An Estimation of Distribution Algorithm based on Nonparametric Density Estimation

Luhan Zhou, Aimin Zhou, Guixu Zhang
Computer Science & Technology Department
East China Normal University
500 Dongchuan Road, Shanghai, China, 200241
Email: {amzhou,gxzhang}@cs.ecnu.edu.cn

Chuan Shi
School of Computer
Beijing University of Posts and Telecommunications
10 Xitucheng Road, Beijing, China, 100876
Email: shichuan@bupt.edu.cn

***Abstract*—Probabilistic models play a key role in an estimation of distribution algorithm(EDA). Generally, the form of a probabilistic model has to be chosen before executing an EDA. In each generation, the probabilistic model parameters will be estimated by training the model on a set of selected individuals and new individuals are then sampled from the probabilistic model. In this paper, we propose to use probabilistic models in a different way: firstly generate a set of candidate points, then find some as offspring solutions by a filter which is based on a nonparametric density estimation method. Based on this idea, we propose a nonparametric estimation of distribution algorithm (nEDA) for global optimization. The major differences between nEDA and traditional EDAs are (1) nEDA uses a generating-filtering strategy to create new solutions while traditional EDAs use a model building-sampling strategy to generate solutions, and (2) nEDA utilizes a nonparametric density model with traditional EDAs usually utilize parametric density models. nEDA is compared with a traditional EDA which is based on Gaussian model on a set of benchmark problems. The preliminary experimental results show that nEDA is promising for dealing with global optimization problems.**

I. Introduction

Many real world applications involve optimizing an objective or cost function. Despite their different nature, some of them could be mathematically formulated as the following *box-constrained continuous optimization problem.*

$$\begin{array}{rl} \min & f(x) \\ \text{s.t} & x \in \Omega \end{array} \qquad (1)$$

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

In many cases, the properties, such as continuity and differentiability, of (1) are unknown or even the mathematical equation is not available. Thus, some conventional optimization techniques, for example the gradient based methods, can not be applied here. *Evolutionary algorithms (EAs)* are a kind of computational models which simulate the biology evolving process for optimization [1]. Due to its nature, EA has nowadays been widely used for tackling optimization problems with bad mathematical properties.

Under the framework of EA, some new techniques, such as *particle swarm optimization* [2], *differential evolution* [3], *artificial immune system* [4], *etc.*, have been developed in recent years. Among them *probabilistic model based evolutionary algorithm* is a new computing paradigm in evolutionary computation. The main feature of these algorithms is that they do not use traditional crossover or mutation operators to generate new solutions. Instead, they explicitly extract global statistical information from their previous search and build a probability distribution model of promising solutions. Based on the extracted information, new solutions are sampled from the model thus built. Compared to traditional EA methods, they emphasize on population distribution information rather than individual location information. The following methods share the above basic ideas and they differ from each other on origins.

- *Ant colony optimization (ACO)* was introduced by Dorigo in 1992 [5]. ACO takes inspiration from the behavior of real ant colonies and is used to solve optimization problems. Ants deposit pheromone on the ground in order to mark some favorable paths that should be followed by other members of the colony. ACO exploits a similar mechanism by constructing a probability matrix, named pheromone model, to denote the probability to choose an edge in a graph and thus sampling new solutions.
- *Cross entropy (CE)* method was proposed by Reuven Rubinstein in 1997 [6]. CE originates from the field of rare event simulation involving the estimation of parameters for a number of probability distributions associated with some rare events. CE methods iteratively generate sample points from the probability model and update model parameters on the basis of the data.
- *Quantum-inspired genetic algorithm (QGA)* was first proposed by Han and Kim in 2000 [7]. QGA simulates the quantum mechanism and uses a Q-bit vector to represent a solution. The Q-bit vector actually denotes probability distributions of all Q-bit to be 0 or 1. A quantum gate is used to generate new individuals.
- *Estimation of distribution algorithm (EDA)* was introduced by Mühlenbein and Paaß in 1996 for the first time [8]. Most of the EDAs aim to discover the vari-

able linkage information from the population to benefit offspring generation. To this end, different models with univariate, bivariate and/or multivariate variable linkages are widely studied [9]. Depending on the models used, EDAs are suitable for both combinatorial and continuous optimization [10], [11].

