strategy is applied to generate offspring solutions. The EDA operator is used in *Lines 3–11*: first, the best solution, i.e., x_1 in the sorted population, is improved by the mean-shift based local search to obtain a candidate solution y', then for each cluster a Gaussian probabilistic model is built and a candidate solution y'' is sampled from the model, finally the elements of the two candidate solutions are combined to form an offspring solution y^k where the ratio is controlled by the parameter P_c. The DE operator is used for the best $N - K$ solutions in *Line 14*.
- *Environmental Selection:* The selection is based on the objective values. In *Line 10*, the solutions generated by the EDA operator try to replace the worst solutions in each generation. In *Line 16*, the solutions generated by the DE operator try to replace the corresponding parent solutions.
- *Offspring Repair:* The newly generated offspring solution y might be infeasible, and it is repaired in *Lines 9* and *15* as follows.

$$y_j = \begin{cases} rand(a_j, x_j) & if\ \ y_j < a_j \\ rand(x_j, b_j) & if\ \ y_j > b_j \\ y_j & \text{otherwise} \end{cases} \tag{2}$$

where $j = 1, \dots, n$, $rand(s, t)$ returns a random number from $[s, t]$, and x is the corresponding parent of y.

It should be noted that the offspring combination strategy in *Line 8* is the same as in Refs. [27,30], which is a chromosome level information fusion strategy. The major part uses an individual level information fusion strategy. Some details of DE/MG will be discussed shortly in the following sections.

2.2. Mean-shift based local search

Mean-shift [31] is a non-parametric feature-space analysis technique for locating the maxima of a density function, which can be applied to optimization [32–34]. It has also been applied to accelerate EAs [29]. In this paper, we use a mean-shift based local search to improve the best solution found so far, and it works as in Algorithm 2. More details of the approach are referred to [29].

Algorithm 2 $y' \leftarrow MeanShift(pop)$.

1 For $j = 1, \ldots, d$, locate the population boundaries

$$u^j \leftarrow \max_{i=1,\ldots,N} x_i^j$$

$$v^j \leftarrow \min_{i=1,\ldots,N} x_i^j$$

2 Calculate the 'bandwidth' as

$$h = \sqrt{\frac{1}{d}\sum_{j=1}^{d}\left(u^j - v^j\right)^2}.$$

3 For $x_1 \in pop$, estimate a new position

$$y' = \frac{\sum_{i=1}^{N} x_i g\left(\left\|\frac{x_1 - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{N} g\left(\left\|\frac{x_1 - x_i}{h}\right\|^2\right)}$$

where $g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ is a Gaussian kernel.

2.3. Gaussian probabilistic model based operator

In order to utilize the distribution information efficiently, a Gaussian probabilistic model based operator is proposed. The main procedure is as follows. First, the population is partitioned into several clusters by K-means [35]. Then, a Gaussian probabilistic model is built in each cluster and a candidate offspring solution is sampled from the model. Finally, the candidate offspring solutions are combined with the candidate offspring generated by the mean-shift local search to form the offspring solutions. The Gaussian model building and sampling is illustrated in Algorithm 3.

Algorithm 3 $y'' \leftarrow GaussianSample(C_k)$.

1 Calculate the mean value of the cluster as

$$\mu = \frac{1}{|C_k|}\sum_{x \in C_k} x.$$

2 Calculate the covariance matrix Σ as

$$\Sigma_{i,j} = \frac{1}{|C_k|}\sum_{x \in C_k}(x_i - \mu_i)(x_j - \mu_j).$$

where $i, j = 1, \ldots, d$.

3 Sample a candidate solution y'' from $N(\mu, \Sigma)$.

2.4. Differential evolution based operator

DE uses crossover and mutation operators to generate offspring solutions [36]. A variety of DE variants have been proposed [10,37,38]. In this paper, we use the following scheme, which is from Ref. [38], to generate solutions where F and CR are two control parameters.

Algorithm 4 $y \leftarrow DE(x, pop')$.

1 Randomly select control parameters $< F, CR >$ from

$\{< 1.0, 0.1 >, < 1.0, 0.9 >, < 0.8, 0.2 >\}$.

2 Randomly select parents from pop' such that

$x^{r_1} \neq x^{r_2} \neq x^{r_3} \neq x$.

3 Generate a vector v via mutation as

$v = x_{r1} + F \cdot (x_{r2} - x_{r3})$.

4 Generate the offspring solution via crossover as

$$y_j = \begin{cases} x_j & if \quad rand(0.0, 1.0) < CR \;\; or \;\; j = j_{rnd} \\ v_j & \text{otherwise} \end{cases}$$

where $j = 1, \ldots, d$, and $j_{rnd} \in \{1, \ldots, d\}$ is a random number.

3. Experimental results

In this section, we investigate the performance of DE/MG. The fist 13 instances, denoted as f_1-f_{13} and with different characteristics, from the YYL test suite [39] are used for this purpose. The original DE [3], DE/EDA [27] and MSDE [29] are used for the comparison study.

3.1. Parameter settings

The parameters in the experiment study are as follows:

– The population size is $N = 100$ for all algorithms, and the dimension of the test instance is $d = 30$ for all test instances.
– All algorithms are executed independently for 30 runs and stopped after $FE = 300,000$ function evaluations.
– In DE/MG, the number of clusters is $K = 10$ and the probability to use the mean-shift information is $P_c = 0.2$, and the parameters in the Gaussian kernel are $\sigma = 1$ and $\mu = 0$.

The Wilcoxon's rank sum test at a 5% significance level is used to do comparison between the different algorithms. In the following tables, $\sim$, $+$ and $-$ indicates a solution obtained by an algorithm is similar, better, or worse than that obtained by a compared algorithm, respectively. The values with bold type denote the best results. All the algorithms are implemented in Matlab R2010b and executed in Lenovo Thinkpad E430 with i5-3210M CPU @ 2.50 GHz, 6.00 GB RAM, and Windows 7.

3.2. Roles of DE/GM components

DE/GM has two main components, i.e., the DE operator and the Gaussian probabilistic model based operator, for generating trial solutions. This section investigates their effects on the performance of DE/GM.

We denote d, gm as the two different operators, i.e., the DE operator and the Gaussian probabilistic model based operator. The following combinations of the two components are studied:

– DE/GM$_d$: DE/GM with the DE operator but without the Gaussian probabilistic model based operator. The DE operator generates trial solutions for all the parent solutions.
– DE/GM$_{gm}$: DE/GM with the Gaussian probabilistic model based operator but without the DE operator. The Gaussian probabilistic model samples trail solutions for all the parent solutions in each cluster.

Table 1
Mean, std., median values of the results obtained by the three comparison algorithms after 300,000 function evaluations over 30 runs for all the test instances.

