NichingEDA: Utilizing the Diversity Inside A Population of EDAs For Continuous Optimization

Weishan Dong, *Student Member, IEEE*, and Xin Yao, *Fellow, IEEE*

***Abstract*— Since the Estimation of Distribution Algorithms (EDAs) have been introduced, several single model based EDAs and mixture model based EDAs have been developed. Take Gaussian models as an example, EDAs based on single Gaussian distribution have good performance on solving simple unimodal functions and multimodal functions whose landscape has an obvious trend towards the global optimum. But they have difficulties in solving multimodal functions with irregular landscapes, such as wide basins, flat plateaus and deep valleys. Gaussian mixture model based EDAs have been developed to remedy this disadvantage of single Gaussian based EDAs. A general framework NichingEDA is presented in this paper from a new perspective to boost single model based EDAs' performance. Through adopting a niching method and recombination operators in a population of EDAs, NichingEDA significantly boosts the traditional single model based EDAs' performance by making use of the diversity inside the EDA population on hard problems without estimating a precise distribution. Our experimental studies have shown that NichingEDA is very effective for some hard global optimization problems, although its scalability to high dimensional functions needs improving. Analyses and discussions are presented to explain why NichingEDA performed well/poorly on certain benchmark functions.**

I. Introduction

Estimation of Distribution Algorithms (EDAs) [1][2] have been studied intensively for their use in continuous optimization domains. Such approaches employ population-based heuristic searching strategies, and have been an active branch of evolutionary algorithms (EAs). The main difference between EDAs and the well-known Genetic Algorithms (GAs) [3] is that a new population is generated with neither crossover nor mutation. Instead, new individuals are sampled from a probability distribution estimated from selected individuals of previous generations. The interrelations between variables are expressed explicitly in EDAs through joint probability distributions. So far, the most widely used probabilistic model in continuous EDAs is the Gaussian model. Several EDAs based on single Gaussian distribution (or variants of Gaussian) [4]-[8] and different forms of Gaussian mixture distribution [9]-[11] have been developed for continuous optimization problems.

According to previous results ([1], [11]-[13]), generally speaking, with a sufficiently large population size, those single Gaussian model based EDAs perform pretty good on simple unimodal functions and those multimodal functions whose global shape has an obvious trend towards the optimum. However, they have difficulties in solving those multimodal functions who have only a few local optima with big flat plateaus, wide basins or deep valleys. EDAs are easily misled by such kinds of deceptive landscapes and then converge to poor results. Methods based on adaptive variance scaling [14]-[16] have been proposed to enhance single Gaussian model based EDAs' explorative abilities. On the other hand, Gaussian mixture model based EDAs try to estimate a complicated probabilistic model to precisely describe the real distribution to solve a problem. Once a precise estimate has been obtained, robust performance on deceptive functions can be guaranteed [11]. But the computational cost to learn a precise mixture distribution is usually expensive. Furthermore, as a branch of optimization algorithms, the aim of EDA is to use the technologies of machine learning to search and locate the global optimum in a search space, rather than to learn and obtain the real distribution. In this paper, we propose a fresh perspective to boost EDAs' performance on those hard problems for single model based EDAs.

Classical single model based EDAs are competent enough to obtain good results on simple functions. When solving complex functions, one of the ideal searching processes of EDA can be imagined as: firstly discover the promising local areas in the search space where the global optimum may lie in, and then locate the global optimum in those areas by local search. Even if we do not have a precise distribution to describe the entire search space, as long as we succeed in locating the promising areas, and in each of the areas, the probabilistic model we used (which may not be so precise, but) is adequate for a local search, the algorithm should also achieve good performance.

Inspired by this thought, in this paper, we propose a new EDA framework called NichingEDA. In NichingEDA, the entire population consists of several separated subpopulations. Each subpopulation is a single model based EDA, named as a subEDA, which also contains a population of individuals, each of which presents a solution in the search space. The collection of several subEDAs as a whole system evolves itself in a classical evolutionary algorithm (EA) manner. Each subEDA interacts and cooperates with each other through resource sharing control (niching selection in this paper), crossover and mutation to explore and exploit the search space. Obviously, NichingEDA utilizes two levels of diversities, i.e., not only the diversity at the subpopulation level, but also the diversity at the population of subpopulations level.

W. Dong is with the Key Laboratory for Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, PR China (e-mail: weishan.dong@ia.ac.cn).

X. Yao is with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK (e-mail: x.yao@cs.bham.ac.uk).

Similar approaches, such as neural network ensembles [17] and Evolutionary Ensembles with Negative Correlation Learning (EENCL) [18] in the evolutionary learning field, GA Ensemble and PBIL Ensemble [19] in the discrete binary coded EA field, have made remarkable advantages on boosting the overall performance of a system. NichingEDA first expands the idea of ensemble method to the continuous EDA field, but note that it is not just a simple extension from a discrete EA ensemble to a continuous one. Searching in infinite continuous search space is more challenging than in finite discrete search space. NichingEDA employs resource sharing control techniques to emphasize the diversity of subpopulations, which is a very important issue for ensemble system and has not been investigated in previous GA Ensemble and PBIL Ensemble.

The distributed island-based EDAs [20] also evolve several subpopulations in parallel. The difference between island-based EDAs and NichingEDA can also be clearly analyzed. Island-based EDA is extended from the island GA [21], but the migration of individuals from one island (subpopulation) to another is replaced by the transfer of model parameters. In island-based EDAs, subpopulations exchange information through a predefined and fixed topological structure (e.g., star or ring), and they are not restricted to local search. Whereas subEDAs in NichingEDA interact with each other as if they are "individuals" of a classical EA. Moreover, subEDAs try to exert their diversities to cover different "local" areas. In a word, subpopulation plays a different role in island-based EDAs and NichingEDA, and NichingEDA operates more like an ensemble system.