The main issues in the above methods include (a) model selection before executing the algorithm, and (b) model building and sampling in the running process. The probabilistic model plays a key role. Although, probabilistic model based evolutionary algorithms have achieved a great success in both the theory analysis [12], [13] and applications [14], in most of these algorithms the probabilistic models have to be selected before the executions. If the properties of the problems are not known, which is the basic assumption of EAs, it is hard to decide which model is proper for the given problems. It will be shown later in this paper that a wrong model will mislead the search process and thus result in a bad performance. The experience of the algorithm designers is critical in such cases.

Like the parametric density models used in most of probabilistic model based EAs, nonparametric density models have also been widely studied to estimate densities [15]. Comparing to the parametric density models, nonparametric density models could describe probability density function for which the functional form is not specified in advance, but which depends on the data itself. Some widely used nonparametric density estimation models include the histogram models, the kernel Parzen models, and the K-nearest-neighbour model. These models have been applied to EDAs recently. In [16], a univariate histogram model is used to extract the population distribution information. In [17], a Parzen-based method is applied to tackle multiobjective optimization problems.

In this paper, we try to use probabilistic models in a different way. For this end, a *nonparametric estimation of distribution algorithm (nEDA)* is proposed. The basic idea of nEDA is that it firstly generates a set of candidate points, and then find some to be offspring solutions by using a nonparametric density estimation method to estimate the density of each candidate point. A Parzen-based model is utilized in nEDA to estimate the densities. The major advantage of nEDA is that the form of the probabilistic model is not selected before executions.

The rest of the paper is organized as follows. Section II describes the basic idea of nEDA and the details to implement it. Section III introduces a comparison algorithm based on Gaussian distribution model. Section IV presents the experimental results of nEDA and its competitor. Some more discussions on nEDA are given as well. Finally, Section V concludes this paper and outlines some topics for further research along this line.

II. Nonparametric EDA

A. Basic Idea

As the algorithm runs, the density of the search space is positively correlated to the solution quality, *i.e.,* more points are around the high quality solutions. Thus, the density of the search space could be used to guide the search. This might be a very basic assumption in EDAs.

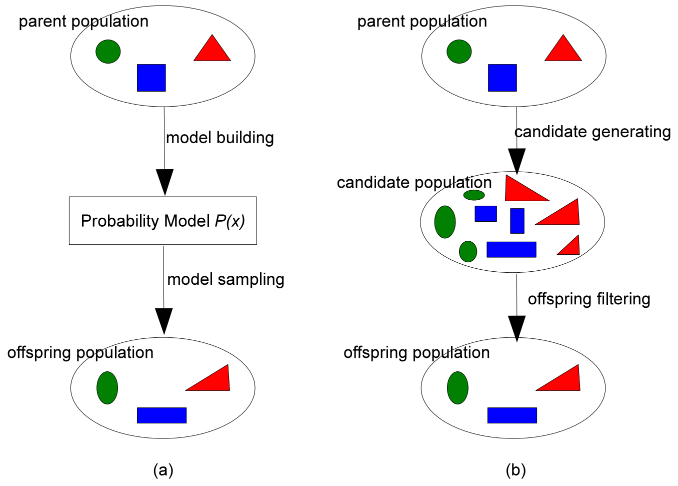

Fig. 1. Illustration of the Offspring Reproduction Process for (a) a generic EDA, and (b) nEDA

In a generic EDA, the density of the search space is estimated by a probabilistic model explicitly. By using this strategy, the model selection and building are crucial to design an EDA. In the fields of statistical and machine learning, there is another kind of techniques, called nonparametric density estimation, which can estimate density but without an explicit probabilistic model. In this paper, we propose to use nonparametric density estimation to guide the search. Unlike a generic EDA which first builds a probabilistic model and then samples new solutions, the proposed nEDA first samples some trial points around each solution and then chooses one as its offspring by considering the density values of the trial points.

Fig. 1 illustrates the difference between a generic EDA and nEDA in offspring reproduction.

B. Algorithm Framework

In each generation, the proposed nEDA maintains

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

The framework of nEDA is shown in Algorithm 1. We would like to explain the algorithm as follows.

