Improving Estimation of Distribution Algorithm on Multi-modal Problems by Detecting Promising Areas

ARTICLE *in* IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART B (CYBERNETICS) · OCTOBER 2014

Impact Factor: 3.78 · DOI: 10.1109/TCYB.2014.2352411

3 AUTHORS, INCLUDING:

Peng Yang
University of Science and Technology of China
5 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Xiaofen Lu
University of Science and Technology of China
5 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Improving Estimation of Distribution Algorithm on Multimodal Problems by Detecting Promising Areas

Peng Yang, *Student Member, IEEE*, Ke Tang, *Senior Member, IEEE*, and Xiaofen Lu

***Abstract*—In this paper, a novel multiple sub-models maintenance technique, named maintaining and processing sub-models (MAPS), is proposed. MAPS aims to enhance the ability of estimation of distribution algorithms (EDAs) on multimodal problems. The advantages of MAPS over the existing multiple sub-models based EDAs stem from the explicit detection of the promising areas, which can save many function evaluations for exploration and thus accelerate the optimization speed. MAPS can be combined with any EDA that adopts a single Gaussian model. The performance of MAPS has been assessed through empirical studies where MAPS is integrated with three different types of EDAs. The experimental results show that MAPS can lead to much faster convergence speed and obtain more stable solutions than the compared algorithms on 12 benchmark problems.**

***Index Terms*—Estimation of distribution algorithms (EDAs), multimodal problems, promising areas.**

I. Introduction

ESTIMATION of distribution algorithms (EDAs) [1], [2] are a new branch of evolutionary algorithms (EAs). An EDA generates new individuals by sampling from a joint probability distribution estimated from the promising individuals of previous generations. Through the joint probability distribution, the structure of a problem can be captured explicitly, which can be employed to guide the optimization. During the past few decades, EDAs have achieved great success in both combinatorial [1], [3]–[6] and continuous domains [7]–[13]. In this paper, EDAs for continuous domain are studied.

The most commonly used EDAs for continuous optimization problems are probably those adopting a single Gaussian joint probability distribution, e.g., univariate marginal distribution algorithm (UMDA$_c$) [1], estimation of multivariate normal algorithm (EMNA$_g$) [1], and eigenspace estimation of distribution algorithm (EEDA) [2], [14]. This type of EDAs have shown competitive performance on relatively simple problems and involve relatively low computational cost. However, they may perform poorly on multimodal problems [15], [16], since the structure of a multimodal problem cannot be well represented by a single Gaussian model [15].

In other words, multimodal problems naturally call for more complicated models that involve several sub-models, each of which evolves a group of individuals. A typical method along this direction is the Gaussian mixture model (GMM) [17], which integrates several single Gaussian models. In [18], the GMM has been adopted into EDAs and its advantages on multimodal problems have been shown. However, since GMM is estimated explicitly by expectation maximization (EM) algorithm [19], the computational cost is usually high. Some researchers thus suggested quite a few alternative (and computationally more efficient) approaches, e.g., clustering techniques [15], [20], [21], niching methods [16], and parallel island models [22], [23] to manage multiple Gaussian models. Among them, clustering based EDAs apply clustering techniques to divide a population into several clusters, and form a sub-model for each cluster. Niching-based EDAs maintain the diversity among sub-models by keeping only one best sub-model alive in one area, island model based EDAs evolve sub-models separately, and sub-models communicate with each other periodically via migration.

Although the above-mentioned approaches can address multimodal problems to some extent, they still suffer from a few difficulties. For the niching based and island model based EDAs, the sub-models are initialized randomly, which may be far from (either global or local) optima. Thus, it may take a long time for the sub-models to move toward real promising areas. Clustering-based EDAs build sub-models based on the clusters of individuals produced by the clustering techniques. Although those corresponding areas usually have better fitness, it is unclear whether all the individuals in a single cluster lie close to the same local or global optimum. If it is not the case, one sub-model will be required to handle another multimodal problem, and hence the potential advantages of using multiple sub-models will not take effect.

Recall that multiple sub-models need to be employed in an EDA because a single model may not represent multimodal problems sufficiently well. Hence, intuitively speaking, it is expected that each sub-model will "cover" one optimum or sub-optimum. In addition, an optimum should be covered by

Manuscript received March 5, 2014; revised August 22, 2014; accepted August 22, 2014. This work was supported in part by the 973 Program of China under Grant 2011CB707006, in part by the National Natural Science Foundation of China under Grant 61329302 and Grant 61175065, in part by the Program for New Century Excellent Talents in University under Grant NCET-12-0512, and in part by the Science and Technological Fund of Anhui Province for Outstanding Youth under Grant 1108085J16. This paper was recommended by Associate Editor P. P. Angelov. *(Corresponding Author: K. Tang.)*

The authors are with the USTC-Birmingham Joint Research Institute in Intelligent Computation and its Applications, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: trevor@mail.ustc.edu.cn; ketang@ustc.edu.cn; xiaofen@mail.ustc.edu.cn).

Digital Object Identifier 10.1109/TCYB.2014.2352411

no more than one sub-model to avoid the waste of searching effort in the same area. From these perspectives, the sub-models can be more effectively maintained if the area of attractions for each optimum can be identified explicitly. Inspired by this consideration, a novel multiple sub-models maintenance technique, named maintaining and processing sub-models (MAPS), is proposed in this paper. MAPS consists of two phases, i.e., maintaining and processing. The maintaining phase detects the promising areas based on the relation between the distributions of individuals and the local optima. As the truncation selection always maintains individuals with better fitness, if the fitness landscape of an area changes acutely, the quantities of selected individuals in this area will also vary sharply. Furthermore, the fitness landscape of an area will not change significantly if no local optimum lies in the area. Thus, the quantities of individuals provide useful information for detecting promising areas that contain local optima. That is, if the quantities of individuals in an area change significantly, the area is more likely to contain a local optimum. Unfortunately, it is nontrivial to precisely acquire the quantities of individuals anywhere in a multidimensional space. Hence, a less precise but much simpler method is proposed here. Briefly speaking, the proposed method constructs histograms of individuals with respect to multiple 1-D subspaces of the original search space. Then, individuals that are close to the same promising area are identified by examining the 1-D sub-spaces sequentially. After that, these individuals will be used to establish sub-models. After the maintaining phase, the sub-models will be processed by some specifically designed operators in the processing phase to enhance the ability of MAPS. Within the EDA framework, the sub-models are evolved separately by a single Gaussian model based EDA.

Empirical studies on 12 widely used benchmark problems have shown that EDAs with MAPS outperform the compared algorithms in terms of both efficiency and solution quality.

The rest of this paper is organized as follows. In Section II, some related work are introduced. In Section III, the MAPS is presented in detail. Experimental results and analyses are given in Section IV. Section V provides the conclusion of this paper.

II. Related Work

To solve multimodal problems, multiple sub-models based EDAs are preferred. In the literature, different sub-models maintenance techniques are adopted, e.g., clustering techniques, niching methods and parallel island models. In this section, a review of those work is presented.

A. Clustering Techniques

Clustering techniques have been commonly used to divide the population of an EA into sub-populations. For example, clustering and estimation of gaussian distribution algorithm (CEGDA) [15] applies the adaptive rival penalized competitive learning (aRPCL), which is an extended version of RPCL [24], [25], to divide the selected population into several clustered sub-populations. RPCL has the ability to automatically detect the number of promising local regions during clustering process without predefining the number of clusters. Additionally, aRPCL can update the covariance matrices during the clustering process. In each generation of CEGDA, aRPCL firstly randomly initializes two positions in the solution space as the center of two clusters. Then, by comparing the Mahalanobis distance between each cluster and each individual, not only is the winner cluster attracted to the individual but also the second winner (rival) is pushed away. This mechanism produces an implicit force to make sure each individual is only assigned to one cluster. After the clustering procedure, CEGDA constructs single Gaussian models, e.g., estimation of gaussian distribution algorithm [26], based on each cluster.