NichingEDA can also be regarded as a combination of EA and EDA to balance between exploration and exploitation. Similar attempts of hybridizing EDA with GA, DE or other techniques [22]-[24] have achieved good progress. Here NichingEDA offers a different way of hybridization to previous approaches and also shows its effectiveness.

The subEDAs in the NichingEDA framework can be applied with arbitrary forms of probabilistic model. To compare NichingEDA with classical continuous EDAs, we apply multivariate Gaussian distribution to the subEDAs in our experiments. Through making good use of the diversity of a population of subEDAs, NichingEDA shows significant advantages on solving those hard problems for single model based EDAs. Although without precise estimate of distribtuion, NichingEDA's performance is also comparable to representatives of the Gaussian mixture model based EDAs. NichingEDA even shows better stabilities on benchmark functions. According to the No Free Lunch Theorem [25], the limitations of NichingEDA are also analyzed.

The remainder of this paper is organized as follows. In Section II, we present the NichingEDA framework. We also propose the evolutionary operators for subEDAs if the Gaussian model is adopted. Experimental studies are given in section III. Our final conclusions are drawn in section IV along with future work.

II. Boosting Continuous EDA's Performance By A Population of SubEDAs

A. NichingEDA Framework

By using a typical evolutionary algorithm framework, the major steps of NichingEDA framework can be summarized as follows.

1) **Initialization**: Generate a population of N subEDAs randomly in EDA space. Set $epoch := 0$.
2) **Evaluation**: Partially evolve each subEDA for a fixed number of generations. Evaluate each subEDA by its best solution found in the partial evolution.
3) **Selection**: Using a niching method to select $R \leq N$ subEDAs to make up a parents set $\mathcal{P}$.
4) **Elitists Preservation**: Select best $E \leq R$ elitist subEDAs from $\mathcal{P}$.
5) **Reproduction**: Generate $N - E$ offspring subEDAs using $\mathcal{P}$. Make up a new population of subEDAs by E elitists and $N - E$ offsprings. Set $epoch := epoch + 1$.
6) **Termination**: Stop if some stopping criterion is reached, go to Step 2) otherwise.

There are two levels of evolution in NichingEDA: partial evolution of individuals in the real search space at the subEDA level, and evolution of subEDAs in the EDA space at the EDA population level. Details about each component of NichingEDA are given in the following sections.

B. Representation of SubEDAs

Note that the initialization, crossover and mutation of subEDAs are all carried out in the EDA space, not the real search space. Each of the subEDAs corresponds to a specific probabilistic model, and such a model can be represented by several parameters. Thus we can immediately code a subEDA in the EDA space by its probabilistic model parameters. E.g., a Gaussian model can be fully described by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. Thus we can use $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to code a single Gaussian based subEDA. With a logical representation of subEDAs, during each epoch, evaluation, selection and reproduction can be carried out in a typical EA manner. Please note that NichingEDA has the assumption that each subEDA is competent for local search. This basic assumption must be satisfied when choosing subEDAs.

C. Initialization of SubEDAs

At the initial epoch, all the subEDAs should be initialized to cover different areas in the search space. The initialization can be implemented by sampling the model parameters of each subEDA within their respective feasible boundaries. E.g., for a Gaussian based subEDA, its initial mean $\boldsymbol{\mu}$ can be sampled uniformly within the problem domain, and its initial covariance matrix $\boldsymbol{\Sigma}$ can be set as an diagonal matrix $\boldsymbol{D}$ which indicates the initial Gaussian covers an appropriate portion of search space. Formally, the initialization of Gaussian based subEDA population can be stated as follows:

Initialization: For subEDA$_i, i \in \{1, 2, \ldots, N\}$, do:

1) Uniformly sample $\boldsymbol{\mu}_i$ in the search space.
2) Set $\boldsymbol{\Sigma}_i := \boldsymbol{D}$.

In NichingEDA, the non-zero elements in $\boldsymbol{D}$ which equals to the initial variances should not be too big for subEDAs should mainly focus on local areas, while the explorative behaviors are primarily carried out at the EDA population level. Simultaneously, they ought not to be too small, otherwise the subEDA will lose its power of utilizing population diversity and perform just like a single point attempt. The efficiency of local search will also be diminished. As we will see, well balancing the behaviors between the two levels of evolution can achieve significant results. And it starts from a rational initialization method.

So far, we have initialized the models, i.e., the probability density functions of subEDAs, but not their subpopulations. Evolution at the subpopulation level is to be carried out in the partial evolutionary process.

D. Evaluation: Partial Evolution of SubEDAs

According to previous analysis, we have the basic assumption that each subEDA in the EDA population is good at searching a local area. If the local area a subEDA covers contains potential global optima, the subEDA should exhibit a significant fitness increasing rate. For a given period of self-evolving, the subEDA ought to find better solution. In NichingEDA, we set all the subEDAs have the same population size N_{sub} and selected size R_{sub}. For subEDA$_i, i \in \{1, 2, \ldots, N\}$, its probability density function (pdf) is denoted as p_i, and its fitness is denoted as f_i. A constant value, $MaxGen_{sub}$, denotes the maximum number of generations a subEDA can evolve in one epoch. Partial evolution evaluation, i.e., the Step 2) of NichingEDA is shown as follows.