	DE/GM$_d$			DE/GM$_{gm}$			DE/GM$_{d,gm}$		
	Mean	std.	Median	Mean	std.	Median	Mean	std.	Median
f_1	2.22e-09 [2](−)	7.91e-10	2.03e-09	1.23e+03 [3](−)	6.17e+02	1.13e+03	**6.09e-70** [1]	6.72e-70	4.14e-70
f_2	9.87e-07 [2](−)	1.57e-07	9.82e-07	9.62e+00 [3](−)	2.37e+00	9.67e+00	**5.29e-38** [1]	3.27e-38	4.71e-38
f_3	1.39e+04 [3](−)	2.42e+03	1.40e+04	3.75e+03 [2](−)	1.45e+03	3.52e+03	**4.11e-05** [1]	8.12e-05	2.32e-05
f_4	3.41e+00 [2](−)	2.87e-01	3.37e+00	2.12e+01 [3](−)	4.22e+00	2.12e+01	**4.84e-19** [1]	1.15e-18	2.39e-19
f_5	4.44e+01 [2](−)	1.09e+01	4.51e+01	2.57e+05 [3](−)	2.01e+05	1.75e+05	**1.92e+00** [1]	1.37e+00	1.81e+00
f_6	**0.00e+00** [1](∼)	0.00e+00	0.00e+00	1.05e+03 [2](−)	4.23e+02	9.97e+02	**0.00e+00** [1]	0.00e+00	0.00e+00
f_7	5.43e-01 [3](−)	1.16e-01	5.36e-01	**5.97e-02** [1](+)	3.57e-02	4.97e-02	8.32e-02 [2]	4.02e-02	7.06e-02
f_8	5.64e-08 [2](−)	2.03e-08	5.13e-08	5.95e+03 [3](−)	8.85e+02	6.05e+03	**0.00e+00** [1]	0.00e+00	0.00e+00
f_9	1.17e-03 [2](−)	4.82e-04	1.06e-03	6.23e+01 [3](−)	1.57e+01	6.08e+01	**0.00e+00** [1]	0.00e+00	0.00e+00
f_{10}	1.13e-05 [2](−)	1.69e-06	1.11e-05	7.29e+00 [3](−)	1.11e+00	7.14e+00	**4.44e-15** [1]	0.00e+00	4.44e-15
f_{11}	7.28e-08 [2](−)	4.36e-08	5.74e-08	9.42e+00 [3](−)	3.51e+00	9.28e+00	**0.00e+00** [1]	0.00e+00	0.00e+00
f_{12}	4.34e-10 [2](−)	1.59e-10	3.91e-10	5.29e+02 [3](−)	1.60e+03	1.68e+01	**1.57e-32** [1]	5.57e-48	1.57e-32
f_{13}	1.18e-09 [2](−)	4.41e-10	1.15e-09	1.90e+05 [3](−)	4.34e+05	4.36e+04	**1.35e-32** [1]	5.57e-48	1.35e-32
rank	2.077			2.692			1.077		
+/−/∼	0/12/1			1/12/0					

Table 2
Mean±std. values obtained by the two comparison algorithms after 300,000 function evaluations over 30 runs for all the test instances.

	DE/GM	DE/GM$_{nms}$
f_1	**6.09e-70**±6.72e-70	3.79e-51±3.01e-51 (−)
f_2	**5.29e-38**±3.27e-38	1.49e-26±6.09e-27 (−)
f_3	**4.11e-05**±8.12e-05	1.08e-05±2.49e-05 (+)
f_4	**4.84e-19**±1.15e-18	2.21e-13±8.51e-14 (−)
f_5	1.92e+00±1.37e+00	**1.28e+00**±9.71e-01 (+)
f_6	**0.00e+00**±0.00e+00	**0.00e+00**±0.00e+00 (∼)
f_7	**8.32e-02**±4.02e-02	1.31e-01±4.71e-02 (−)
f_8	**0.00e+00**±0.00e+00	**0.00e+00**±0.00e+00 (∼)
f_9	**0.00e+00**±0.00e+00	**0.00e+00**±0.00e+00 (∼)
f_{10}	**4.44e-15**±0.00e+00	**4.44e-15**±0.00e+00 (∼)
f_{11}	**0.00e+00**±0.00e+00	2.47e-04±1.35e-03 (−)
f_{12}	**1.57e-32**±5.57e-48	**1.57e-32**±5.57e-48 (∼)
f_{13}	**1.35e-32**±5.57e-48	**1.35e-32**±5.57e-48 (∼)
+/−/∼		2/5/6

Table 3
Times of different rank values obtained by DE/GM with different K and d on all the 13 test instances.

d	K	Rank				
		1	2	3	4	Mean
30	2	4	4	2	3	2.31
	5	7	2	3	1	1.85
	10	10	2	1	0	1.31
	15	10	2	0	1	1.39
50	2	2	4	3	4	2.69
	5	7	1	4	1	1.92
	10	7	5	1	0	1.54
	15	10	1	0	2	1.54
100	2	2	2	3	6	3.00
	5	3	2	6	2	2.54
	10	6	5	2	0	1.69
	15	7	3	0	3	1.92

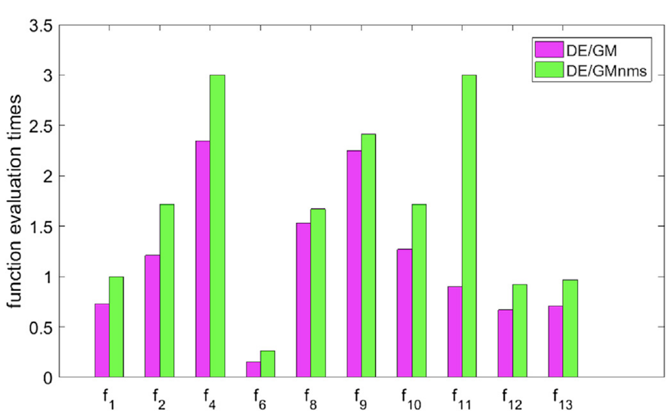


Fig. 1. Average numbers of function evaluations to reach the goal $f < 10^{-14}$ for DE/GM and DE/GM$_{nms}$.

– DE/GM$_{d,gm}$: DE/GM with the Gaussian probabilistic model based operator and the DE operator, which is actually DE/GM itself.

The instances $f_1 - f_{13}$ are used for the study, and the parameter settings are same as in Section 3.1. The mean, standard deviation, and median values of the runs on the 13 instances obtained by the algorithms are shown in Table 1.

It shows that DE/GM$_{d,gm}$ obtains the best results on $f_1 - f_6$, and $f_8 - f_{13}$, whereas DE/GM$_{gm}$ fails on all the test instances except on f_7. DE/GM$_d$ works well on $f_1 - f_2$ and has the same result on f_6 as DE/GM$_{d,gm}$. It is obvious that DE/GM$_{d,gm}$ outperforms DE/GM$_{gm}$ on all the functions except on f_7 although they both fail on this problem.