There are some other clustering techniques [21], [27], [28] that are used to maintain multipopulations. Bosman and Thierens [21] adopt randomized leader algorithm and k-means clustering method so as to factorize the nonlinear dependency in the sample set into a linear one. In this way, iterated density estimation algorithms can be very efficient for solving each sub-population. In [27], a fitness-aided ordering scheme is devised for deciding the input sequence of individuals for clustering and it can effectively categorize the individuals by using the (available) information about fitness landscape. Chen and Hu [28] utilize affinity propagation (AP) clustering analysis to adaptively partition the niches and mine searching information from the evolution process.

B. Niching Methods

In addition to clustering, niching techniques, which are commonly employed when seeking multiple optima with EAs, there are also off-the-shelf approaches to enhance EDAs performance on multimodal problems.

NichingEDA [16] generates sub-populations randomly and adopts a niching method called clear procedure [29] to maintain sub-populations within the solution space, so that promising local regions can probably attract one of them. Clear procedure [29] is based on the concept of limited resources of the environment, i.e., in a local area, only the best several sub-populations can survive after competition. Thus, at each generation, each sub-population firstly self-evolves for a fixed number of generations and the clear procedure will be utilized to keep each local area occupied by a predefined number of sub-populations. While some sub-populations have been discarded, new sub-populations will be generated by applying crossover and mutation operators to representative sub-populations, which means each sub-population will be regarded as an individual in classical EAs.

C. Island Model

Distributed or parallel EAs are typical EAs that maintain multipopulations in the evolutionary process. Hence, typical techniques in this context, such as the island model, have also been introduced into EDAs. The concept of island has been adopted in genetic algorithms (GAs) [30] and has been a branch of parallel genetic algorithms (PGAs) [31]. Similar to niching methods, the Island model also works for maintaining sub-populations during evolution. However, the Island model neither discards sub-populations nor generates new

ones. It maintains sub-populations by independently and occasionally exchanging information through a predefined and fixed topological structure (e.g., star or ring) [22], [32]. This information exchange (called migration) is the key of island based GAs. An example for the progress along this direction is the IslandEDA [22].

IslandEDA can be viewed as a counterpart of island GA [30] in the EDA domain. IslandEDA firstly initializes several sub-populations randomly, which is the same as CEGDA and NichingEDA, and then utilizes the migrating operators above to drive sub-populations (islands) to explore the solution space. Instead of migrating individuals from one sub-population to another, IslandEDA transfers model parameters among local models. This replacement has three advantages [32].

1) A probabilistic model maintains more information than a subset of individuals because it always represents and resumes a larger number of them. Besides, it may contain additional information about relationships among variables.
2) The information carried by model parameters is more compact and explicitly represented, and hence is easier to be analyzed.
3) The migration of models is much more flexible than migration of individuals especially when there are fewer migrating individuals.

In addition to IslandEDA, other variants of island-based EDAs that migrate individuals rather than model parameters, have also been investigated [23]. As discussed above, IslandEDA is more efficient than other island-based EDA, especially when there are fewer migrating individuals.

III. MAPS Algorithm

As introduced in Section I, although a few approaches have successfully improved the performance of EDAs on multimodal problems, they still suffer from some drawbacks. In fact, the difficulties of those approaches are mainly because they neither do not explicitly detect different promising areas in the search space, nor guarantee that each sub-model will only spend its search effort on one promising area at a time. In order to overcome these drawbacks, we propose a novel multiple sub-models maintenance technique in this paper, named MAPS. MAPS consists of two phases, i.e., maintaining and processing. In this section, these two phases are described in detail.

A. Maintaining Phase

The maintaining phase detects the promising areas with the help of the relation between the distributions of individuals and local optima. As the truncation selection preserves more individuals in the areas with better fitness, the quantities of individuals vary in the areas with different fitness. In this case, if the fitness landscape of an area changes acutely, the quantities of individuals in them will vary sharply. Notice that the fitness landscape of the areas will not significantly change unless it contains local optima. Thus, the quantities of individuals provide useful information for detecting promising areas that consist of local optima. In other words, if the quantities change significantly in an area, the area is more likely to contain a local optimum. Unfortunately, it is nontrivial to precisely acquire the quantities of individuals anywhere in a multidimensional space. Hence, MAPS adopts a less precise but much simpler method. Briefly speaking, it constructs histograms of individuals with respect to multiple 1-D sub-spaces of the original search space. Then, individuals that are close to the same promising area are identified by examining the 1-D sub-spaces sequentially.

To construct the histogram in a 1-D sub-space, all the individuals are first projected onto this sub-space. The projected individuals, denoted as *proPop*, are only used to construct the histogram on this 1-D sub-space. The interval between the leftmost and rightmost projected individuals is equally divided into several bins. For each bin, the number of individuals falling into it is counted. The width of a bin is set to be $(maxloc - minloc)/\lceil |population|/5 \rceil$, where *maxloc* and *minloc* are the locations of the rightmost and leftmost individuals, and $|population|$ denotes the size of the selected population. It can be seen that the width of bins changes with different populations, which makes sense because smaller populations need narrower bins than the larger populations. The major steps in constructing a histogram in 1-D sub-space can be as follows.

MakeHistogram(*population*, *V*, dim)
1) $proPop \doteq population \cdot V[\text{dim}]$;
2) $maxloc \doteq \text{Max}(proPop)$;
3) $minloc \doteq \text{Min}(proPop)$;
4) $width \doteq (maxloc - minloc)/\lceil \frac{|proPop|}{5} \rceil$;
5) **for** i $\doteq$ 1 **to** $|proPop|$
6) $order \doteq \lceil (proPop[\text{i}]\text{-}minloc)/width \rceil$;
7) $freq[order] + +$;
8) **Return** *freq*;

where max and min means the maximal and minimal value of variable of *proPop*, the *freq* is an array where *freq*[i] represents the number of individuals falling into the *i*th bin, *V* is a vector that denotes the direction of each 1-D sub-space, dim indicates the order of 1-D sub-spaces that is currently observed. The choice of *V* and dim is introduced later in this subsection.

As the histogram is constructed, the change of *freq* of bins can be observed to detect the promising areas. In fact, this change is to highlight the different *freq* among the 1-D histogram. And these differences can be simply defined as the ratio of height between the *freq* of two consecutive bins. The larger the ratio is, the more acute the change will be. If any ratio is larger than *e* (the Euler's number), the area covered by the corresponding bins is assumed to contain local optima. To calculate the ratio, these corresponding bins should first be marked. In turn, the ratio is used to judge whether the area covered by these bins contains local optima or not. For this purpose, intuitively, we first observe the 1-D histogram from the leftmost side to the rightmost side to locate the relatively higher bins, then those desired sets of bins can be confirmed around the higher bins.

In terms of detail, when the observation begins, one would keep updating the flag large as the bin with currently largest frequency. The observation goes on for one bin after another, until the current bin is *e* times lower than large. Then the location of large as "higher bin" is recorded since it implies that the area there changes acutely. After that, the flag large is reset

as the first following bin that is higher than its previous one, i.e., it is expected to start at a rather low place for successive observations. The major steps in finding the "higher bin" are as follows.

```
FINDHIGHERBIN(freq)
resetFlag ≒ false;
l ≒ 1;
num ≒ 0;
1)  for i ≒ 1 to |bin|
2)    if freq[i] > freq[large]
3)      large ≒ i;
4)    if e · freq[i] < freq[large]
5)      num++;
6)      higherBin[num] ≒ large;
7)      resetFlag ≒ true;
8)    if resetFlag = true & freq[i]> freq[i-1]
9)      large ≒ i;
10)     restFlag ≒ false;
11) Return higherBin;
```

Here *higherBin* is an array that records the bin of each located "higher bin," *freq* is the output of MAKEHISTOGRAM and *resetFlag* is a boolean variable that is used to help reset the *large*.