- *Initialization:* The initial population is uniformly randomly sampled from the search space in *Line 1*.
- *Termination Condition:* Like other EAs, the algorithm terminates when a given maximum number of generations, $MaxIter$, is reached in *Line 2*.
- *Selection Procedure:* It should be noted that the selection procedure is performed in *Lines 8-10* in which the offspring is compared with its parent and the better one will survive into the next generation.
- *Reproduction Procedure:* The reproduction procedure can be divided into three sub-procedures: the generating pro-

Algorithm 1: Procedure of Nonparametric EDA

```
1  Initialize a set of solutions $\{x_1, \cdots, x_N\}$ and evaluate them;
2  while not terminate do
3      foreach $x \in \{x_1, \cdots, x_N\}$ do
4          Generate $M$ trial points $y_1, \cdots, y_M$ around $x$;
5          Estimate the density of the trial points: $P(y_1), \cdots, P(y_M)$;
6          Select a candidate $y^* \in \{y_1, y_2, \cdots, y_M\}$ as the offspring of $x$ based on their density values;
7          Evaluate $y^*$;
8          if $f(y^*) < f(x)$ then
9              Replace $x$ by $y^*$;
10         end
11     end
12 end
```

cedure in *Line 4*, the density estimating procedure in *Line 5* and the offspring filtering procedure in *Line 6*. These sub-procedures will be discussed with detail in the following sections.

C. Candidate Generating

Unlike generic EDAs which sample some new solutions from probabilistic models, nEDA first generates some candidate points around each solution for further selection. Thus, the candidate points should be 'like' their parent but not be the same as their parent, *i.e.* the candidates should be close but not too close to their parent.

To this end, in this paper we generate a candidate set $\{y_1, y_2, \cdots, y_M\}$, where M is an algorithm parameter, for each parent x by sampling from a Gaussian distribution as

$$y_i \sim N(\mu, \Sigma)$$

where $i = 1, 2, \cdots, M$, μ is the mean, and Σ is the covariance matrix, which is a diagonal matrix, of the Gaussian distribution.

Let

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

and

$$x^* = \arg\min_{i=1,\cdots,N} f(x_i).$$

The diagonal element $(\sigma^i)^2$ of Σ is sampled from another Gaussian distribution as

$$\sigma^i \sim N(|x^i - x^{*,i}|, |x^i - \bar{x^i}|^2)$$

where $i = 1, 2, \cdots, n$. In this equation, the uncertainty of the candidates is related to the distances between the parent x and the best and the mean center points of the current population.

D. Density Estimating

There are many nonparametric density estimation methods in the fields of statistical and machine learning. For a Parzen window method [18], the density estimation is usually calculated as

$$P(y) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{w}\varphi\left(\frac{||y - x_i||}{w}\right)\right).$$

By considering the contributions of objective values, we propose a modified density estimation function as follow

$$P(y) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1/\hat{f}(x_i)}{\sum_{j=1}^{N} 1/\hat{f}(x_j)}\frac{1}{w}\varphi\left(\frac{||y - x_i||}{w}\right)\right),$$

where

- w is the window width and it is estimated as
$$w = \left(\frac{1}{n}\sum_{j=1}^{n}\left(\bar{a}^j - \underline{b}^j\right)^2\right)^{1/2}$$
where $\bar{a}^j = \arg\max_{i=1,\cdots,N} x_i^j$ and $\underline{b}^j = \arg\min_{i=1,\cdots,N} x_i^j$;
- $\hat{f}(x_i) = f(x_i) + 10^{-50} > 0$ is a modification of the objective function and it is assumed that $f(x_i) \geq 0$ for all the test instances used in this paper;
- $\varphi(u)$ is a window function which is defined as
$$\varphi(u) = \frac{1}{\sqrt{2\pi}}e^{-u^2/2}.$$

It should be noted that (1) the window size is not fixed but changes as the running process, and (2) the contribution of each point to the density is not the same, and the ones with high quality contribute more to the density by using $\frac{1/\hat{f}(x_i)}{\sum_{j=1}^{N} 1/\hat{f}(x_j)}$ as the weight of the window function.