The statistical results suggest that the DE operator plays an important role in generating solutions in DE/GM. DE/GM$_{gm}$ performs not well on all the functions, which indicates that the Gaussian probabilistic model based operator itself can not generate high quality offspring solutions. The reason might be that generating too many solutions in each cluster by the Gaussian probabilistic model based operator might lead to premature and the solutions are easy to fall into local optimum. The comparison between DE/GM$_d$ and DE/GM$_{d,gm}$ shows that DE/GM$_{d,gm}$ is better than DE/GM$_d$ on most test instances except on f_6. This suggests that, the Gaussian probabilistic model based operator can create trial solutions that guides the population to the optimal solution efficiently.

To conclude, the DE operator and the Gaussian probabilistic model based operator both play important roles in DE/GM and can work collaboratively to achieve good results. The major reason might be, as discussed in the previous sections, that the two operators use different information in offspring generation, and a fusion of the two operators can help the algorithm to make use of the information from different angles and thus to generate high quality offspring solutions.

3.3. Contribution of mean-shift method

In the procedure of the Gaussian probabilistic model based operator, we improve the generated solution by combining with the solution created by a mean-shift based local search. As shown in Ref. [29], the mean-shift based search plays a guidance role in searching optimal solutions. In this section, we denote DE/GM without the mean shift method as DE/GM$_{nms}$.

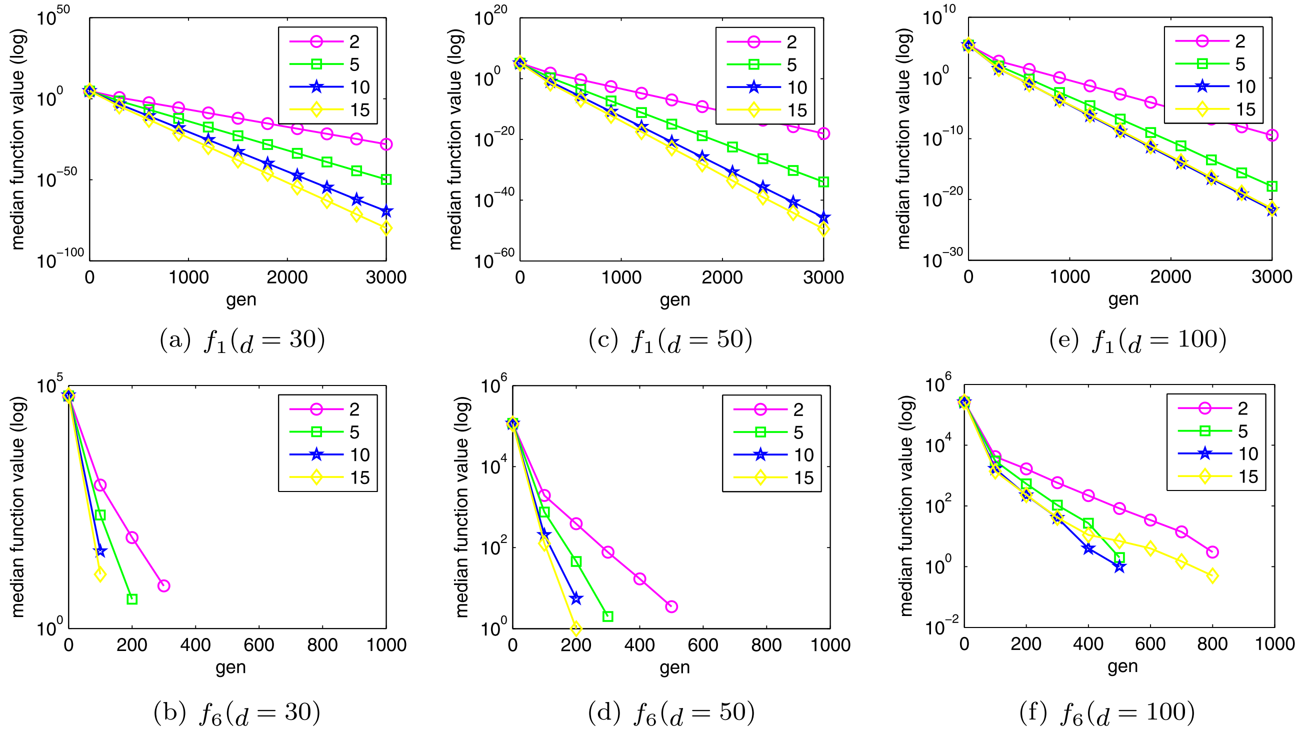


Fig. 2. Median function values versus function evaluations obtained by DE/GM with different settings of K and d on f_1 and f_6.

Table 4
Times of different rank values obtained by DE/GM with different P_c and d on all the 13 test instances.

d	P_c	Rank				
		1	2	3	4	Mean
30	0.1	7	2	2	2	1.92
	0.2	7	2	3	1	1.85
	0.5	9	3	1	0	1.38
	0.8	9	1	0	3	1.77
50	0.1	8	1	0	4	2.00
	0.2	7	3	3	0	1.69
	0.5	7	3	3	0	1.69
	0.8	7	2	1	3	2.00
100	0.1	5	4	2	2	2.08
	0.2	7	3	3	0	1.69
	0.5	2	4	7	0	2.38
	0.8	4	1	0	8	2.92

The comparison results of the two algorithms are shown in Table 2. It is clear that DE/GM outperforms DE/GM$_{nms}$ on $f_1 - f_4$, f_7 and f_{11}. DE/GM$_{nms}$ is better than DE/GM on f_5 although they both fail on f_5. The two algorithms obtain similar results on f_6, $f_8 - f_{10}$, f_{12} and f_{13}. Fig. 1 plots the average numbers of function evaluations required to achieve $f < 10^{-14}$ on $f_1 - f_{13}$, except on f_3, f_5 and f_7, by the two algorithms. It is clear that DE/GM converges faster than DE/GM$_{nms}$. Moreover, DE/GM$_{nms}$ does not achieve the goals on $f_3 - f_5$, f_7 and f_{11}. Hence, combining the solutions created by the probabilistic model with the solutions generated from the mean-shift method can help to find the optimal solution quickly in a large extent.

3.4. Sensitivity to control parameters

In this section, we investigate the sensitivity of DE/GM to the clustering number, K, and the percentage of the solution created by mean-shift search method used in the solution generated by Gaussian model method, P_c.