As the "higher bins" are found, their corresponding sets of bins can easily be confirmed around them. For each "higher bin," count from itself to both sides one by one to locate the left and right frontiers of this set of bins. If any bin at left (right) hand is e times lower than the "higher bin," it is regarded as the left (right) frontier of this set of bins. The individuals, without being projected, between both frontiers are regarded as a sub-population that covers the promising area on this 1-D sub-space. The major steps in confirming the sets of bins on an 1-D sub-space are as follows.

```
CONFIRMBINS(higherBin, freq)
1)  for i ≒ 1 to |higherBin|
2)    for j ≒ higherBin[i] to 1
3)      if e · freq[j] < freq[higherBin[i]]
4)        leftFrontier ≒ j;
5)    for j ≒ higherBin[i] to length[bin]
6)      if e · freq[j] < freq[higherBin[i]]
7)        rightFrontier ≒ j;
8)    Form subPop[i] with individuals in the bins
      between leftFrontier and rightFrontier;
9)  Return subPop;
```

Here the *subPop* is an array and *subPop*[i] stands for the sub-population around *higherBin*[i]. *higherBin* is the output of FINDINGHIGHERBIN.

In the view of multidimensional space, the given selected population will be iteratively observed in all 1-D sub-spaces. That is, only the sub-populations produced in the previous sub-space will be observed in the next sub-space. Otherwise, they will be ignored. Intuitively, the observation process is like a filtering process where the individuals belonging to no sub-populations are left out and only the individuals on the promising areas remain. Thus, as the iterative observation goes on, the quantity of the remained individuals gradually decreases, and the ones that ultimately remain are all on promising areas.

Supposing the multidimensional space has been represented by M 1-D sub-spaces, the major steps of iterative observation in each 1-D sub-space are described as follows.

```
ITEROBSERVE(subPop, V, dim)
1)  freq := MAKEHISTOGRAM(subPop, V, dim);
2)  higherBin := FINDHIGHERBIN(freq, dim);
3)  subPop := MARKBINS(higherBin, freq, dim);
4)  if dim < M
5)    for each subPop
6)      finSubPop := ITEROBSERVE(subPop, V, dim + 1);
7)  Return finSubPop;
```

Notice that, in order to distinguish the individuals in each sub-population, these sub-populations will be observed separately in successive 1-D sub-space. That is, each sub-population produced in the current sub-space is the input of the next iterative observation.

Before constructing the 1-D histograms, these being observed sub-spaces should be chosen. In the most straightforward way, these sub-spaces can be the axes of the search space, i.e., the Cartesian coordinates. However, when the problems are rotated or not parallel to the axis, the observation process may be ineffective since the distribution of individuals on those sub-spaces may be correlated somehow. To solve this problem, the sub-spaces should be along the eigenvectors of the covariance matrix of the distribution. For the purpose of calculating the eigenvectors, principal component analysis (PCA) [33] is adopted. Concretely, first the covariance matrix is obtained by estimating the truncatedly selected population with maximum likelihood estimation (MLE) to represent the distribution. Then, the matrix is decomposed with eigen-decomposition and the eigenvalues and the corresponding eigenvectors, i.e., principal components, are obtained. According to the PCA algorithm, the larger the eigenvalue is, the more important the principal component will be. Thus, several larger principal components whose eigenvalues add up to 85% of the whole are selected to be the 1-D sub-spaces. To make the maintaining phase comprehensively understood, a pseudo-code of maintaining is given below.

```
MAINTAINING(offspring)
1)  covMatrix := Cov(offspring);
2)  [D,V′] := EigenDecompose(covMatrix);
3)  Select the 1-D sub-spaces V from V′;
4)  finSubPop := ITEROBSERVE(offspring, V, 1);
5)  Return finSubPop;
```

Here, *offspring* is the truncatedly selected population. Cov is the function of estimating the covariance matrix, i.e., *covMatrix*, with *offsprings* by MLE. D is the eigenvalues after eigen-decomposition, i.e., EigenDecompose, while the V' is the eigenvectors and V is the selected 1-D spaces.

B. Processing Phase

As the landscape of some problems may contain a huge number of local optima, the Maintaining phase may locate many promising areas and thus produce numerous

sub-populations. However, to evolve them all is computationally expensive and unnecessary since only the global optimum is pursued. Thus, all the sub-populations need to compare themselves with each other with their best individual, and only the best ten of them are chosen to be evolved.

After selecting sub-populations, the sub-models should be initialized for them. Since sub-models are considered as Gaussian distribution in this paper, the mean vector, and the covariance matrix should be initialized. Commonly, the mean vector of a sub-model is initialized as the average of the individuals belonging to its corresponding sub-population. To initialize the covariance matrix, one thing we notice is that the distribution of each sub-population may be influenced by the range of the size of its covering area. That is, the areas with different sizes make the individuals variously distributed, which reflects different covariance matrices. However, the size of areas should be irrelevant to the height of local optima within them. Thus, in order to give each selected sub-population an equal opportunity, the covariance matrices are initialized as a constant diagonal matrix, i.e., $\Sigma = (h - l/10)I$, rather than estimate the individuals by MLE. The h and l are the boundaries of the search space. Here, ten is the upper bound of selected sub-populations, and I is the identity matrix. Then, these sub-models can be evolved with any kind of single Gaussian model-based EDAs.

Since the histogram is less precise in describing the change of quantities of individuals, there might be a situation where a group of individuals on a promising area becomes several smaller sub-populations after the observation. Although this situation will not influence the exploitation on this area, it may waste function evaluations (FEs) in employing more than one sub-model in the same area. In order to remedy this, the Euclidean distance is adopted to test whether two sub-models are similar or not. That is, if the distance between the mean vectors of two sub-models is smaller than the threshold, they are assumed to be similar to each other. One would then keep the sub-model with best individual alive and leave out the others. The threshold is set to be $(h - l)/100$, where h and l mean the boundaries of the search space.

Except for the problem-dependent operators above, a general difficulty on the multimodal problems is also considered due to the relation between sub-models and the promising areas becomes clear. This difficulty is that some promising areas may always attract the attention of sub-models during the optimization. This will also waste FEs. In MAPS, if any sub-model gets premature, which means that area does not contain global optimum, that area should be avoided in the following evolution and that sub-model should no longer be evolved. Then that sub-model is recorded. If any other sub-model is similar to one of the recorded sub-models, it is simply disregarded. Hence, a sub-model is assumed premature if the fitness of its best individual stays unchanged at a resolution of 1e-4 for ten generations. To make the processing phase clear to implement, the majors steps are listed below. For convenience, the original sub-populations produced by the maintaining phase are denoted as *SS*, the set of sub-models that are estimated from the selected sub-populations are denoted as *ES*, and the set of recorded premature sub-models are denoted as *DS*. The instances of sub-models in each set are denoted as the corresponding lower-case, i.e., *ss*, *es*, and *ds*.

PROCESSING(*ES*)
1) **for each** *es* separately
2) Sample N_{sub} individuals from *es* and evaluate them. Then truncatedly select R_{sub} individuals;
3) Update the parameters of *es* with selected individuals;
4) **for** each *es* separately
5) **if** *es* is premature
6) Record *es* into *DS*;
7) Delete *es* from *ES*;
8) **if** *es* is similar with any other *es*
9) Keep only the best one of them;
10) **if** *es* is similar with any *ds*
11) Delete *es* from *ES*;
12) **Return** *ES*;

C. Full Steps of MAPS

Although the maintaining phase plays a key role in the MAPS algorithm, it will not be executed in every generation. In fact, only when all the sub-models get premature and thus none can be used to evolve, i.e., $ES = \Phi$, where Φ is the null set, the maintaining phase will be executed with a uniformly initialized population. To summarize the MAPS algorithm and get every detail clear, the full steps of MAPS are listed as follows.