E. Offspring Filtering

Based on the density values of all the candidate points, a point will be selected as the offspring of a parent solution. In this paper, the one with the biggest density value is selected as

$$y^* = \arg\max_{y \in \{y_1, \cdots, y_M\}} P(y).$$

III. A Generic EDA for Comparison

To estimate the performance of nEDA, a generic EDA is used for comparison study. The following multivariate Gaussian model is thus used as the probabilistic model in the algorithm.

$$x \sim N(\mu, \Sigma).$$

In model building process, the mean is calculated as

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

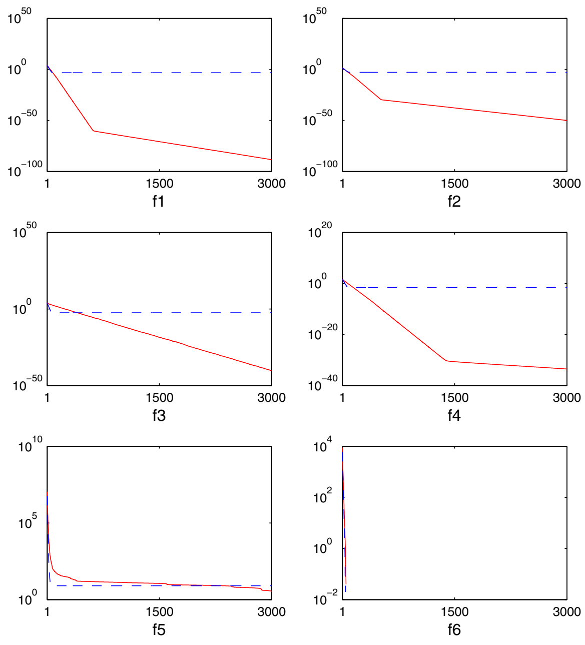

Fig. 2. The average objective values versus generations on f_1-f_6: the solid lines are for nEDA and the dash lines are for mGaussian EDA.

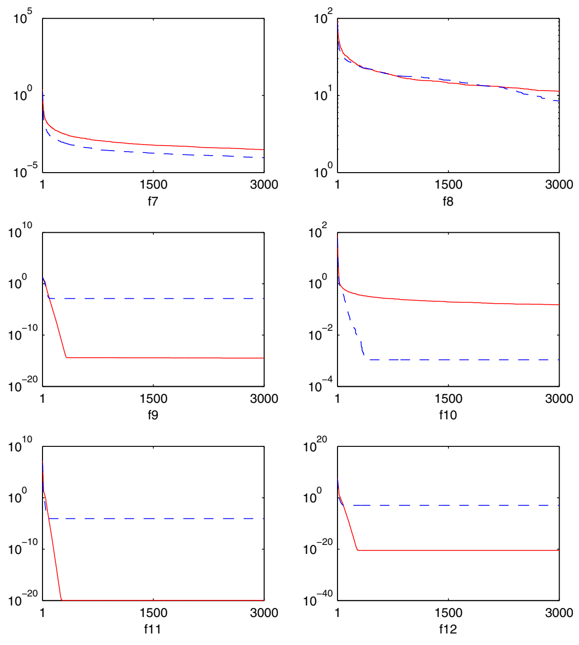

Fig. 3. The average objective values versus generations on f_7-f_{12}: the solid lines are for nEDA and the dash lines are for mGaussian EDA.

and each element $\Sigma^{i,j}$ of the covariance matrix Σ is calculated as

$$\Sigma^{i,j} = \frac{1}{N-1}\sum_{k=1}^{N}(x_k^i - \mu^i)(x_k^j - \mu^j)$$

where $i, j = 1, 2, \cdots, n$.

We denote the algorithm using multivariate Gaussian model as *mGaussian EDA*. This algorithm could be regarded as a simplified version of the *estimation of multivariate normal distribution algorithm (EMNA)* [19].

IV. Experiments

A. Test Instances and Parameter Settings

The performance of nonparametric is assessed by 12 widely used test instances from [20]. The details of these test instances are listed in Table I.

The parameters for experimental study are as follows.

- The dimension of the test instance is $n = 10$.
- The population size for two algorithms is $N = 100$.
- The maximum number of generation is $MaxIter = 3,000$.
- The number of candidates in nEDA is $M = 10$.
- The experimental results are based on 50 independent execution of the two algorithms for each instance.