3.4.1. Sensitivity to K

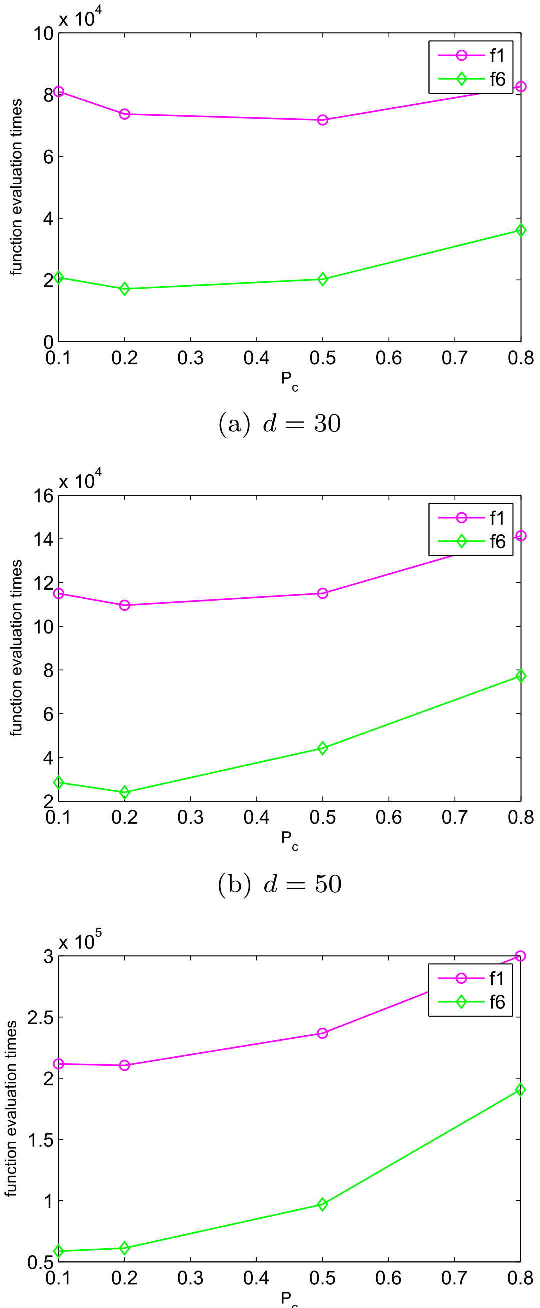
In K-means, K is denotes the number of clusters that will be produced. In general, the upper bound of the number of clusters can be set to be $\sqrt{N}$ as suggested in Ref. [40]. In DE/GM, K also decides how many solutions would be generated by the Gaussian model based operator. DE/GM with $K = 2$, 5, 10, and 15 are tested on the 13 test instances in this section.

The rank of median value of the results obtained by DE/GM under different K with $d = 30$, 50, and 100 are shown in Table 3. It is clear that DE/GM with $K = 10$ obtains the best mean results among all the settings for all the search dimensions. DE/GM with $K = 2$ performs the worst. This indicates DE/GM with small values of K may fail to detect the statistical information of the population, while DE/GM with big values of K may not be able to balance the computational costs assigned to the two kinds of operators. It can be concluded that $K = 10$ might be a good choice for DE/GM.

To visualize the convergence performance, Fig. 2 plots the median function values versus function evaluation obtained by DE/GM with different settings of K and d on f_1 and f_6. From Fig. 2, we can see that DE/GM with $K = 10$ and 15 outperform DE/GM with other K settings. However, DE/GM with $K = 15$ is not stable as DE/GM with $K = 10$ on f_6 with $d = 100$. It can be concluded that DE/GM with $K = 10$ is more stable.

3.4.2. Sensitivity to P_c

This section studies the effect of P_c, the percentage of the solution generated by the mean-shift search operator used for the solution created by Gaussian model reproduction. In the experiments, P_c is set to be 0.1, 0.2, 0.5, and 0.8, the variable dimension d is set to be 30, 50, and 100, and the other parameters are the same as in Section 3.1.



(a) $d = 30$

(b) $d = 50$

(c) $d = 100$

Fig. 3. Median function evaluations used to achieve the goal $f < 10^{-14}$ for DE/GM with different settings of P_c and d on f_1 and f_6.

The rank of median value of the results obtained by DE/GM under different P_c with $d = 30$, 50, and 100 are shown in Table 4. From the table, we can see that when the variable dimension is low, i.e., $d = 30$, DE/GM with $P_c = 0.5$ obtains the best results. As the variable dimension increases, DE/GM works the best with $P_c = 0.2$ or 0.5 for $d = 50$, and $P_c = 0.2$ for $d = 100$.

To further investigate the influence of P_c, Fig. 3 plots the average number of function evaluations used to achieve the goal $f < 10^{-14}$ with different control parameter P_c when $d = 30$, 50, and 100 on f_1 and f_6. It is shown that DE/GM with different settings of P_c has slightly effect on the performance in terms of solution quality when $d = 30$. The performance of $P_c = 0.8$ is sensitive to the dimension.

3.5. Scalability to variable size

In the following, we study the influence of the problem dimensionality on the performance of DE/GM and the compared algorithms.

Firstly, DE/GM is applied to the 13 test instances with variable dimensions of $d = 10$, 20, 30, 50, and 100. The other control parameters are the same as in Section 3.1. Table 5 presents the statistical results. It is clear that as expected, the performance of DE/GM decreases as the dimensionality increases. However, on some instances, DE/GM can still achieve acceptable results when the dimension increases.

Secondly, all the four comparison algorithms are applied to the 13 test instances with variable dimensions of $d = 30$, 50, and 100. The other control parameters are the same as in Section 3.1. For each instance with each dimension, the mean rank values of the results obtained are calculated, and are shown in Fig. 4. From the figure, we can see that the rank values of DE/GM are better than those of the other three algorithms for all the three variable dimensions. DE always performs the worst in the experiments. MSDE obtains similar rank value to DE/GM for problems with $d = 100$, but performs worse than DE/GM for problems with $d = 30$ and 50. DE/EDA does not work well as DE/GM for all the variable dimensions.

3.6. Comparison results and analysis

In this section, we compare DE/GM to other three algorithms, the original DE, DE/EDA, and MSDE that we proposed before. As we know, DE creates new solutions by crossing a solution from the current population and a solution obtained by the DE mutation. In DE/EDA, one part of a trial solution generated comes from the DE mutation and the rest part is sampled by EDA. The details of DE/EDA can be found in Ref. [27]. In MSDE, the population are created by the DE operator and the mean-shift operator.

Table 6 shows the mean and standard deviation of the results obtained by the four algorithms after 300,000 function evaluations over 30 independent runs on the 13 test instances. The rank represents the ranking of the four algorithm and the best result is given the value 1. The Wilcoxon's rank sum test is applied to statically compare the function values obtained by DE/GM and other algorithms. The results of Wilcoxon's rank sum test and rank are shown in Table 6. Fig. 5 shows the mean function values obtained by the four algorithms versus the function evaluations. Table 7 presents The average function evaluations and the number of successful run to achieve $f < 10^{-4}$ for the four algorithms. The analyses of the statistical results are as follows.