MAPS
1) Uniformly initialize N individuals and truncatedly select $R \leq N$ of them to form *offspring*;
2) $SS :=$ MAINTAINING(*offspring*);
3) Sort all *ss* in descending order by the fitness of the best individuals in the corresponding sub-population.
4) Pick *ss* from the first order on, if *ES* is not full and *ss* is dissimilar to any *es* and *ds*, add its corresponding sub-model into *ES*.
5) $ES :=$ PROCESSING(*ES*);
6) **if** some stopping criterion is reached
7) Stop;
8) **elseif** $ES = \Phi$
9) Go to 1;
10) **else**
11) Go to 5;

IV. EXPERIMENTAL STUDIES

Experiments are carefully designed to verify the ability of MAPS on solving multimodal problems. In the following subsections, they are described in detail.

A. Algorithm Settings

In order to show the effectiveness of the multiple sub-models maintenance technique of MAPS, some multiple sub-models based EDAs should be necessarily included in the compared algorithms. For this purpose, three related work

TABLE I
EXPERIMENTAL SETTINGS FOR ALL THE TEST ALGORITHMS

	CEGDA	NichingEDA	IslandEDA	MDE_pBX	MAPS
N	2000	–	–	100	1000
R	500	–	–	–	500
Sub-models Number	automatically determined, initially 2	100	8	–	less than 10
N_{sub}	automatically determined	100	32	–	100
R_{sub}	–	25	16	–	25
Others	α_c = 0.05 α_r = 0.002 Extra units = 5%	Elitist sub-models = 20 $MaxGen_{sub}$ = 5 Niching capacity = 5 Dissimilarity = 1000	–	q =15% n = 1.5 F_m = 0.5 Cr_m = 0.6	$\eta = e$ E_{sub} = 10

TABLE II
BRIEF INFORMATION ABOUT TEST FUNCTIONS

Function Name	Function Number	Dimension	Domain	Optimum
Ackley	f_1	10	[-32,32]	0
Bohachevsky	f_2	10	[1,15]	0
Generalized Rosenbrock	f_3	10	[-30,30]	0
Schaffer	f_4	10	[10,100]	0
Foxholes	f_5	10	[-65.536,65.536]	0.9980038378
Schwefel 2.13	f_6	10	[-π,π]	-460
Generalized Rastrigin	f_7	10	[-5.12,5.12]	0
Shifted Rotated Griewanks	f_8	10	[-600,600]	-180
Shifted Rotated Weierstrass	f_9	10	[-0.5,0.5]	90
TwoPeaks	f_{10}	5	[-100,100]	10.10532602
ThreePeaks	f_{11}	5	[-100,100]	10.10532602
Shekel (n=5)	f_{12}	4	[0,10]	10.10327912

introduced in Section II, e.g., CEGDA [15], NichingEDA [16], and IslandEDA [22], are chosen. Moreover, in the view of the ability of problem-solving, comparisons between MAPS and those canonical EAs are also required. In [34]–[37], quite a few EAs-based methods have been suggested to deal with multimodal problems. However, most of them focus on finding multiple local optima simultaneously, which unfits our goal. Apart from that, there are still some state-of-the-art approaches that can be adopted to search for the global optimum on multimodal problems [38]–[41]. In this experiment, modified DE (MDE) with p-best crossover (MDE_pBX) [42], a recently proposed differential evolution (DE), has also been employed as a compared algorithm. In MDE_pBX, a new mutation operator, called DE/current-to-gr_best/1, is suggested. It uses the best of a group of randomly selected solutions from the current population to perturb the parent vector. Besides, a novel scheme of adapting two of its vital parameters is also proposed to improve its performance. Empirical studies have shown that MDE_pBX can statistically outperform some well-known DE variants on a wide variety of tested problems.

For MAPS, since it is independent from the employed single Gaussian model-based EDA, it is comprehensive to test how MAPS performs when employing different single Gaussian model-based EDAs. Thus, MAPS employing UMDA$_c$ [1], denoted as MAPS$_{\text{UMDA}}$, the MAPS employing EMNA$_g$ [1], denoted as MAPS$_{\text{EMNA}}$, and the MAPS employing EEDA [2], [10], denoted as MAPS$_{\text{EEDA}}$ are also included in the experiment. The reason for choosing these three EDAs is that they are all typical single Gaussian model-based EDAs and are widely used to evolve sub-models [16], [22]. Thus, it is easy to compare and analyse the different behaviors between the compared EDAs based algorithms. For these three MAPS based EDAs, the only difference during the implementation process happens in the step 3) of the PROCESSING. Considering they all estimate the mean vector by MLE, the differences among them exist when estimating covariance matrix. Detailedly, UMDA$_c$ assumes variables are independent, and it thus estimates a covariance matrix by separately estimating the standard deviations of each dimension. In contrast, EMNA$_g$ is a classic multivariate Gaussian model-based EDA, and it estimates covariance matrix by MLE. EEDA is a variant of EMNA$_g$ that fine-tunes the minimal eigenvalue of the covariance matrix to the maximal one, after which it has an approximation to the negative gradient of the fitness function.

Parameters for the four compared algorithms were set accordingly to the original publications. Parameters for three MAPS based EDAs were set to be the same. For each algorithm, the settings are fixed throughout all experiments. The detailed settings are listed in Table I, in which N indicates the population size, R denotes the number of selected individuals, N_{sub} means the number of individuals in each sub-population, R_{sub} represents the number of selected individuals in each sub-population, and E_{sub} is the number of elitism in each sub-population.

B. Experimental Protocol

Twelve functions are chosen. All of them are frequently used as the test functions, including Ackley function (f_1), Bohachevsky function (f_2), generalized Rosenbrock function (f_3), Schaffer function (f_4), Foxholes function (f_5), Schwefel 2.13 function (f_6), generalized Ratrigin function (f_7), shifted rotated Griewanks function (f_8), shifted rotated Weierstrass function (f_9), TwoPeaks (f_{10}), ThreePeaks (f_{11}), and Shekel (n = 5) (f_{12}). More detailed descriptions about these functions can be found in [15] and [43]–[45].

The maximum number of evaluations for each function is set to 5e5. All the tested algorithms terminate when the number of function evaluations reach this limit. The settings of these 12 problems are listed in Table II. All the