B. Experimental Results

nEDA is compared with mGaussian EDA on the 12 test instances listed in Table I. The statistical results of mean and std. values of the two algorithms after 1500 and 3000 generations are shown in Table II. The run time performance of the two algorithms are drawn in Figs. 2 and 3.

From Table II, we can draw some conclusions as follows.

- The statistical values of 1500 generations are very consistent with those of 3000 generations.
- It is clear that on f_1, f_2, f_3, f_4, f_9, f_{11}, and f_{12}, mGaussian EDA was trapped in local optima. On the contrary, nEDA could tackle these problems successfully.
- It is very strange that even on f_1, which is the simplest one among the 12 test instances, mGaussian EDA failed. For this problem, the multivariate Gaussian model should be the best probabilistic model to fit it. The reason might be that the population size, which is 100 in the experiments, is not enough to build an accurate multivariate Gaussian model.
- Only on f_6, both nEDA and mGaussian EDA could find the optima exactly.
- On f_5, f_8 and f_{10}, neither nEDA nor mGaussian EDA could tackle them successfully. The two algorithms performed more or less similar.

Except the above conclusions, Figs. 2 and 3 show more information: for most of the test instances, nEDA converged quickly and the results did not change much after 500 generations. This might be the reason why it failed on f_9 and f_{10}. It is, however, a good property for other problems like f_2 and f_4. How to balance the convergence and diversity of nEDA is worth for future research.

TABLE I
TEST INSTANCES USED IN COMPARISON STUDY

Test Instance	Search Space
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$f_2(x) = \sum_{i=1}^{n} \|x_i\| + \prod_{i=1}^{n} \|x_i\|$	$[-10, 10]^n$
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^n$
$f_4(x) = \max \{\|x_i\|\}$	$[-100, 100]^n$
$f_5(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-30, 30]^n$
$f_6(x) = \sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor^2$	$[-100, 100]^n$
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + rand[0, 1)$	$[-1.28, 1.28]^n$
$f_8(x) = \sum_{i=1}^{n} \left[x_i^2 - 10 \cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]$
$f_9(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$
$f_{10}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$
$f_{11}(x) = \frac{\pi}{n} \left\{10 \sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10 sin^2(\pi y_i + 1)] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^n$
$f_{12}(x) = 0.1 \left\{sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ where $u(\cdot)$ is the same as in f_{11}.	$[-50, 50]^n$

C. Scalability of Nonparametric EDA

In the above section, nEDA was compared with mGaussian EDA on problems with variable dimension $n = 10$. It might be interesting to investigate the scalability of nEDA. For this purpose, nEDA is applied to the 12 instances with dimension $n = 10$, 20, and 30. The mean values of nEDA over 50 runs for the test instances after 3000 generations are shown in Table III.

It is for sure that the performance decreases as the dimension increases, which is shown by most of the test instances. However, for some problems like f_{10} and f_{11}, the results of the problems with higher dimensions are better than those with lower dimensions. The reason might be that nEDA is not stable on these problems and few bad runs influenced the statistical results very much.

On f_1, f_2, f_6, f_9, f_{11} and f_{12}, the scalabilities are good when $n \le 20$. When $n = 30$, nDEA does not perform well for all the 12 test instances.

D. Number of Trial Points of Nonparametric EDA

The trial points are sampled from their parent and one of them will be chosen as the offspring. Thus, it is necessary

TABLE II
MEAN AND STD. VALUES OF THE TWO ALGORITHMS WITH GENERATION=1500 AND 3000 OVER 50 RUNS FOR ALL THE TEST INSTANCES.