3.6.1. DE/GM vs. DE

In Table 6, according to the Wilcoxon's rank sum test on the solutions found by these algorithms, " $\sim$ ", " $+$ ", and " $-$ " denote the results obtained by DE and DE/EDA are similar to, better than, or worse than that obtained by DE/GM. DE/GM outperforms DE in most functions after 300,000 function evaluations. DE/GM converges to the global optimal on f_6, f_8, f_9, f_{11} and DE converges to the global optimal on f_6. The statistical results of the two algorithms are consistent with each other on f_6 and f_7. From the view of the average rank, DE/GM is obviously better than DE in general. The performances on f_{12} and f_{13} are similar to each other. As is shown in Fig. 5 and Table 7, DE/GM converges faster than DE on most test instances. It can be seen that DE/GM

Table 5
Mean±Std. values of the results obtained by the DE/GM algorithms with different variable dimensions on all the 13 test instances.

d	10	20	30	50	100
f_1	7.38e-153±1.18e-152	3.32e-95±4.26e-95	6.09e-70±6.72e-70	2.23e-46±3.19e-46	7.23e-23±7.23e-23
f_2	2.26e-83±2.32e-83	3.51e-51±1.92e-51	5.29e-38±3.27e-38	6.36e-27±2.93e-27	2.97e-15±2.97e-15
f_3	8.96e-50±1.24e-49	3.27e-14±4.68e-14	4.11e-05±8.12e-05	1.69e+00±7.32e-01	3.62e+02±3.62e+02
f_4	1.35e-46±1.02e-46	6.81e-27±4.26e-27	4.84e-19±1.15e-18	8.31e-01±7.83e-01	8.07e+00±8.07e+00
f_5	0.00e+00±0.00e+00	6.44e-10±1.46e-09	1.92e+00±1.37e+00	3.60e+01±1.07e+01	1.47e+02±1.47e+02
f_6	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_7	6.28e-02±3.09e-02	6.94e-02±3.43e-02	8.32e-02±4.02e-02	1.14e-01±3.70e-02	1.37e-01±1.37e-01
f_8	0.00e+00±0.00e+00	0.00e+00±0.00e+00	0.00e+00±0.00e+00	1.84e-11±1.33e-12	1.29e+04±1.29e+04
f_9	0.00e+00±0.00e+00	0.00e+00±0.00e+00	0.00e+00±0.00e+00	3.40e+01±3.45e+00	3.49e+02±3.49e+02
f_{10}	1.01e-15±6.49e-16	4.44e-15±0.00e+00	4.44e-15±0.00e+00	4.44e-15±0.00e+00	6.35e-02±6.35e-02
f_{11}	0.00e+00±0.00e+00	2.47e-04±1.35e-03	0.00e+00±0.00e+00	2.47e-04±1.35e-03	1.97e-03±1.97e-03
f_{12}	4.71e-32±1.67e-47	2.36e-32±8.35e-48	1.57e-32±5.57e-48	2.07e-03±1.14e-02	1.04e-03±1.04e-03
f_{13}	1.35e-32±5.57e-48	1.35e-32±5.57e-48	1.35e-32±5.57e-48	1.35e-32±5.57e-48	2.20e-03±2.20e-03

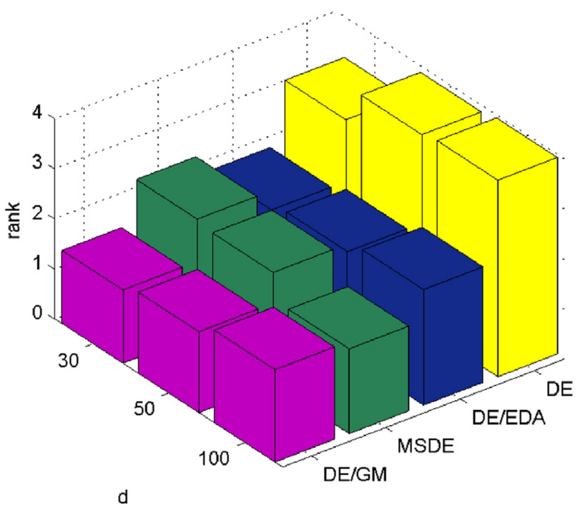


Fig. 4. Ranks of the results obtained by four algorithms with $d = 30$, 50, and 100 on all the test instances.

reaches the convergent stage earlier than DE on $f_1 - f_{13}$. In DE/GM, there are obvious downward trends of mean function value on $f_1 - f_6$, $f_8 - f_{13}$. It is evident that DE/GM converges quickly in its early stage and gets better results than DE on most test instances. The reason that DE/GM converges faster than DE/EDA could be that the Gaussian probabilistic model based operator plays a significant role at the convergence rate in searching for a local optimal solution.

3.6.2. DE/GM vs. DE/EDA

DE/EDA is an improved DE algorithm based on EDA. Both DE/GM and DE/EDA apply probability model techniques to DE for offspring reproduction. From Table 6, It can be seen that DE/GM has a better performance than DE/EDA on $f_1, f_2, f_4, f_8 - f_9$ and two algorithms are consistent with each other on $f_6, f_{10} - f_{13}$. DE/EDA outperforms DE/GM on f_3 and f_5. The average rank of DE/GM is better than that of DE/EDA. It is obvious from Fig. 5 that DE/GM converges faster than DE/EDA on $f_2, f_4, f_8 - f_9, f_{11}$ and the speed of convergence is similar to DE/EDA on $f_1, f_6, f_{10}, f_{12} - f_{13}$.

3.6.3. DE/GM vs. MSDE

DE/GM extends the work in MSDE by adding the EDA operator to the search process, which combines both the global population information and the individual information. The comparison of the two algorithms is shown in Table 6. DE/GM outperforms MSDE on most test instances except f_3 and f_7. The statistical results of the two algorithms are consistent with each other on f_6 and f_{11}. According to the average rank, DE/GM is better than MSDE. It is obvious from Fig. 5 that DE/GM converges faster than MSDE on f_1, f_2, f_4, f_6, f_8, f_9, f_{11} and f_{12}. The reason might be that the EDA operator speed up the search. The speed of convergence of the two algorithms is similar on f_3, f_5 and f_{10}.

Table 6
Mean±Std. values of the results obtained by the four comparison algorithms after 3.0×10^5 function evaluations over 30 runs on all the 13 test instances.