TABLE III
EXPERIMENTAL RESULTS

Function	f_1			f_2			f_3		
Algorithm	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
CEGDA	3.20e-08	5.56e-05	8.29e-05	1.90e+01	2.39e+02	1.94e+02	6.69e+00	7.85e+00	6.65e-01
IslandEDA	1.59e-07	1.43e-01	5.08e-01	1.72e-04	2.09e+00	1.37e+00	1.92e-01	7.61e+00	3.63e+00
NichingEDA	1.07e-03	2.18e-03	8.93e-04	1.99e-03	1.73e-01	1.95e-01	5.37e+00	7.91e+00	1.15e+00
MDE_pBX	**0**	6.58e-02	3.29e-01	**0**	1.03e-01	5.14e-01	**0**	**0**	0
$MAPS_{UMDA}$	4.36e-06	**1.19e-05**	5.70e-06	4.95e-10	**1.50e-08**	5.41e-08	1.05e-01	2.38e+00	1.94e+00
$MAPS_{EMNA}$	3.36e-07	2.21e-04	7.82e-04	2.31e+01	1.05e+02	5.99e+01	4.56e-02	4.78e+00	2.00e+00
$MAPS_{EEDA}$	5.45e-06	1.46e-05	9.95e-06	2.01e-07	5.69e-07	2.79e-07	2.88e-01	4.37e+00	1.60e+00
Function	f_4			f_5			f_6		
Algorithm	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
CEGDA	4.87e-02	3.03e-01	2.08e-01	1.19e-07	3.84e-01	6.79e-01	**0**	2.37e+02	5.53e+02
IslandEDA	3.16e-02	7.48e-02	4.27e-02	**0**	7.95e-02	2.75e-01	3.81e+00	1.13e+02	1.32e+02
NichingEDA	2.69e-04	4.93e-03	6.16e-03	1.30e-09	6.20e-06	1.64e-05	1.22e+01	4.36e+01	2.42e+01
MDE_pBX	1.75e-01	5.78e-01	2.69e-01	**0**	**0**	0	**0**	1.28e+03	1.71e+03
$MAPS_{UMDA}$	7.07e-07	1.93e-04	3.87e-04	**0**	**0**	0	1.31e-01	**1.49e+01**	1.98e+01
$MAPS_{EMNA}$	**4.13e-13**	**1.06e-05**	2.96e-05	**0**	**0**	0	5.27e+00	1.76e+02	3.42e+02
$MAPS_{EEDA}$	2.89e-05	1.39e+00	1.65e+00	**0**	3.85e-07	1.06e-06	2.52e-02	2.23e+01	9.14e+01
Function	f_7			f_8			f_9		
Algorithm	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
CEGDA	**0**	**1.69e-01**	3.88e-01	**0**	2.89e+00	2.88e+00	6.43e-01	2.91e+00	2.06e+00
IslandEDA	1.80e-06	6.58e-01	6.26e-01	3.98e-04	7.01e-02	7.81e-02	2.47e+00	6.07e+00	2.32e+00
NichingEDA	4.97e+00	2.57e+01	1.17e+01	2.41e-01	6.75e-01	1.88e-01	5.44e-01	1.45e+00	4.72e-01
MDE_pBX	**0**	1.02e+00	9.84e-01	**0**	**4.24e-02**	1.76e-01	**0**	4.36e+00	2.47e+00
$MAPS_{UMDA}$	5.56e-08	3.04e-01	5.40e-01	1.93e-05	1.29e-01	1.91e-01	2.17e-01	7.12e-01	3.37e-01
$MAPS_{EMNA}$	3.50e-10	7.76e-01	6.95e-01	1.49e-01	1.56e+00	1.39e+00	1.73e-01	1.34e+00	5.17e-01
$MAPS_{EEDA}$	1.33e+01	1.93e+01	2.87e+00	2.20e-06	3.38e-01	2.39e-01	6.03e-03	**8.14e-02**	7.75e-02
Function	f_{10}			f_{11}			f_{12}		
Algorithm	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
CEGDA	**0**	4.33e-01	5.18e-01	1.37e-01	2.49e-01	1.57e-01	2.12e-01	1.38e+00	8.87e-01
IslandEDA	**0**	4.21e-02	2.02e-01	**0**	7.28e-01	1.03e+00	2.32e-05	3.15e+00	2.41e+00
NichingEDA	**0**	1.62e-01	3.78e-01	**0**	2.72e-01	4.57e-01	**0**	1.67e-01	3.46e-01
MDE_pBX	1.01e+01	1.01e+01	0	5.05e+00	8.89e+00	2.20e+00	**0**	**0**	0
$MAPS_{UMDA}$	**0**	1.00e-08	6.78e-09	**0**	1.00e-08	5.86e-09	**0**	1.08e-09	2.00e-09
$MAPS_{EMNA}$	**0**	**0**	0	**0**	**0**	0	**0**	**0**	0
$MAPS_{EEDA}$	**0**	2.00e-08	1.08e-08	**0**	1.00e-08	9.61e-09	**0**	2.31e-09	5.77e-09

All the results were obtained based on 25 independent runs with MaxFEs=5e+05. The column headed Best, Mean, and Std presents the best, average, and standard deviation, respectively. For each column, the best result is highlighted in boldface.

results (in terms of solution quality)[1] have been averaged over 25 independent runs (shown in Table III). Three MAPS-based EDAs are separately compared with all the other algorithms with the two-sided Wilcoxon rank-sum test at a 0.05 significance level, and the corresponding results are presented in Table IV. Two solutions will be regarded as the same if the difference between their fitness (i.e., the objective function value) is smaller than 1e-13. The optimization speed curves of the algorithms on each problem are shown in Figs. 6–17. In them, the lower the curve is, the faster the optimization speed of that algorithm will be.

[1]The quality of a solution x is measured via its function error value $|f(x) - f(x^*)|$, where x^* is the global optimum of f. According to that, a solution is better if it is closer to 0.0 than the compared ones. When discussing the solution quality of an algorithm, the quality of the best solution (i.e., the final solution) obtained by the algorithm is referred to.

C. Results and Analyses

The analysis is divided into four parts based on different points of view.

1) Unimodal Problem: The Ackley problem (f_1) is a unimodal as well as separable problem. The difficulty with this problem is that the global optimal area becomes increasingly narrower which makes the convergence of optimization difficult. As seen from Table III, all the algorithms have

TABLE IV
COMPARISON BETWEEN THREE MAPS BASED EDAS AND FOUR COMPARED ALGORITHMS ON THE 12 TEST FUNCTIONS

Comp. A	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
CEGDA	w	w	w	w	w	d	d	w	w	d	w	w
IslandEDA	w	w	w	w	w	w	w	d	w	w	w	w
NichingEDA	w	w	w	w	w	w	w	w	w	d	w	w
MDE_pBX	w	w	l	w	d	d	d	l	w	w	w	d
Comp. B	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
CEGDA	d	w	w	w	w	w	l	d	d	w	w	w
IslandEDA	w	l	w	w	w	d	d	l	w	w	w	w
NichingEDA	w	l	w	w	w	d	w	l	d	w	w	w
MDE_pBX	w	l	l	w	d	d	d	l	w	w	w	d
Comp. C	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
CEGDA	w	w	w	d	w	w	l	w	w	d	w	w
IslandEDA	w	w	w	l	w	w	l	l	w	w	w	w
NichingEDA	w	w	w	l	w	w	d	w	w	d	w	w
MDE_pBX	w	w	l	d	l	d	l	l	w	w	w	d

Results are presented with w, d and l, standing for that MAPS based EDAs are superior, comparable (i.e., statistically insignificant), and inferior to the compared algorithms. The Comp. A, B and C in each first flank stands for the comparison between the compared algorithms and $MAPS_{UMDA}$, $MAPS_{EMNA}$ and $MAPS_{EEDA}$.

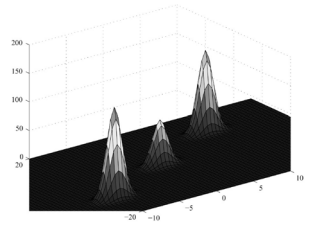


Fig. 1. 3-D-landscape of ThreePeaks.

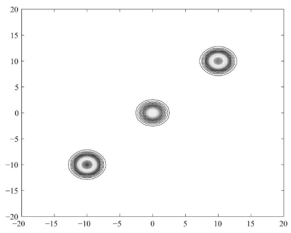


Fig. 2. 2-D-landscape of ThreePeaks.

located the global optimal area but only MDE_pBX can reach the global optimum, though unstably. The reason for those multipopulation based algorithms may be that several sub-populations are inherently unfit for unimodal problems since most sub-populations probably have spent many FEs exploring the global optimal area and rarely obtain benefits, while the sub-population that has located the global optimal area contains insufficient individuals and thus easily gets trapped. On the contrary, MDE_pBX, containing only one population, can allocate all its search resources to exploitation at the converging stage. Unfortunately, the narrow global optimal area sometimes makes MDE_pBX premature as well. Despite that, three MAPS based EDAs have a relatively better performance than compared algorithms from the view of average performance. Specifically, although IslandEDA employs the same single model-based EDA with $\text{MAPS}_{\text{UMDA}}$, i.e., UMDA_c, it performs much worse than $\text{MAPS}_{\text{UMDA}}$. This can be attributed to the maintaining phase as it detects the global optimal area accurately and quickly. Due to the same reason, $\text{MAPS}_{\text{EEDA}}$ outperforms NichingEDA. As seen from Fig. 6,

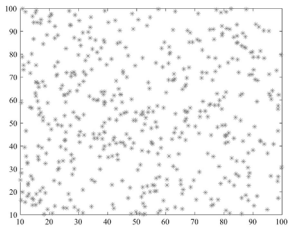


Fig. 3. Selected individuals at the first generation are shown.