Evaluation: For subEDA$_i, i \in \{1, 2, \ldots, N\}$, do partial evolution:

```
1) $generation := 0$.
2) While $generation < MaxGen_{sub}$, do:
    a) Sample $N_{sub}$ individuals from current pdf $p_i$.
    b) Select $R_{sub} \leq N_{sub}$ individuals according to a selection method.
    c) Estimate/update $p_i$ based on $R_{sub}$ selected individuals.
    d) $generation := generation + 1$.
3) $f_i :=$ The fitness of the best solution found.
```

Specifically, if a multivariate Gaussian is used in subEDAs, then p_i should be the multivariate normal pdf of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. In a typical multivariate Gaussian based EDA, e.g., EMNA$_{global}$[1], the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are computed by the maximum likelihood estimates.

After subEDAs partially evolve themselves independently, we have the fitness of each subEDA. These steps are almost the same as the main loop of a classical continuous EDA, except that the evolution is restricted to a pre-fixed number of generations. Thus it is called a "partial evolution". Because we do not need to fully evolve a subEDA until it converges if its f_i increases too slow which indicates the local area is not a promising one. We ought to save those function evaluations and use them to explore new untested areas. For those good subEDAs we can use selection and elitist preservation to ensure their survival and continuous searching those local areas.

E. Niching Selection

As an individual in the EDA population, a subEDA itself also contains a collection of individuals. If the distributions of individuals of two subEDAs are overlapping each other, or even one is superposed by the other, obviously, both of the subEDAs will test a same area and waste multiple similar attempts. By adopting niching methods which have been developed in traditional GA field for the purpose of identifying multiple optima [26], we can prevent the crowding of subEDAs, and promote the explorative ability.

1) Clearing: The niching method we chose in NichingEDA is the clearing procedure [27], which is regarded as an effective approach having advantages over the fitness sharing and other niching methods [28].

Clearing procedure is based on the concept of limited resources of the environment and can be easily applied to NichingEDA framework. In clearing, the capacity k of a niche (a local area) specifies the maximum number of subEDAs that this niche can accept. Without loss of generality, we assume that $k \leq N$, the EDA population size. Clearing preserves the fitness of the k best subEDAs (dominant subEDAs) of the niche and resets the fitness of the others (dominated subEDAs). SubEDAs belong to the same niche if their distance in the EDA space or the search space is less than a dissimilarity threshold (clearing radius) θ. Clearing can also be coupled with elitism to preserve the best subEDAs of the niches during the evolution.

In NichingEDA, clearing also serves as a selection method. Those dominant subEDAs who survive after clearing constitute the parents set $\mathcal{P}$. Thus in NichingEDA with clearing selection, the selected size $R = |\mathcal{P}|$, is an integer variable whose range is $[k, N]$ rather than a constant.

2) Measuring the Similarity Between SubEDAs: Niching methods require a distance metric to describe the similarity/dissimilarity between two individuals. Such a distance can be characterized based on either genotypic or phenotypic similarity. In NichingEDA, we have to propose a method to measure the distance d_{ij} between two subEDAs, subEDA$_i$ and subEDA$_j$, where $i, j \in \{1, 2, \ldots, N\}, i \neq j$. Here the term "distance" should not be taken rigorously. For the important point is to define the similarity, d_{ij} may not satisfy the triangle inequality, but have to be symmetric.

One feasible method is using the Euclidian distance metric in the real search space:

$$d_{ij} = \sum_{m=1}^{R_{sub}} \sum_{n=1}^{R_{sub}} \|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2, \tag{1}$$

where vector $\boldsymbol{x}_m$ is a selected individual belongs to subEDA$_i$, and vector $\boldsymbol{x}_n$ is a selected individual belongs to subEDA$_j$. Note that the pdf of a subEDA is estimated by R_{sub} selected individuals, so the computation of d_{ij} should also be based on selected individuals.

Euclidian metric is easy to implement and do not rely on the model a subEDA uses. But the individuals which are sampled according to a pdf contain random noise, whereas the information of the more reliable probabilistic model is not utilized. According to our empirical tests, in NichingEDA, using a distance metric defined in genotypic space (EDA space, or say, the probabilistic model space) is more effective than Euclidian distance metric defined in phenotypic space (search space).

The Kullback-Leibler distance [29] has been widely used to measure the similarity of two distributions. Taking Gaussian distributions for instance, we can define d_{ij} by the Kullback-Leibler distance to compute the corresponding distance between Gaussian model based subEDAs.

Suppose two Gaussian distributions $\mathcal{N}_0 : (\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $\mathcal{N}_1 : (\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, their Kullback-Leibler distance is given by:

$$\begin{aligned} D_{KL}(\mathcal{N}_0 \| \mathcal{N}_1) &= \frac{1}{2}(log\left(\frac{det\boldsymbol{\Sigma}_1}{det\boldsymbol{\Sigma}_0}\right) + tr(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) \\ &+ (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - n), \end{aligned} \quad (2)$$

where n denotes the number of variables.

As we see, the Kullback-Leibler distance is not symmetric. Thus we prefer to define d_{ij} in a popular way [30] to make it symmetric:

$$d_{ij} = D_{KL}(\mathcal{N}_i \| \mathcal{N}_j) + D_{KL}(\mathcal{N}_j \| \mathcal{N}_i). \quad (3)$$

Besides, we should set a dissimilarity threshold θ to couple with the clearing selection. If $d_{ij} < \theta$, we suppose subEDA$_i$ and subEDA$_j$ belong to a same niche.