	DE/GM	DE/EDA	DE	MSDE
f_1	**6.09e-70±6.72e-70** [1]	9.52e-70±2.29e-69 [2](−)	2.22e-09±7.91e-10 [4](−)	8.18e-20±4.41e-20 [3](−)
f_2	**5.29e-38±3.27e-38** [1]	2.45e-33±3.13e-33 [2](−)	9.87e-07±1.57e-07 [4](−)	1.34e-10±4.39e-11 [3](−)
f_3	4.11e-05±8.12e-05 [3]	**1.94e-15±3.22e-15** [1](+)	1.39e+04±2.42e+03 [4](−)	7.82e-06±8.75e-06 [2](−)
f_4	**4.84e-19±1.15e-18** [1]	4.87e-08±8.01e-08 [2](−)	3.41e+00±2.87e-01 [4](−)	8.43e-07±2.74e-07 [3](−)
f_5	1.92e+00±1.37e+00 [3]	**1.33e-01±7.28e-01** [1](+)	4.44e+01±1.09e+01 [4](−)	1.36e+01±5.04e-01 [2](−)
f_6	**0.00e+00±0.00e+00** [1]	**0.00e+00±0.00e+00** [1](~)	**0.00e+00±0.00e+00** [1](~)	**0.00e+00±0.00e+00** [1](~)
f_7	8.32e-02±4.02e-02 [3]	**2.20e-03±6.37e-04** [1](+)	5.43e-01±1.16e-01 [4](−)	1.18e-02±1.26e-02 [2](−)
f_8	**0.00e+00±0.00e+00** [1]	3.75e+02±2.43e+02 [4](−)	5.64e-08±2.03e-08 [2](−)	4.81e-03±9.04e-03 [3](−)
f_9	**0.00e+00±0.00e+00** [1]	6.53e+01±4.95e+01 [4](−)	1.17e-03±4.82e-04 [2](−)	1.05e+01±9.10e+00 [3](−)
f_{10}	**4.44e-15±0.00e+00** [1]	4.91e-15±1.23e-15 [2](−)	1.13e-05±1.69e-06 [4](−)	7.12e-11±2.60e-11 [3](−)
f_{11}	**0.00e+00±0.00e+00** [1]	1.07e-03±2.80e-03 [3](−)	7.28e-08±4.36e-08 [2](−)	**0.00e+00±0.00e+00** [1](~)
f_{12}	**1.57e-32±5.57e-48** [1]	**1.57e-32±5.57e-48** [1](~)	4.34e-10±1.59e-10 [3](−)	6.66e-22±4.15e-22 [2](−)
f_{13}	**1.35e-32±5.57e-48** [1]	**1.35e-32±5.57e-48** [1](~)	1.18e-09±4.41e-10 [3](−)	1.38e-20±1.36e-20 [2](−)
Rank	1.462	1.923	3.154	2.308
+/−/~		3/7/3	0/12/1	0/11/2

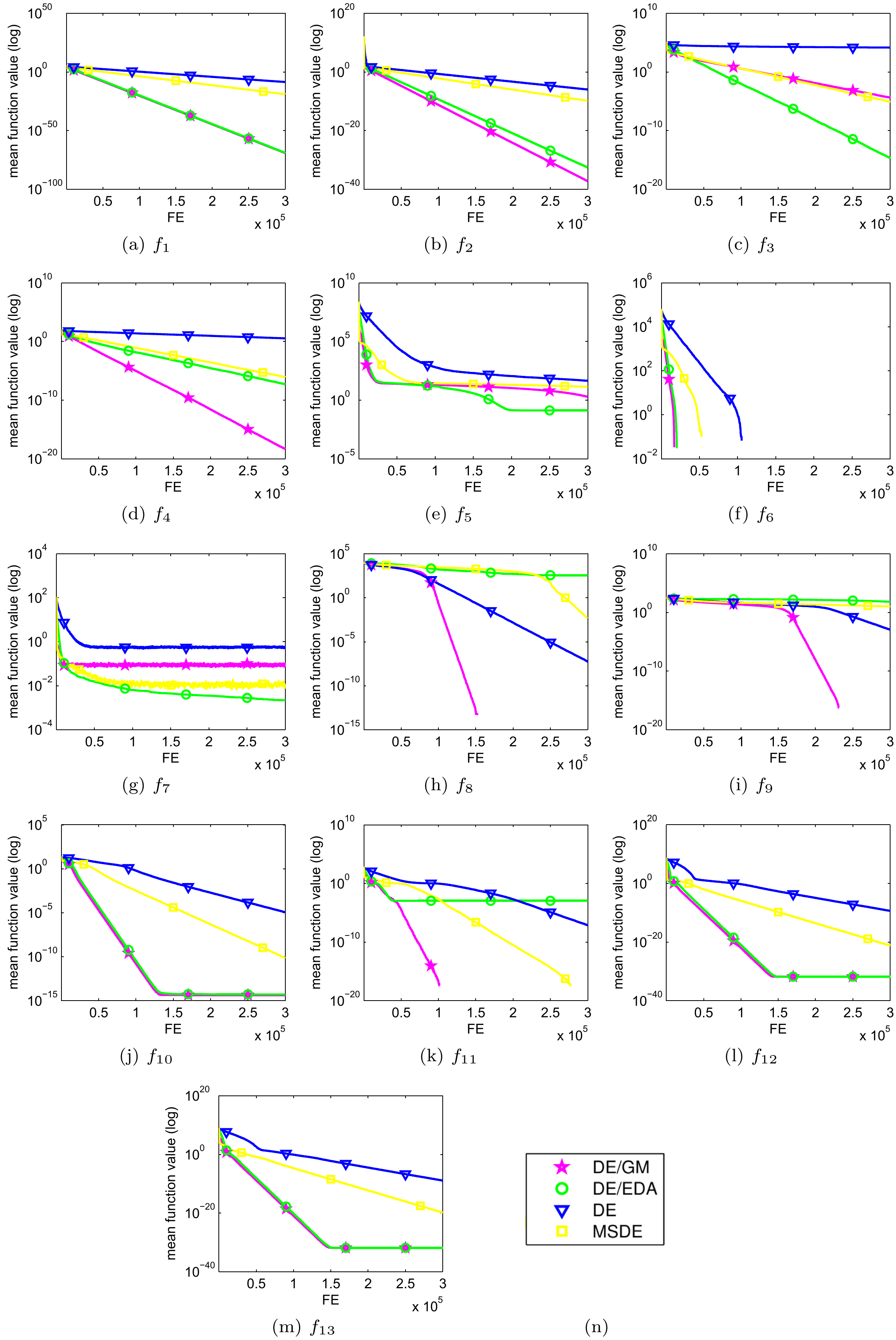


Fig. 5. Mean function value of the best solutions obtained by the four algorithms versus function evaluation on all the 13 test instances.

Table 7
The average function evaluations and number of successful runs to achieve $f < 1.0 \times 10^{-4}$ on all the test instances for DE/GM, DE/EDA, DE and MSDE.

	DE/GM	DE/EDA	DE	MSDE
f_1	0.33 (30)	0.35 (30)	1.95 (30)	1.03 (30)
f_2	0.44 (30)	0.53 (30)	2.23 (30)	1.46 (30)
f_3	2.84 (30)	1.32 (29)	NA (0)	2.60 (30)
f_4	0.84 (30)	1.93 (30)	NA (0)	2.16 (30)
f_5	NA (0)	2.04 (28)	NA (0)	NA (0)
f_6	0.17 (30)	0.21 (30)	1.06 (30)	0.53 (30)
f_7	NA (0)	NA (0)	NA (0)	NA (0)
f_8	1.14 (30)	NA (0)	2.27 (30)	NA (0)
f_9	1.83 (30)	NA (0)	NA (0)	NA (0)
f_{10}	0.44 (30)	0.47 (30)	2.57 (30)	1.40 (30)
f_{11}	0.47 (30)	0.37 (28)	2.30 (30)	1.15 (30)
f_{12}	0.25 (30)	0.30 (30)	1.79 (30)	0.74 (30)
f_{13}	0.29 (30)	0.33 (30)	1.89 (30)	0.91 (30)

Table 8
Average CPU time (seconds) used by DE/GM, DE/EDA, DE, MSDE on all the 13 test instances.