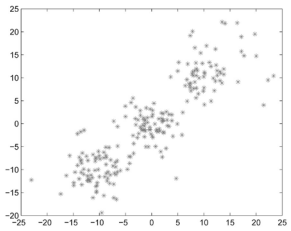


Fig. 4. Selected individuals at the second generation are shown.

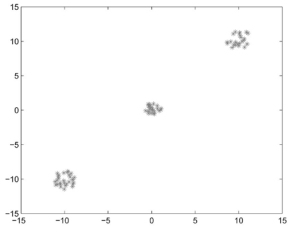


Fig. 5. Selected individuals at the fifth generation are shown.

all three MAPS based EDAs have a much faster optimization speed than the compared algorithms.

2) Problem With Strong Interdependencies: The Rosenbrock problem (f_3) is a multimodal problem that has a very small range of global optimal area among the large range of basins. Intuitively speaking, the difficulty for solving this problem is that there is only one quite narrow valley connecting global optimum and other areas. Besides, the variables are strongly interdependent from each other, which makes the valley more difficult to go through. Thus, f_3 was used to show the advantage of estimation of gaussian networks algorithm by BGe metric over UMDA_c and mutual information maximization for input clustering in [10], which assumes multiple interdependencies among variables.

MDE_pBX shows dominant advantages over the others on this problem. In fact, it can be seen in the literature that most variants of DE are capable of solving Rosenbrock. The reason behind such a phenomenon may be that their effective mutation strategies have the potential to drive the population to the better neighbor area. And this helps the population go through the narrow valley easily. For the remaining EDAs, all three MAPS based EDAs perform better than the others. This is because the Maintaining phase can directly locate the narrow valley as well as the global optimal area. This can be seen from Fig. 8 that both the $\text{MAPS}_{\text{UMDA}}$ and $\text{MAPS}_{\text{EEDA}}$ quickly locate the narrow valley. While the $\text{MAPS}_{\text{EMNA}}$ is stocked at fitness 1e3 for a period, it might be that the interdependency among variables learned by $\text{MAPS}_{\text{EMNA}}$ is far

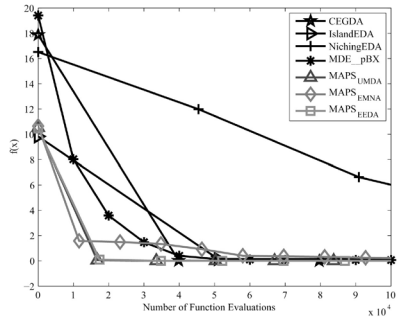


Fig. 6. Optimization speed curves on f_1.

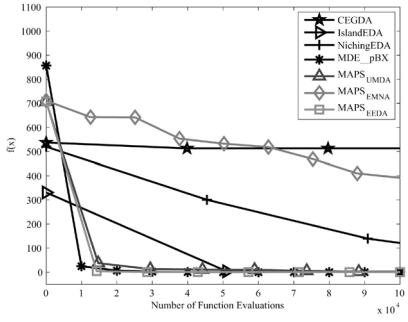


Fig. 7. Optimization speed curves on f_2.

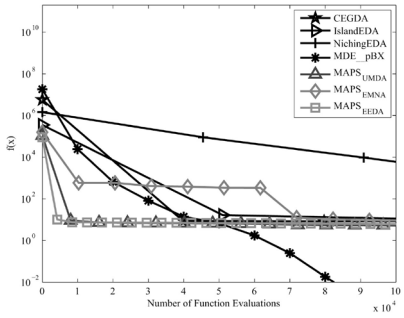


Fig. 8. Optimization speed curves on f_3.

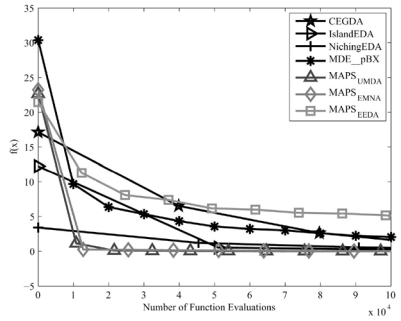


Fig. 9. Optimization speed curves on f_4.

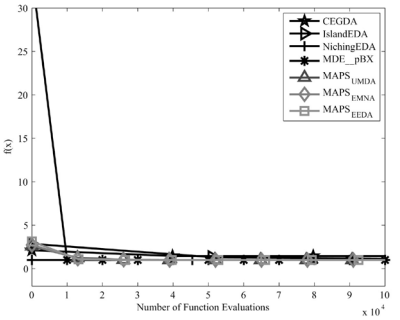


Fig. 10. Optimization speed curves on f_5.

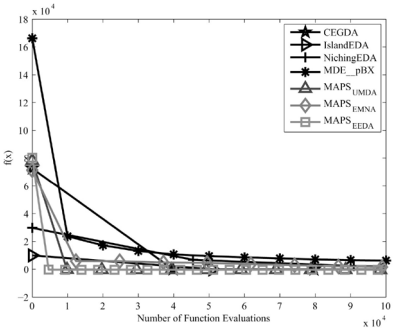


Fig. 11. Optimization speed curves on f_6.

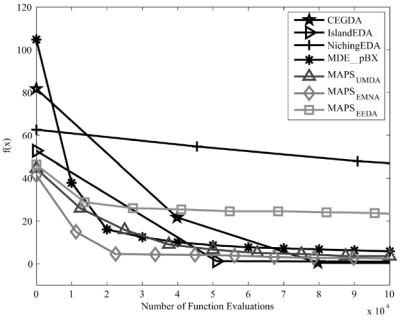


Fig. 12. Optimization speed curves on f_7.

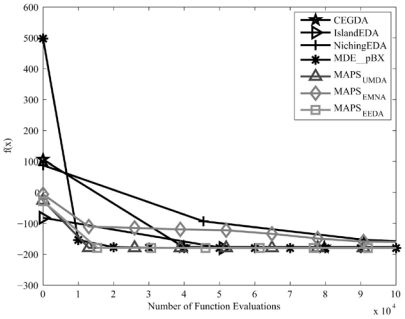


Fig. 13. Optimization speed curves on f_8.

from the real one, and the optimization is misled. However, once the global optimal area is detected, the fitness value decreases rapidly.

3) Multimodal Problems With Few Local Optima: In this part, the results on six test functions, i.e., f_5, f_6, f_9–f_{12}, are discussed. These six problems have in common the feature that the landscape contains several local optima. Tables III and IV show that all the three MAPS based EDAs outperform the three compared EDAs on these six problems. On f_5, f_{10}–f_{12}, $\text{MAPS}_{\text{EMNA}}$ reaches the global optimum in every single run, while $\text{MAPS}_{\text{UMDA}}$ and $\text{MAPS}_{\text{EEDA}}$ are a little less accurate. It can easily be inferred that EMNA_g can capture the local structure of these problems better than UMDA_c and EEDA. Conversely, $\text{MAPS}_{\text{UMDA}}$ and $\text{MAPS}_{\text{EEDA}}$

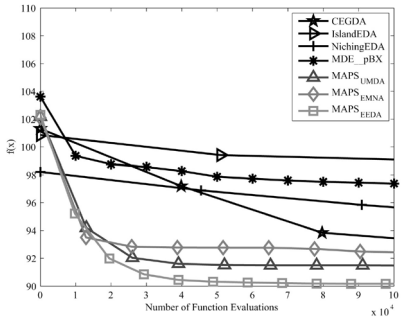


Fig. 14. Optimization speed curves on f_9.