F. Elitist Preservation

In NichingEDA, we adopts the recommended elitism with clearing that to preserve only the winners of clearing with a fitness above the average of the entire population before clearing [27]. In addition, the maximum number of elitists is controlled by a parameter E. We only survive at most E best elitists to the next epoch to leave enough room for new offspring EDAs in the population.

NichingEDA uses the partial evolution along with this elitism mechanism to balance the exploration and exploitation. Once a subEDA succeed in locating a promising local area, we should ensure that it survives to the next epoch to continue exploiting that area. Elitists preservation is necessary and important for assurance of the exploitive performance of NichingEDA.

G. Reproduction

The reproduction step is carried out by first pick out two parent subEDAs from $\mathcal{P}$ through sampling without replacement, then recombine their probability model to reproduce one offspring subEDA, and at last mutate the offspring. Because NichingEDA is designed to solve those hard, deceptive problems for classical single model based EDAs, to maintain the diversity of the EDA population, we use uniform sampling to let each subEDA in P has the same probability of $1/R$, where $R = |\mathcal{P}|$, to be chosen as a parent.

The designs of crossover and mutation operators are related to the model we chose at the subEDA level. Whatever model we use, reproduction of new offspring EDAs should be performed in the genotypic space (EDA space).

Again we take Gaussian model as an example. Suppose two subEDAs, $(\boldsymbol{\mu}_{\text{p1}}, \boldsymbol{\Sigma}_{\text{p1}})$ and $(\boldsymbol{\mu}_{\text{p2}}, \boldsymbol{\Sigma}_{\text{p2}})$ have been chosen as parents, their offspring $(\boldsymbol{\mu}_{\text{off}}, \boldsymbol{\Sigma}_{\text{off}})$ can be generated in the following steps.

Reproducing An Offspring:

1) Randomly sample a real number $\delta \in [0, 1]$.
2) Crossover two parents by linear combination.
   a) $\boldsymbol{\mu}_{\text{off}} := \delta\boldsymbol{\mu}_{\text{p1}} + (1 - \delta)\boldsymbol{\mu}_{\text{p2}}$.
   b) $\boldsymbol{\Sigma}_{\text{off}} := \delta\boldsymbol{\Sigma}_{\text{p1}} + (1 - \delta)\boldsymbol{\Sigma}_{\text{p2}}$.
3) Mutate $\boldsymbol{\mu}_{\text{off}}$.
   a) Uniformly sample a vector $\boldsymbol{v}$ in the search space.
   b) Randomly sample a real number $\eta \in [0, 1]$.
   c) $\boldsymbol{\mu}_{\text{off}} := \eta\boldsymbol{\mu}_{\text{off}} + (1 - \eta)\boldsymbol{v}$.
4) Mutate $\boldsymbol{\Sigma}_{\text{off}}$.
   a) Randomly sample a real number $\zeta \in [0, 1]$.
   b) $\boldsymbol{\Sigma}_{\text{off}} := \zeta\boldsymbol{\Sigma}_{\text{off}} + (1 - \zeta)\boldsymbol{D}$, where $\boldsymbol{D}$ is the same as in initialization step.

As we can see, in order to exert the explorative abilities of evolutionary operators, we put sufficient randomness into the mutation steps. This is consistent with the initial inspiration of NichingEDA that to use subEDAs for local search and to explore the search space by recombination to make use of the diversity inside the EDA population.

III. Experimental Studies

A. Algorithm Implementation

1) Selection of SubEDA: To compare NichingEDA with exist classical EDAs, in all the experiments, we use Gaussian model based subEDAs. Several previously developed Gaussian model based EDAs, including single model based EDAs and mixture model based ones are included in comparisons. But it should be emphasized that not only Gaussian model, but also any other feasible probabilistic models can be used in NichingEDA framework, as long as the basic assumption that a subEDA is good at local search can be satisfied.

Because evolving N subEDAs simultaneously makes NichingEDA spend function evaluations N times faster than one classical EDA, in order to draw a fair performance comparison in terms of function evaluation cost, we set subpopulation size N_{sub} of each subEDA to a small number. Previous studies [12][14] and empirical results show that when the population size of an EDA is too small, the algorithm performance will be fast deteriorating. Considering the basic assumption of NichingEDA, we choose EEDA [31] to be the subEDA component in all the experiments.

EEDA can be seen as a variant of classical multivariate Gaussian based EDAs. Its only difference to the classical ones is the minimum eigenvalue of the covariance matrix will be resetted to the value of the maximum eigenvalue after the maximum likelihood estimation to have an approximation to the negative gradient of the fitness function. Experimental study [32] indicates that EEDA has more robust performance than maximum likelihood estimated Gaussian based EDAs with small population sizes. By using EEDA as subEDA, we can maintain the multivariate searching ability and simultaneously make the performance of subEDAs do not be worsen too much considering the small subpopulation size N_{sub} we can use.

In all the following experiments, each subEDA is an EEDA with truncation selection and elitism. For our aim is to use subEDAs for exploitive search rather than explorative search, to make them have a faster convergence rate, we prefer a small selected size which reflects an intensive selection pressure. Considering the numerical problem of multivariate Gaussian based EDAs [32][33], we use a covariance matrix repairing method, ECMR0 [32], along with EEDA to remove the computation errors. But note that ECMR0 does not alter the searching behaviors of EEDA. It just helps to repair the computation error which may cause numerical problems and lead to unexpectable breakdown of those multivariate Gaussian based EDAs.