	$generation = 1500$		$generation = 3000$	
Instance	nEDA	mGaussian EDA	nEDA	mGaussian EDA
f_1	**1.583e-71**±1.235e-71	3.681e-04±1.251e-03	**3.380e-89**±3.386e-89	3.681e-04±1.251e-03
f_2	**1.563e-38**±6.869e-39	3.994e-03±1.539e-02	**8.816e-51**±6.125e-51	3.994e-03±1.539e-02
f_3	**3.137e-19**±8.726e-19	6.188e-04±3.862e-03	**5.702e-41**±1.541e-40	6.188e-04±3.862e-03
f_4	**2.147e-31**±3.879e-32	2.071e-02±9.422e-02	**3.004e-34**±1.018e-34	2.071e-02±9.422e-02
f_5	1.142e+01±2.774e+01	**8.124e+00**±1.931e+00	**3.452e+00**±8.028e+00	8.124e+00±1.931e+00
f_6	**0.000e+00**±0.000e+00	**0.000e+00**±0.000e+00	**0.000e+00**±0.000e+00	**0.000e+00**±0.000e+00
f_7	3.022e-04±2.883e-04	**1.862e-04**±8.036e-05	1.790e-04±1.414e-04	**1.040e-04**±4.462e-05
f_8	**1.450e+01**±3.550e+00	1.615e+01±4.077e+00	**1.121e+00**±3.185e+00	9.899e+00±6.059e+00
f_9	**3.713e-15**±9.736e-16	2.020e-03±6.074e-03	**3.357e-15**±1.379e-15	2.020e-03±6.074e-03
f_{10}	1.945e-01±1.187e-01	**1.671e-03**±3.655e-03	1.515e-01±1.100e-01	**1.671e-03**±3.655e-03
f_{11}	**1.096e-20**±9.120e-36	2.202e-04±1.281e-03	**1.096e-20**±9.120e-36	2.202e-04±1.281e-03
f_{12}	**3.485e-21**±1.520e-36	4.673e-03±2.250e-02	**3.485e-21**±1.520e-36	4.673e-03±2.250e-02

TABLE IV
NUMBER OF TRIAL POINTS OF NEDA OVER 50 RUNS FOR ALL THE TEST INSTANCES WITH $M = 1, 5, 10, 15, 20, 25, 30$ AFTER 3000 GENERATIONS.

Instance	$M = 1$	$M = 5$	$M = 10$	$M = 15$	$M = 20$	$M = 25$	$M = 30$
f_1	2.383e-59	2.955e-106	3.380e-89	2.623e-111	**1.108e-111**	1.935e-30	2.331e-19
f_2	1.752e-43	**4.316e-65**	8.816e-51	7.547e-16	1.054e-13	6.868e-12	1.924e-11
f_3	1.622e-04	6.595e-20	**5.702e-41**	3.777e-29	7.153e-14	2.082e-12	9.705e-12
f_4	1.082e-07	5.047e-34	**3.004e-34**	2.364e-23	4.830e-14	1.687e-12	8.242e-12
f_5	6.485e+00	6.513e+00	**3.452e+00**	1.114e+01	7.276e+00	1.194e+01	1.374e+01
f_6	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**
f_7	1.106e-03	2.578e-04	1.790e-04	1.261e-04	1.346e-04	**1.221e-04**	1.488e-04
f_8	6.872e+00	2.993e+00	**1.121e+00**	9.970e+00	9.811e+00	1.059e+01	1.485e+01
f_9	3.926e-15	3.428e-15	3.357e-15	2.310e-02	2.310e-02	**2.647e-15**	2.310e-02
f_{10}	1.820e-01	**6.286e-02**	1.515e-01	9.097e-02	1.157e-01	1.130e-01	1.254e-01
f_{11}	**1.096e-20**	**1.096e-20**	**1.096e-20**	**1.096e-20**	**1.096e-20**	6.022e-16	6.220e-03
f_{12}	**3.485e-21**	**3.485e-21**	**3.485e-21**	**3.485e-21**	3.486e-21	1.526e-15	5.990e-14

TABLE III
MEAN VALUES OF nEDA OVER 50 RUNS FOR ALL THE TEST INSTANCES WITH $n = 10, 20, 30$ AFTER 3000 GENERATIONS.

Instance	$n = 10$	$n = 20$	$n = 30$
f_1	3.380e-89	8.872e-30	4.339e-01
f_2	8.816e-51	2.979e-16	3.978e-03
f_3	5.702e-41	2.013e+03	2.261e+04
f_4	3.004e-34	6.330e-01	3.659e+01
f_5	3.452e+00	4.233e+01	3.088e+04
f_6	0.000e+00	0.000e+00	3.140e+00
f_7	1.790e-04	1.134e-02	2.220e-01
f_8	1.121e+00	9.614e+01	3.129e+02
f_9	3.357e-15	7.052e-15	2.834e-01
f_{10}	1.515e-01	1.062e-01	8.711e-01
f_{11}	1.096e-20	5.482e-21	1.215e+04
f_{12}	3.485e-21	3.488e-21	8.607e+04

to discuss how many trial points need to be sampled. In this section, nEDA is applied to the 12 instances with number of trial points $M = 1, 5, 10, 15, 20, 25$, and 30. The mean values of nEDA over 50 runs for the test instances after 3000 generations are shown in Table IV.