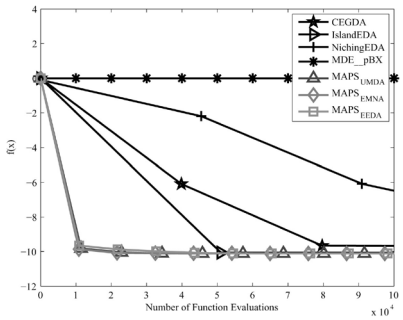


Fig. 15. Optimization speed curves on f_{10}.

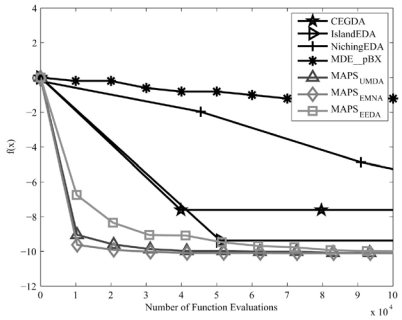


Fig. 16. Optimization speed curves on f_{11}.

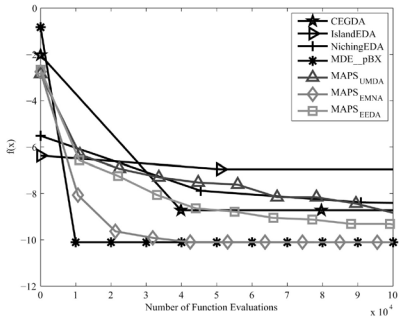


Fig. 17. Optimization speed curves on f_{12}.

outperform $\text{MAPS}_{\text{EMNA}}$ a little on f_6 and f_9 due to the similar reason. Figs. 10, 11, and 14–17 show that the MAPS based EDAs have a very fast optimization speed. Although the curve of NichingEDA in Fig. 10 seems to be lower, it does not reach the global optimum. This is because, the 25 deep holes of f_5 have quite similar fitnesses. And as NichingEDA samples as much as 10 000 individuals within one generation, it is quite easy to obtain a rather good fitness very early. However, this does not mean the sub-models have covered the global optimal area.

MDE_pBX is also able to solve f_5 and f_{12}, and it converges even faster than the MAPS based EDAs on f_{12}. However, its performances on f_{10} and f_{11} are unacceptable. These two problems have in common the feature that the fitness values of basins approach to 0, which provides very little information for the optimization. Hence, MDE_pBX can easily be misled on these two problems. Conversely, the other algorithms can obtain satisfactory solutions by benefiting from the multipopulation schemes. In order to illustrate how MAPS based EDAs work on these two problems, the optimization process of $\text{MAPS}_{\text{EMNA}}$ on the 2-D f_{11} is traced. Specifically, the selected individuals at the first, second, and the fifth generations are shown in Figs. 3–5, respectively. f_{11} has three local optima in the center of the landscape, and the 3-D and 2-D landscapes are shown in Figs. 1 and 2. In the first generation, the individuals are simply uniformly initialized and truncatedly selected. It can be seen in Fig. 3 that, the closer to the center of landscape, the denser the individuals will be, which indicates that the change of quantities of individuals in the center is more acute. Then the maintaining phase is carried out and produces three sub-populations as well as sub-models. The three clusters of individuals in Fig. 4 are sampled from those three sub-models. After that, these three sub-models are evolved with EMNA_g to exploit each promising area. At the fifth generation, these three clusters of individuals are very distinct and their range is very small, which indicates the sub-models have converged.

Specifically, f_9 is a rotated problem whereby the trend of landscape is not parallel to the axis. IslandEDA compromises on this problem. CEGDA, NichingEDA and MDE_pBX obtain comparatively good results while the optimization speed appears very slow. Conversely, as the PCA technique is adopted and the individuals are projected to each principal component iteratively when observing, all MAPS based EDAs perform quite well.

4) Multimodal Problems With Many Local Optima: The rest of the tested problems, i.e., f_2, f_4, f_7, and f_8 belong to this part. Different from the previous part, these problems contain many local optima.

As seen in Tables III and IV, none of the three MAPS based EDAs can dominate the compared algorithms. However, for each problem, there always exists at least one MAPS based EDA that can outperform the compared algorithms. It can be inferred that the bad performance of a certain MAPS based EDA may suffer due to preassumed probabilistic model not fitting the structures of those problems. For example, f_2 is a separable problem, of which the variables are independent of each other. $\text{MAPS}_{\text{EMNA}}$ is compromised on this problem because EMNA_g assumes that all the variables probabilistically interdependent. On the contrary, UMDA_c regards the variables as separate, which can capture the structure of f_2. Hence, $\text{MAPS}_{\text{UMDA}}$ performs significantly better than $\text{MAPS}_{\text{EMNA}}$. Although EEDA also assumes that the variables

are fully connected, it differs from EMNA_g in that it can potentially drive the optimization to the negative gradient of the fitness function. As a consequence, $\text{MAPS}_{\text{EEDA}}$ also outputs very acceptable solutions. Another example is the comparisons on f_4. $\text{MAPS}_{\text{EMNA}}$ and $\text{MAPS}_{\text{UMDA}}$ can outperform the three compared algorithms while $\text{MAPS}_{\text{EEDA}}$ shows no advantages over them. In f_4, a negative gradient of fitness function may mislead the optimization $\text{MAPS}_{\text{EEDA}}$.

V. Conclusion

In this paper, a novel multiple sub-models maintenance technique, named MAPS, is proposed to improve the performance on multimodal problems. In MAPS, the relation between the distribution of individuals and the local optima is studied and used to help detect the promising areas. As the promising areas are found, sub-models have been initialized on them directly, which saves many FEs and thus accelerates the optimization speed. In order to enhance the ability of MAPS, some specified operators are also designed. An experiment with 11 multimodal problems and one uni-modal problem is carried out to test the effectiveness of MAPS. The performances of MAPS employing different kinds of single Gaussian model-based EDAs are also discussed. The experimental studies show that MAPS based EDAs can outperform the compared algorithms with a faster optimization speed on most tested problems. When facing different kinds of problems MAPS based EDAs show better reliability. Besides, as the PCA is adopted, three MAPS based EDAs also perform well on rotated problems.

References

[1] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer, 2001.

[2] W. Dong and X. Yao, "Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms," *Inf. Sci.*, vol. 178, no. 15, pp. 3000–3023, 2008.

[3] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-163, 1994.

[4] G. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Aug. 1999.

[5] J. S. D. Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, 1997, pp. 424–430.

[6] M. Pelikan and H. Muehlenbein, "The bivariate marginal distribution algorithm," in *Advances in Soft Computing*, R. Roy, T. Furuhashi, and P. Chawdhry, Eds. London, U.K.: Springer, 1999, pp. 521–535.

[7] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," in *Parallel Problem Solving from Nature—PPSN V* (Lecture Notes in Computer Science), vol. 1498, A. Eiben, T. Bck, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 1998, pp. 418–427.

[8] S. Rudlof and M. Kppen, "Stochastic hill climbing with learning by vectors of normal distributions," in *Proc. 1st On-Line Workshop Soft Comput.*, Nagoya, Japan, 1996, pp. 60–70.

[9] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation* (Studies in Fuzziness and Soft Computing), vol. 192, J. Lozano, P. Larraaga, I. Inza, and E. Bengoetxea, Eds. Berlin, Germany: Springer, 2006, pp. 75–102.