2) Parameter Settings: Through all the experiments, we set the same parameters for NichingEDA. As a result of choosing EEDA as the subEDA, all the steps in NichingEDA, including initialization, evaluation, reproduction, etc., are all implemented by the proposed approaches in previous sections which are designed for Gaussian models. We use distance metric described in (3), pre-defined niching capacity k and dissimilarity threshold θ in the clearing selection. All the settings are listed in Table I.

TABLE I
SETTINGS OF NICHINGEDA

Parameter	Value
N	100
R	Automatically determined in clearing.
E	20
subEDA	EEDA with ECMR0
N_{sub}	100
R_{sub}	25
$\boldsymbol{D}$	$(\frac{u-l}{30})^2\boldsymbol{I}$ *
$MaxGen_{sub}$	5
k (Niching capacity)	5
θ (Dissimilarity threshold)	10^3

* where u and l are the upper bound and the lower bound, respectively, of the problem domain. $\boldsymbol{I}$ is an identity matrix.

B. NichingEDA vs. Single Model Based EDAs

The first comparison is made between NichingEDA and three classical single Gaussian model based EDAs, which are UMDA$_c^G$[1], EMNA$_{global}$[1] and EEDA itself as a single algorithm. They employ different model hypothesis, but are all based on a Gaussian model. To prevent previously mentioned numerical problems of multivariate Gaussian model, we use the implementations EMNA$_{global}$-CMR [33] and EEDA with ECMR0 [32] on behalf of the original EMNA$_{global}$and EEDA.

We use a same setting for all the three EDAs through all experiments. Their population sizes are set to 2000, and a truncation selection size of 1000. Their initial populations are sampled randomly within the function domain, and elitism is used in evolution. All these settings are commonly used in previous literatures when studying these algorithms ([1], [7], [13], [31]-[33]).

We select five test functions from [34], including f_1 (Sphere function), f_{10} (Ackley function), f_{14} (Foxholes function), f_{15} (Kowalik function) and f_5 (Rosenbrock function). All these functions are minimization problems. More detailed descriptions of these functions can be found in [34]. The Sphere function is a simple unimodal function. The Ackley function is a multimodal function with many local optima, but its global landscape has an obvious and regular trend towards the global optimum. These two functions are widely used in studying optimization algorithms. Single Gaussian model based EDAs can solve them efficiently. While for the Foxholes, Kowalik and Rosenbrock functions, the shapes of the landscape are somehow weird. The Foxholes function has 25 deep holes, among which the global optimum is hard to differentiate from the other local optima. The Kowalik function and the Rosenbrock function both have a big flat plateau and contain basins and valleys. They are supposed to be hard for single Gaussian based EDAs.

We set a maximum number of evaluations for each functions. All the tested algorithms are stopped when their evaluation numbers reach this limit. The settings of each function are listed in Table II.

TABLE II
SETTINGS FOR TEST FUNCTIONS IN COMPARING NICHINGEDA WITH SINGLE MODEL BASED EDAS

Function	Dimension	Domain	Optimum	Max. Eval
Sphere	10	$[-100, 100]$	0	5×10^5
Ackley	10	$[-32, 32]$	0	5×10^5
Foxholes	2	$[-65.536, 65.536]$	0.9980	5×10^5
Kowalik	4	$[-5, 5]$	0.0003075	5×10^6
Rosenbrock	10	$[-30, 30]$	0	5×10^6

The experimental results are summarized in Tables III-VII. All the results have been averaged over 30 independent runs. The best values, mean values and standard errors are computed based on the best solutions that an algorithm have found in multiple runs.

From the results we can see classical single Gaussian based EDAs indeed have good performance on Sphere and Ackley functions. While the convergence rate of NichingEDA is slower in terms of the function evaluations cost, because NichingEDA keeps on testing new areas in the search space even if it has already located the only "right" local area. However, NichingEDA successfully exhibits its initial inspiration to perform the best in all the three hard problems for single Gaussian based EDAs. It reaches the global optimum in all the runs of the Foxholes function, and almost all the runs on the Kowalik function. It also shows faster convergence rate than the single Gaussian based EDAs on the Rosenbrock function. Whereas single Gaussian based EDAs are easily misled by the shapes of these functions to converge to local optima. The comparisons clearly show that NichingEDA makes good use of the diversity of an EDA population and overcome the shortages of single model based EDAs. On the other hand, NichingEDA may converges slower on those simple functions that single model based EDAs are competent for.

C. NichingEDA vs. Mixture Model Based EDAs

The second comparison is made between NichingEDA and two representatives of those Gaussian mixture based EDAs, CEGNA$_{\text{BGe}}$and CEGDA [11]. These two EDAs are designed for dealing with multimodal functions by taking advantage of clustering techniques. Both algorithms employ a finite mixture of multivariate Gaussian distribution. Experimental studies have shown these two algorithms' effectiveness on solving functions TwoPeaks, ThreePeaks and Shekel, which can be hardly solved by single Gaussian based EDAs [11]. At the same time, [11] also exhibits that CEGNA$_{\text{BGe}}$and CEGDA may have difficulties on solving multimodal function that has too many local optima, e.g., Schwefel function. The details of all the above functions can be found in [11].