	DE/GM	DE/EDA	DE	MSDE
f_1	31.83	0.90	6.34	5.14
f_2	32.55	0.91	6.63	4.77
f_3	32.37	0.86	10.13	4.81
f_4	32.28	0.87	7.37	4.77
f_5	33.78	0.91	8.12	4.83
f_6	32.84	0.94	7.48	4.83
f_7	36.38	2.21	11.02	7.09
f_8	35.24	1.16	8.67	7.37
f_9	34.26	1.10	8.04	5.05
f_{10}	34.79	1.02	8.88	4.94
f_{11}	34.82	1.04	8.89	4.98
f_{12}	58.41	1.94	12.87	6.10
f_{13}	36.42	1.93	12.78	6.16

3.6.4. Time used in search

The CPU time is another major concern when applying EAs. The average time used by the four algorithms after 30 runs are recorded in Table 8. It can be seen that DE/GM requires more CPU time than the other three algorithms. The time used by DE/GM is 1.92–33.48 times of those used by other methods. Compared with DE/EDA, DE/GM may take much time in the process of clustering.

3.7. Combining Gaussian probabilistic model based operator with other evolutionary algorithms

In DE/GM, the DE operator and the Gaussian probabilistic model based operator are both used to generate trial solutions for each generation. Considering the characteristic and the effect of the Gaussian probabilistic model based operator, we investigate the feasibility and usability of applying the Gaussian probabilistic model based operator to other evolutionary algorithms with their own operators in this section. According to the idea of DE/GM, we apply the Gaussian probabilistic model based operator to JADE [37] and CoDE [38], which are two classical and efficient evolutionary algorithms. The main procedure is that some solutions are generated by the operator from the current evolutionary algorithm and the rest solutions are created by the Gaussian probabilistic model based operator. Denoted the two hybrid algorithms as JADE/GM and CoDE/GM. Then comparing them with the initial algorithm respectively. The parameter settings are as follows.

- JADE: the parameters N = 100, p = 0.05, c = 0.1, F = 0.5 and CR = 0.9 as suggested in [37].
- CoDE: the three control parameter settings are [F = 1.0, Cr = 0.1], [F = 1.0, Cr = 0.9], [F = 0.8, Cr = 0.2]. The three selected trial vector generation strategies are rand/1/bin, rand/2/bin, current-to-rand/1. The details are described in Ref. [41].

Table 9 shows the statistical results of CoDE, CoDE/GM, JADE and JADE/GM on the 13 test instances respectively.

It can be seen that CoDE/GM is better than CoDE on $f_1 - f_5$. They both achieve the optimal solution on f_6 and CoDE/GM achieves the optimal solution on f_6, f_8, f_{11}. CoDE outperforms CoDE/GM on f_7. CoDE/GM has a striking improvement on the performance compared to CoDE on $f_1 - f_5$, especially on f_1. The statistical result shows that CoDE/GM worse than CoDE on one test instances, better than CoDE on five test instances and similar to CoDE on seven test instances.

In term of JADE, it performs well on most test instances except on f_5,f_7 in Table 9. It obtains the optimal value on f_6, $f_8 - f_9$, and f_{11}. JADE/GM has a good performance on $f_1 - f_3$, $f_5 - f_6$, $f_8 - f_{13}$. Compared to JADE, JADE/GM achieves better results on $f_1 - f_3$, $f_5 - f_6$. It is clear that JADE/GM declines the performance of JADE on f_4 and improves JADE on f_5. The statistical result is that CoDE/GM worse than CoDE on two test instances, better than CoDE on four test instances and similar to CoDE on seven test instances.

The above analysis suggests that the Gaussian probabilistic model based operator has a good effect on the procedure of generating solutions. The idea of the hybrid algorithm with the Gaussian probabilistic model based operator and the basic operator is a direction in improving the ability of finding the optimal solutions.

Table 9
Mean, Std. values of the results obtained by the four comparison algorithms after 300,000 function evaluations over 30 runs for all the test instances.

	CoDE	CoDE/GM	JADE	JADE/GM
f_1	9.88e-009±3.80e-009	**1.74e-032**±9.06e-033 (+)	6.55e-126±3.59e-125	**1.18e-150**±5.87e-150 (+)
f_2	1.56e-005±2.75e-006	**1.49e-017**±3.34e-018 (+)	9.70e-038±5.30e-037	**1.18e-060**±5.74e-060 (+)
f_3	5.81e-002±3.19e-002	**3.20e-005**±2.36e-005 (+)	**5.12e-035**±1.45e-034	2.95e-033±9.73e-033 (−)
f_4	2.71e-001±5.49e-002	**3.18e-008**±1.11e-008 (+)	**5.45e-014**±1.92e-013	5.69e-001±3.87e-001 (−)
f_5	2.10e+001±5.55e-001	**5.14e+000**±1.14e+000 (+)	1.33e-001±7.28e-001	**3.79e-022**±1.76e-021 (+)
f_6	**0.00e+000**±0.00e+000	**0.00e+000**±0.00e+000 (~)	**0.00e+000**±0.00e+000	**0.00e+000**±0.00e+000 (~)
f_7	**1.18e-002**±4.16e-003	1.38e-001±4.05e-002 (−)	3.94e-001±8.34e-002	**8.35e-002**±3.09e-002 (+)
f_8	1.10e-005±6.62e-006	**0.00e+000**±0.00e+000 (+)	**0.00e+000**±0.00e+000	**0.00e+000**±0.00e+000 (~)
f_9	7.51e+000±1.44e+000	**6.39e-005**±1.32e-004 (+)	**0.00e+000**±0.00e+000	**0.00e+000**±0.00e+000 (~)
f_{10}	2.67e-005±6.38e-006	**4.44e-015**±0.00e+000 (+)	**4.44e-015**±0.00e+000	**4.44e-015**±0.00e+000 (~)
f_{11}	1.96e-006±7.72e-006	**0.00e+000**±0.00e+000 (+)	**0.00e+000**±0.00e+000	**0.00e+000**±0.00e+000 (~)
f_{12}	6.26e-010±2.59e-010	**1.57e-032**±5.57e-048 (+)	**1.57e-032**±5.57e-048	**1.57e-032**±5.57e-048 (~)
f_{13}	3.13e-009±1.58e-009	**2.38e-032**±7.96e-033 (+)	**1.35e-032**±5.57e-048	**1.35e-032**±5.57e-048 (~)
+/−/~	11/1/1		4/2/7	

4. Conclusions

In this paper, we proposed a hybrid algorithm for global optimization problems, named DE/GM. In our approach, some of the new solutions are created by the Gaussian probabilistic model based operator and the rest of solutions are generated by the DE operator, which combines individual and population information efficiently. The Gaussian probabilistic model based operator is a new reproduction strategy based on clustering and mean-shift, which balances the exploration and the exploitation search abilities.