We can see that when $5 \leq M \leq 25$, nEDA performs better than nEDA with small or big M values. This could be explained as follows. If M is too small, the density of trial points might not be precise enough to pick up offspring. While if M is too big, the chosen offspring is too close to its parent which might make nEDA hard to find a better solution.

V. CONCLUSIONS

In this paper, we used nonparametric density estimation methods to guide the search of EDAs. Based on this idea, a nonparametric EDA (nEDA) was thus proposed. Unlike a generic EDA which first builds a probabilistic model and then samples offspring solutions from the model thus built, nEDA first randomly generates a set of candidates around the parents and then select some as offspring solutions by using their estimated density values. The proposed method was compared with a generic EDA based on multivariate Gaussian model on 12 widely used test instances. The statistical results shown that nEDA performed better than the generic EDA on most of the test instances.

It should be noted that nEDA is just a conceptual algorithm to illustrate our idea on the reproduction procedure of EDAs. Nonparametric EDA does not perform perfectly in this paper. It is worth to improve its performance and some future work may include

- trying some other nonparametric density estimation methods;
- investigating how to combine both the objective values and density values to guide the search;
- balancing algorithm convergence and population diversity;
- improving its scalability and stability;
- comparing with other algorithms on more test instances like the CEC 2005 test problem [21].

ACKNOWLEDGMENT

This work is supported by National Basic Research Program of China (No.2011CB707104), National Science Foundation of China (No.61005050), Program for New Century Excellent Talents in University (No.NCET-08-0193), and East China Normal University Innovation Program (No.78210059).

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.
[2] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco,US: Morgan Kaufmann, 2001.
[3] K. V. Price, *New Ideas in Optimization*. McGraw-Hill, 1999, ch. An Introduction to Differential Evolution, pp. 79–108.
[4] T. Fukuda, K. Mori, and M. Tsukiyama, "Immune networks using genetic algorithm for adaptive produciton scheduling," in *15th IFAC World Congress*, vol. 3, 1993, pp. 57–60.
[5] M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT Press, 2004.
[6] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning*. Springer Verlag, 2004.
[7] K.-H. Han and J.-H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *IEEE Congress on Evolutionary Computation (CEC 2000)*, 2000, pp. 1354–1360.
[8] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I: Binary parameters," in *Parallel Problem Solving from Nature (PPSN IV)*, ser. LNCS, vol. 1411, 1996, pp. 178–187.
[9] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.
[10] J. Sun, Q. Zhang, and E. Tsang, "DE/EDA: A new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3, pp. 249–262, 2005.
[11] H. Mühlenbein and T. Mahnig, "Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning," *Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 157–192, 2002.
[12] Q. Zhang, "On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 80–93, 2004.
[13] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 127–136, 2004.
[14] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds., *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Secaucus, NJ, USA: Springer-Verlag, 2006.
[15] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.
[16] Q. Zhang, J. Sun, and E. Tsang, "Combinations of estimation of distribution algorithms and other techniques," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 273–280, 2007.

[17] M. Costa and E. Minisci, “MOPED: A multi-objective parzen-based estimation of distribution algorithm for continuous problems,” in *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization*, 2003, pp. 282–294.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification, Second Edition*. John Wiley & Sons, Inc., 2001.

[19] P. Larrañaga, J. Lozano, and E. Bengoetxea, “Estimation of distribution algorithms based on multivariate normal and gaussian networks,” the Department of Computer Science and Artificial Intelligence, University of the Basque Country, Tech. Rep. KZZA-IK-1-01, 2001.

[20] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[21] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, “Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization,” Nanyang Technological University and Indian Institute of Technology, Tech. Rep. 2005005, 2005.