We borrow the results of CEGNA$_{\text{BGe}}$and CEGDA on these functions from [11] to compare with NichingEDA. The settings of the test functions are shown in Table VIII. For both CEGNA$_{\text{BGe}}$and CEGDA, the population size $= 2000$, the selected size $= 500$, $\alpha_c = 0.05$ and $\alpha_r = 0.002$. The maximum evaluation number is 4×10^5 for all the functions. The results are averaged over 30 independent runs. The settings of NichingEDA keep the same as Table I. The comparisons are shown in Tables IX-XIII.

From the results we can see on the TwoPeaks and ThreePeaks functions, NichingEDA even shows more stable performance than CEGNA$_{\text{BGe}}$and CEGDA that the standard errors are all zeros. On the Schwefel function, NichingEDA also shows better stability and achieve better results in terms of the mean best value of solutions. But on Shekel functions, opposite results can be observed between the low dimensional version and the high dimensional one. For the 4d Shekel function, NichingEDA is comparable to CEGNA$_{\text{BGe}}$and CEGDA. But for the 30d Shekel function, the performance of NichingEDA is poor. Analysis of the pool result is given in the following section.

D. When NichingEDA Might Fail and Why

The poor result of NichingEDA on 30d Shekel function in Table XII can be explained as when only $N_{sub} = 100$ individuals are sampled in a high dimensional space, and only $R_{sub} = 25$ individuals among them are used to estimate a covariance matrix, it is probable that the covariance matrix is very inaccurate and becomes an ill-conditioned matrix. In this case, the Kullback-Leibler distance metric also becomes unreliable. Its value honestly reflects the distance between two Gaussian models, but does not reflect the true distance between two subpopulations. Through logging the value of d_{ij} in a run of NichingEDA on the 30d Shekel function, we can find that d_{ij} is constantly larger than 10^{17} from the beginning of evolution to the end! It means that in clearing

TABLE III
EXPERIMENTAL RESULTS FOR THE SPHERE FUNCTION

Algorithm	Best	Mean	Std
UMDA$_c^G$	9.2923e-044	1.6529e-043	5.0208e-044
EMNA$_{global}$	5.0713e-044	8.6746e-044	2.6313e-044
EEDA	6.1299e-040	1.0708e-039	3.1585e-040
NichingEDA	3.3656e-010	4.5074e-009	9.3440e-009

TABLE IV
EXPERIMENTAL RESULTS FOR THE ACKLEY FUNCTION

Algorithm	Best	Mean	Std
UMDA$_c^G$	4.4409e-016	4.4409e-016	0
EMNA$_{global}$	4.4409e-016	4.4409e-016	0
EEDA	4.4409e-016	4.4409e-016	0
NichingEDA	5.2405e-006	2.4350e-005	1.3137e-005

TABLE V
EXPERIMENTAL RESULTS FOR THE FOXHOLES FUNCTION

Algorithm	Best	Mean	Std
UMDA$_c^G$	0.9980	1.1691	0.3406
EMNA$_{global}$	0.9980	1.0747	0.2543
EEDA	0.9980	1.0019	0.0156
NichingEDA	0.9980	0.9980	0

TABLE VI
EXPERIMENTAL RESULTS FOR THE KOWALIK FUNCTION

Algorithm	Best	Mean	Std
UMDA$_c^G$	0.00087045	0.0026	0.0010
EMNA$_{global}$	0.00066336	0.0020	7.5969e-004
EEDA	0.0010	0.0011	1.0183e-004
NichingEDA	0.00030750	0.00030874	2.7899e-006

TABLE VII
EXPERIMENTAL RESULTS FOR THE ROSENBROCK FUNCTION

Algorithm	Best	Mean	Std
UMDA$_c^G$	8.0733	8.1958	0.0374
EMNA$_{global}$	7.3978	7.7806	0.1713
EEDA	6.1079	6.3004	0.1259
NichingEDA	0.6366	4.6258	1.6986

TABLE VIII
SETTINGS FOR TEST FUNCTIONS IN COMPARING NICHINGEDA WITH MIXTURE MODEL BASED EDAS

Function	Dimension	Domain	Type	Optimum
TwoPeaks	5	$[-100, 100]$	Max.	10.1053
ThreePeaks	5	$[-100, 100]$	Max.	10.1053
Shekel (n=5)	4	$[0, 10]$	Max.	10.1033
Shekel (n=5)	30	$[0, 10]$	Max.	10.0134
Schwefel	30	$[-500, 500]$	Min.	-12569.5

TABLE IX
EXPERIMENTAL RESULTS FOR THE TWOPEAKS FUNCTION

Algorithm	Best	Mean	Std
CEGNA$_{\text{BGe}}$	10.1053	10.1053	3.55e-015
CEGDA	10.1053	10.0999	5.92e-003
NichingEDA	10.1053	10.1053	0

TABLE X

EXPERIMENTAL RESULTS FOR THE THREEPEAKS FUNCTION

Algorithm	Best	Mean	Std
CEGNA$_{BGe}$	10.1053	10.1053	3.55e-015
CEGDA	10.1053	10.1048	7.99e-004
NichingEDA	10.1053	10.1053	0

TABLE XI

EXPERIMENTAL RESULTS FOR THE SHEKEL FUNCTION WITH FIVE LOCAL OPTIMA (DIMENSION = 4)

Algorithm	Best	Mean	Std
CEGNA$_{BGe}$	10.1033	10.1033	8.8818e-015
CEGDA	10.1033	10.1033	8.8818e-015
NichingEDA	10.1033	10.1033	1.77e-007

TABLE XII

EXPERIMENTAL RESULTS FOR THE SHEKEL FUNCTION WITH FIVE LOCAL OPTIMA (DIMENSION = 30)