[10] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. Pena, "Optimization by learning and simulation of Bayesian and Gaussian networks," Dept. Comput. Sci. Artif. Intell., Univ. Basque Country, San Sebastián, Spain, Tech. Rep. EHU-KZAA-IK-4/99, 1999.

[11] C. Ahn, R. Ramakrishna, and D. Goldberg, "Real-coded Bayesian optimization algorithm: Bringing the strength of BOA into the continuous world," in *Proc. Genet. Evol. Comput. (GECCO)*, Seattle, WA, USA, 2004, pp. 840–851.

[12] P. Bosman and D. Thierens, "Expanding from discrete to continuous estimation of distribution algorithms: The idea," in *Parallel Problem Solving from Nature—PPSN VI* (Lecture Notes in Computer Science), vol. 1917, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 2000, pp. 767–776.

[13] D. Thierens, "The linkage tree genetic algorithm," in *Parallel Problem Solving from Nature—PPSN XI* (Lecture Notes in Computer Science), vol. 6238, R. Schaefer, C. Cotta, J. Koodziej, and G. Rudolph, Eds. Berlin, Germany: Springer, 2010, pp. 264–273.

[14] M. R. Wagner, A. Auger, and M. Schoenauer, "EEDA: A new robust estimation of distribution algorithm," Unité recherche, INRIA, Orsay, France, Res. Rep. RR-5190, 2004.

[15] Q. Lu and X. Yao, "Clustering and learning Gaussian distribution for continuous optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 195–204, May 2005.

[16] W. Dong and X. Yao, "NichingEDA: Utilizing the diversity inside a population of EDAs for continuous optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Hong Kong, Jun. 2008, pp. 1260–1267.

[17] X. Song, K. Yang, and M. Pavel, "Density boosting for Gaussian mixtures," in *Neural Information Processing* (Lecture Notes in Computer Science), vol. 3316, N. Pal, N. Kasabov, R. Mudi, S. Pal, and S. Parui, Eds. Berlin, Germany: Springer, 2004, pp. 508–515.

[18] B. Li, R.-T. Zhong, X.-J. Wang, and Z.-Q. Zhuang, "Continuous optimization based-on boosting Gaussian mixture model," in *Proc. 18th Int. Conf. Pattern Recognit.*, vol. 1. Hong Kong, 2006, pp. 1192–1195.

[19] C. B. Do and S. Batzoglou, "What is the expectation maximization algorithm?" *Nat. Biotechnol.*, vol. 26, pp. 897–899, Aug. 2008.

[20] M. Gallagher, M. Frean, and T. Downs, "Real-valued evolutionary optimization using a flexible probability density estimator," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 1999, pp. 840–846.

[21] P. A. Bosman and D. Thierens, "Advancing continuous ideas with mixture distributions and factorization selection metrics," in *Proc. Optim. Build. Probab. Models Workshop Genet. Evol. Comput. Conf. (GECCO)*, 2001, pp. 208–212.

[22] L. delaOssa, J. A. Gamez, and J. M. Puerta, "Initial approaches to the application of island-based parallel EDAs in continuous domains," *J. Parallel Distrib. Comput.*, vol. 66, no. 8, pp. 991–1001, 2006.

[23] J. Madera, E. Alba, and A. Ochoa, "A parallel island model for estimation of distribution algorithms," in *Towards a New Evolutionary Computation* (Studies in Fuzziness and Soft Computing), vol. 192, J. Lozano, P. Larraaga, I. Inza, and E. Bengoetxea, Eds. Berlin, Germany: Springer, 2006, pp. 159–186.

[24] L. Xu, A. Krzyiak, and E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 636–649, Jul. 1993.

[25] L. Xu, "Rival penalized competitive learning, finite mixture, and multisets clustering," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 3. Anchorage, AK, USA, May 1998, pp. 2525–2530.

[26] R. Shachter and C. Kenley, "Gaussian influence diagrams," *Manage. Sci.*, vol. 35, no. 5, pp. 527–550, May 1989.

[27] C. W. Ahn and R. S. Ramakrishna, "Clustering-based probabilistic model fitting in estimation of distribution algorithms," *IEICE Trans. Inf. Syst.*, vol. E89-D, no. 1, pp. 381–383, Jan. 2006.

[28] B. Chen and J. Hu, "An adaptive niching EDA based on clustering analysis," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Barcelona, Spain, Jul. 2010, pp. 1–7.

[29] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, 1996, pp. 798–803.

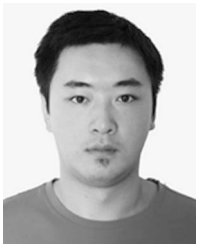
[30] D. Whitley, S. Rana, and R. B. Heckendorn, "The island model genetic algorithm: On separability, population size and convergence," *J. Comput. Inf. Technol.*, vol. 7, pp. 33–47, Nov. 1998.

[31] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity*, vol. 4, pp. 31–52, May 1999.

[32] L. delaOssa, J. A. Gamez, and J. Puerta, "Migration of probability models instead of individuals: An alternative when applying the island model to EDAs," in *Parallel Problem Solving from Nature—PPSN VIII* (Lecture Notes in Computer Science), vol. 3242, X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervs, J. Bullinaria, J. Rowe, P. Tio, A. Kabn, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 2004, pp. 242–252.

[33] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer, 2005.

[34] W. Gao, G. G. Yen, and S. Liu, "A cluster-based differential evolution with self-adaptive strategy for multimodal optimization," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1314–1327, Aug. 2014.
[35] S. Das, S. Maity, B. Y. Qu, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization: A survey of the state-of-the-art," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 71–88, 2011.
[36] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multi-modal optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 387–402, Jun. 2013.
[37] J. Yao, N. Kharma, and P. Grogono, "Bi-objective multi-population genetic algorithm for multimodal function optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 80–102, Feb. 2010.
[38] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[39] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.
[40] W. Y. Gong, Z. H. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 397–413, Apr. 2011.
[41] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 107–124, Feb. 2012.
[42] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 482–500, Apr. 2012.
[43] P. N. Suganthan *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," School Elect. Electron. Eng., Nanyang Technol. Univ., Singapore, Tech. Rep. #2006005, Dec. 2005.
[44] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[45] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Parallel Problem Solving from Nature—PPSN VIII* (Lecture Notes in Computer Science), vol. 3242, X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervs, J. Bullinaria, J. Rowe, P. Tio, A. Kabn, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 2004, pp. 282–291.



Peng Yang (S'14) received the B.Eng. degree in computer science and technology from the University of Science and Technology of China (USTC), Hefei, China, in 2012, where he is currently pursuing the Ph.D. degree from the USTC-Birmingham Joint Research Institute in Intelligent Computation and its Applications, School of Computer Science and Technology.

His current research interests include evolutionary computation, estimation of distribution algorithms, and their real-world applications.

Ke Tang (S'05–M'07–SM'13) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2007.

From 2007 to 2011, he was a Lecturer and Associate Professor with the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. Since 2011, he has been a Professor with the USTC-Birmingham Joint Research Institute in Intelligent Computation and its Applications (UBRI), USTC. His current research interests include evolutionary computation, machine learning, data mining, large scale global optimization, dynamic and robust optimization, and real-world applications. He has authored/co-authored over 80 refereed publications.

Dr. Tang is an Associate Editor of the IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE, the *Computational Optimization and Applications* and the *Frontiers of Computer Science* journals.

Xiaofen Lu received the B.Eng. degree in computer science from the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China, in 2009, where she is currently pursuing the Ph.D. degree from the USTC-Birmingham Joint Research Institute in Intelligent Computation and its Applications (UBRI).

Her current research interests include evolutionary computation and surrogate-based optimization.