To evaluate the performance of our algorithm, we compared DE/GM with three algorithms, DE/EDA, DE and MSDE. 13 test instances are used as the benchmark functions. The experimental results indicate that DE/GM can obtain good performance and has a faster convergence rate compared to DE on all the test instances. Moreover, DE/GM outperforms DE/EDA on some instances and improves the work on MSDE. The role of DE/GM components are studied and the results show that the two reproduction operators are both necessary. The effect of the algorithm parameters, K and P_c, are also empirically studied on four selected test instances. The experimental results indicate that the square root of population size is a considerable choice for the number of clusters and DE/GM is not very sensitive to the parameters P_c. The effect of mean-shift search method is also discussed. The statistical results show that the local search method is efficient and effective. The scalability to variable size, and the consumed CPU time have also been studied. In addition, we apply the Gaussian probabilistic model based operator to other evolutionary algorithms to investigate the usability by hybridizing it with other operators. JADE and CoDE are used to test it and both get high-quality solutions.

There are several research avenues worthwhile exploring in the future. First, recall that K-means needs the number of clusters in the beginning during the procedure and it increases the complexity of our algorithm. Hence, we will work on other cluster methods without parameters to improve the efficient of our algorithm. Second, the application of the Gaussian based operator will be studied.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No. 61731009, 61673180, and 61703382, the Shanghai Clearing House under the project of 'artificial intelligence methods for complex 0–1 financial optimization', and the Science and Technology Commission of Shanghai Municipality under research Grant 14DZ2260800.

References

[1] T. Back, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, IOP Publishing Ltd., 1997.
[2] D.E. Goldberg, M.P. Samtani, Engineering optimization via genetic algorithm, in: Electronic Computation (1986), ASCE, 1986, pp. 471–482.
[3] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[4] S. Das, S.S. Mullick, P. Suganthan, Recent advances in differential evolution–an updated survey, Swarm Evolut. Comput. 27 (2016) 1–30.
[5] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[6] J. Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 760–766.
[7] H. Mühlenbein, G. Paass, From recombination of genes to the estimation of distributions i: binary parameters, in: International Conference on Parallel Problem Solving from Nature (PPSN IV), Springer, 1996, pp. 178–187.
[8] P. Larranaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publisher, 2002.
[9] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: IEEE Congress on Evolutionary Computation (CEC 2004), vol. 2, IEEE, 2004, pp. 1980–1987.
[10] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer Science & Business Media, 2006.
[11] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. 38 (1) (2008) 218–237.
[12] F. Neri, E. Mininno, Memetic compact differential evolution for cartesian robot control, IEEE Comput. Intell. Mag. 5 (2) (2010) 54–65.
[13] L. Wang, L.-P. Li, Fixed-structure controller synthesis based on differential evolution with level comparison, IEEE Trans. Evol. Comput. 15 (1) (2011) 120–129.
[14] G.W. Greenwood, Using differential evolution for a subclass of graph theory problems, IEEE Trans. Evol. Comput. 13 (5) (2009) 1190–1192.
[15] G. Wu, R. Mallipeddi, P. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[16] J. Wang, J. Liao, Y. Zhou, Y. Cai, Differential evolution enhanced with multiobjective sorting-based mutation operators, IEEE Trans. Cybern. 44 (12) (2014) 2792–2805.
[17] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[18] F. Penunuri, C. Cab, O. Carvente, M. Zambrano-Arjona, J. Tapia, A study of the classical differential evolution control parameters, Swarm Evolut. Comput. 26 (2016) 86–96.
[19] Q. Wei, X. Qiu, Dynamic differential evolution algorithm with composite strategies and parameter values self-adaption, in: 2015 Seventh International Conference on Advanced Computational Intelligence (ICACI), IEEE, 2015, pp. 271–274.
[20] Y. Ping Chen, T.-L. Yu, K. Sastry, D. E. Goldberg, reportA Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms, Natural Computing Laboratory (NCLab), Department of Computer Science, National Chiao Tung University, Technical Report NCL-TR-2007009.
[21] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, Swarm Evolut. Comput. 1 (3) (2011) 111–128.
[22] S. Khor, Linkage structure and genetic evolutionary algorithms, in: Y.-p. Chen (Ed.), Exploitation of Linkage Learning in Evolutionary Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 3–23.
[23] A. Zhou, Q. Zhang, Y. Jin, E. Tsang, T. Okabe, A model-based evolutionary algorithm for bi-objective optimization, in: IEEE Congress on Evolutionary Computation (CEC 2005), 2005, pp. 2568–2575.
[24] T.A. El-Mihoub, A.A. Hopgood, L. Nolle, A. Battersby, Hybrid genetic algorithms: a review, Eng. Lett. 13 (2006) 124–137.
[25] C. Grosan, A. Abraham, Hybrid evolutionary algorithms: methodologies, architectures, and reviews, in: A. Abraham, C. Grosan, H. Ishibuchi (Eds.), Hybrid Evolutionary Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 1–17.
[26] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evolut. Comput. 2 (2012) 1–14.
[27] J. Sun, Q. Zhang, E.P. Tsang, De/eda: a new evolutionary algorithm for global optimization, Inf. Sci. 169 (3) (2005) 249–262.
[28] W. Gong, A. Zhou, Z. Cai, A multioperator search strategy based on cheap surrogate models for evolutionary optimization, IEEE Trans. Evolut. Comput. 19 (5) (2015) 746–758.
[29] H. Fang, A. Zhou, G. Zhang, An estimation of distribution algorithm guided by mean shift, in: Conference on Bio-inspired Computing: Theories and Applications, 2016.
[30] H. Mühlenbein, J. Bendisch, H.-M. Voigt, From recombination of genes to the estimation of distributions ii: continuous parameters, in: International Conference on Parallel Problem Solving from Nature (PPSN IV), Springer, 1996, pp. 188–197.
[31] K. Fukunaga, L.D. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, IEEE Trans. Inf. Theory 21 (1975) 32–40.
[32] D. Comaniciu, P. Meer, Mean shift analysis and applications, in: The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999. vol. 2, IEEE, 1999, pp. 1197–1203.
[33] D. Comaniciu, V. Ramesh, P. Meer, Real-time tracking of non-rigid objects using mean shift, in: IEEE Conference on Computer Vision and Pattern Recognition, 2000. vol. 2, IEEE, 2000, pp. 142–149.
[34] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Mach. Intell. 24 (5) (2002) 603–619.
[35] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, University of California Press, 1967, pp. 281–297.
[36] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[37] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[38] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (1) (2011) 55–66.
[39] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[40] W. Sheng, S. Swift, L. Zhang, X. Liu, A weighted sum validity function for clustering with a hybrid niching genetic algorithm, IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) 35 (6) (2005) 1156–1167.
[41] H. Guo, Y. Li, J. Li, H. Sun, D. Wang, X. Chen, Differential evolution improved with self-adaptive control parameters based on simulated annealing, Swarm Evolut. Comput. 19 (2014) 52–67.