Algorithm	Best	Mean	Std
CEGNA$_{BGe}$	10.0134	10.0134	8.8818e-015
CEGDA	10.0134	10.0134	8.8818e-015
NichingEDA	0.3550	0.2230	0.0564

TABLE XIII

EXPERIMENTAL RESULTS FOR THE SCHWEFEL FUNCTION

Algorithm	Best	Mean	Std
CEGNA$_{BGe}$	-10773.1	-6760.35	2624.33
CEGDA	-8712.31	-5922.54	1893.51
NichingEDA	-8733.51	-8005.39	270.755

selection, none of the subEDAs is cleared by the dissimilarity threshold $\theta = 10^3$ in every epoch. So the effect of clearing selection to locate promising areas becomes ineffective. Although EEDA as a subEDA has robust performance with small population size, on specific functions such as 30d Shekel, it still requires a larger population size than just 100 to maintain a reliable estimate of a local area.

So we can say, the poor result of NichingEDA is related to the model of subEDAs, the distance metric, the problem size and also the shape of the function. All these factors have effects on the basic assumption of NichingEDA: the subEDA has to be competent for local search. If this assumption is not satisfied, performance of NichingEDA will be definitely poor. Besides, NichingEDA contains two levels of evolution, and each one requires a sufficient population size to the problem size. As a special case, when Gaussian based subEDAs are used, they require a large enough subpopulation size to obtain reliable estimate of the covariance matrix. Thus the total population size that NichingEDA needs to solve a problem grows faster than ordinary EDAs.

To prove our analysis, we run NichingEDA on 10d version of the Shekel function and hold the parameters of NichingEDA unchanged. The results are put together with the 4d and 30d results, which is shown in Table XIV.

From the results we see the trend is obvious that for fixed parameters the performance of NichingEDA decreases as the problem size grows. In accordance with previous analysis, we should also note that the divided sub search space for each subEDA to do local search should be too big to match the basic assumption of NichingEDA. However, when the problem size becomes larger, the number of such sub-spaces to be exploited also increases rapidly. The two-level structure of NichingEDA requires both large enough population sizes on the two levels of evolution. As a result, the larger the problem size is, the more probable the basic assumption of NichingEDA is not satisfied. This seems to be the primary disadvantage of NichingEDA.

Nevertheless, when solving problems which do not require large subpopulation size for local search, and at the upper level, EDA population size is adequate, NichingEDA has reliable and robust performance. By using other kinds of subEDAs instead of Gaussian based EDAs, the performance downtrend may become slower and controllable. It is possible to develop new EDAs whose performance is not so sensitive to population sizes for the purpose of local search. Then by applying them to NichingEDA framework, we may develop more robust NichingEDA approaches with good scalability. On the other hand, designing more robust subEDA distance metric which is not sensitive to subEDA population size is also beneficial.

After all, through experimental studies, the NichingEDA framework shows the effectiveness of utilizing the diversity inside a population of EDAs. On those hard problems for single model based EDAs, NichingEDA shows its significant effectiveness. Although without precise estimates of the entire distribution, NichingEDA's performance is also comparable to mixture model based EDAs as long as the basic assumption of NichingEDA is satisfied.

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a new EDA ensemble method named as the NichingEDA framework, which is designed to overcome the weakness of classical single model based EDAs. Through employing niching and recombination, an EDA population which consists of several subEDAs evolves as a whole system to solve continuous optimization problems. Experimental studies show the efficiency of utilizing the diversity inside a population of EDAs. NichingEDA successfully boosts single model based EDAs' performance on the traditional "EDA-hard problems". NichingEDA provides a new effective way to deal with multimodal functions without precisely estimating the real distribution as in those mixture model based EDAs.

As we analyzed, NichingEDA has a similar idea to classical ensemble methods [17][18]. SubEDAs interact and

TABLE XIV

RESULTS OF NICHINGEDA ON 4D, 10D AND 30D SHEKEL FUNCTIONS

Dimension	Best	Mean	Std
4	10.1033	10.1033	1.77e-007
10	10.0416	6.7310	2.3809
30	0.3550	0.2230	0.0564

cooperate with each other to solve one problem together. But note that in classical ensemble methods, not only the learing/evolutionary process is carried out by cooperation, but also the solution to a problem is usually a combination of the components. Whereas NichingEDA tries to find out the only best individual component inside a best subEDA component. Similar to the ensemble methods, the structure of NichingEDA framework makes it easy to be implemented in parallel.

Future work on NichingEDA includes designing or applying other probabilistic models at the subEDA level to further improve the performance and scalability of NichingEDA. Sound distance metric and new evolutionary operators are also to be carefully designed for new probabilistic models. Finally, in-depth analysis is needed to understand when and why NichingEDA may fail.

ACKNOWLEDGMENT

The authors would like to thank Fuxin Li, Prof. Jue Wang and Prof. Wensheng Zhang for their constructive comments. This work is supported by a grant from the Chinese Academy of Sciences for Outstanding Young Scholars from Overseas (No. 2F03B01) and an EPSRC grant (EP/C520696/1).

REFERENCES

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

[2] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," Lecture Notes in Computer Science 1411, in *Parallel Problem Solving from Nature - PPSN IV*, Springer-Verlag, London, UK, 1996, pp. 178-187.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, USA, 1989.

[4] S. Rudlof and M. Köppen, "Stochastic hill climbing with learning by vectors of normal distributions," in *Proc. 1st Online Workshop on Soft Computing (WSC1)*, Nagoya, Japan, 1996, pp. 60-70.

[5] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," in *Parallel Problem Solving from Nature - PPSN V*, pp. 418-427, Springer-Verlag, Berlin, 1998.

[6] M. R. Gallagher, M. Frean, and T. Downs, "Real-valued evolutionary optimization using a flexible probability density estimator," *GECCO-1999*, pp. 840-846.

[7] P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," *GECCO-2000*, pp. 201-204, 2000.

[8] P.A.N. Bosman and D. Thierens, "Continuous Iterated Density Estimation Evolutionary Algorithms Within The IDEA Framework," in *Proc. OBUPM Workshop at GECCO-2000*, pp. 197-200, 2000.

[9] M.R. Gallagher, M. Frean, and T. Downs, "Real-valued evolutionary optimization using a flexible probability density estimator," *GECCO-1999*, pp. 840-846.

[10] P. A. N. Bosman and D. Thierens, "Advancing continuous IDEAs with mixture distributions and factorization selection metrics," in *Proceedings of OBUPM Workshop at GECCO-2001,* San Francisco, California, U.S.A.: Morgan Kaufmann, pp. 208-212, 2001.

[11] Q. Lu, and X. Yao, "Clustering and Learning Gaussian Distribution for Continuous Optimization," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 35, no. 2, pp.195-204, May 2005.

[12] S. Kern, S. D. Müller, N. Hansen, D. Büche, J. Ocenasek, and P. Koumoutsakos, "Learning Probability Distributions in Continuous Evolutionary Algorithms - A Comparative Review," *Natural Computing,* vol. 3, no. 1, pp. 77-112, 2004.

[13] W. Dong, F. Li, and X. Yao, "An Efficient Multivariate Gaussian Based EDA Using Low-rank Cholesky Updates," *IEEE Transactions on Evolutionary Computation, Special Issue on Evolutionary Algorithms based on Probabilistic Models*, Submitted in 2007.

[14] P. A. N. Bosman and J. Grahl, "Matching Inductive Search Bias and Problem Structure in Continuous Estimation-of-Distribution Algorithms." Technical Report 03/2005, Dept. of Logistics, Mannheim Business School, 2005.

[15] J. Grahl, P. A. N. Bosman, and F. Rothlauf, "The Correlation-Triggered Adaptive Variance Scaling IDEA," *GECCO-2006*, ACM Press, New York, New York, 2006, pp. 397-404.

[16] P. A. N. Bosman, J. Grahl, and F. Rothlauf, "SDR: A Better Trigger for Adaptive Variance Scaling in Normal EDAs," *GECCO-2007*, ACM Press, New York, NY, USA, 2007, pp. 492-499.

[17] X. Yao and Y. Liu, "Making use of population information in evolutionary aritificial neural networks," *IEEE Trans. Syst., Man, Cybern.*, B., vol. 28, no. 3, pp. 417-425, 1998.

[18] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary Ensembles with Negative Correlation Learning," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 380-387, 2000.

[19] S. Zhou and Z. Sun, "Can Ensemble Method Convert a 'Weak' Evolutionary Algorithm to a 'Strong' One?," In *CIMCA'05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*, IEEE Computer Society, Washington, DC, USA, 2005, pp. 68-74.

[20] L. delaOssa, J. A. Gámez and J. M. Puerta, "Initial Approaches to the Application of Islands-Based Parallel EDAs in Continuous Domains," in *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, pp. 580-587, June 14-17, 2005.

[21] R. Tanese, "Distributed Genetic Algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 434-439, 1989.

[22] Q. Zhang, J. Sun, E. Tsang and J. Ford, "Hybrid Estimation of Distribution Algorithm for Global Optimsation," *Engineering Computation*, vol. 21, no. 1, 2004, pp. 91-107.

[23] Q. Zhang, J. Sun and E. Tsang, "Evolutionary Algorithm with Guided Mutation for the Maximum Clique Problem," *IEEE Trans. on Evolutionary Computation*, vol. 9, no.2, pp. 192-200, 2005.

[24] J. Sun, Q. Zhang and E. Tsang, "EDA+DE: New Evolutionary Algorithm for Global Optimisation," *Information Sciences* (169), 2005, pp. 249-262.

[25] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67-82, Apr. 1997.

[26] S. W. Mahfoud, *Niching Methods for Genetic Algorithms*, doctoral dissertation, University of Illinois at Urbana-Champaign, 1995.

[27] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation*, Nagoya, Japan, 1996, pp. 798-803.

[28] B. Sareni and L. Krähenbühl, "Fitness Sharing and Niching Methods Revisited," *IEEE Trans. Evol. Comput.*, vol. 2, no. 3, pp. 97-106, Sept. 1998.

[29] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp.79-86, 1951.

[30] H. Jeffreys, "An invariant form for the prior probability in estimation problems," In *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, vol. 186, no. 1007, pp. 453-461, 1946.

[31] M. R. Wagner, A. Auger, and M. Schoenauer, "EEDA: A New Robust Estimation of Distribution Algorithm," Rapport de Recherche (Research Report) RR-5190, INRIA, 2004.

[32] W. Dong and X. Yao, "Unified Eigen Analysis on Multivariate Gaussian Based Estimation of Distribution Algorithms," *Information Sciences*, 2008, Accepted.

[33] W. Dong and X. Yao, "Covariance Matrix Repairing in Gaussian Based EDAs," In *2007 IEEE Congress on Evolutionary Computation, CEC2007*, Singapore, 2007, pp. 415-422.

[34] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82-102, Jul. 1999.