Fast and Efficient Evolutionary Algorithms Based on Bayesian Networks

Youlong Yang, Yanying Li, Wensheng Wang and Wenming Yang

***Abstract*—Evolutionary algorithms are powerful search techniques which have been used successfully in many different domains. Parallel evolutionary algorithm has become a research focus due to its easy implement and promise substantial gains in performance. In this paper a framework of tree-model-based parallel evolutionary algorithm (T-PEA) is proposed. The presented method employs Bayesian Dirichlet metric to construct a tree model from a set of potential solutions, which is then used to model potential solutions and guide exploration in the search space. The correctness and rationality of the proposed method for learning tree models are analyzed and proved in the context of genetic and evolutionary. The method is important not only for T-PEA, but also for machine learning and data mining. The experimental results show that the proposed algorithm can efficiently and rapidly converge and obtain the optimal solution from all test functions.**

***Index Terms*—evolutionary algorithms, parallel algorithms, graphical models, tree models, Bayesian Networks**

I. Introduction

Motivated by the analogy of evolution and population genetics, evolutionary algorithm (EA) is proposed to solve optimization problems by random searching. Its robustness and efficiency have been demonstrated in searching higher dimensional varied spaces in a wide field of applications, including classification, forecasting, machine learning, ecological, social systems and so on [1], [2], [3], [4], [5], [6]. Parallel evolutionary algorithms (PEAs) can be divided into two types based on two basic ways, namely the standard parallel approach and the decomposition approach [7], [8], [9]. The first way is that a sequential EA model is implemented on a parallel computer. To do this, we divide the mission of evaluating the population among a lot of processors. A single master processor is selected to control the total population and do the selection. Some slave-processors are used to receive individuals which are recombined to create offsprings. Before returned to the master, these offsprings should have been evaluated. All the processes are done synchronously, where the master proceeds to next generation after receiving the fitness values for all the population. The main characteristic of the decomposition approach is that the full population exists in a certain distributed form. This method consists mostly of Coarse-Grained PEA (migration model or island model [10], [11]) and Fine-Grained PEA (diffusion model or neighborhood model [12]). In recent years, some combinations of the previous methods are presented with adding complexity in some situations, such as the hybrid parallel algorithms [13], [14], [15], [16].

The goal in optimization is to find the best solution of an optimization problem, with respect to one or more criteria. In order to use an EA to solve optimization problems, a suitable model should be firstly designed for representing those solutions. Bayesian Network (BN) can do this work well. BNs are graphical models which combine probability theory and graph theory. So far a BN is one of the most effective tools to deal with uncertainty problems. A BN consists of a directed acyclic graph (DAG) and parameters. The DAG qualitatively represents the conditional independent relationships among variables, and the parameters quantitatively show the conditional independent relationships among variables. The Bayesian optimization algorithm (BOA) is a typical scheme that a BN is successfully embedded in solving optimization problems. BOA uses BNs to model promising solutions and subsequently guide the exploration of the search space [17], [18], [19]. In the first step, BOA generates the initial population of strings at random with a uniform distribution, but the initial population can also be biased to a particular region in the search space. From the current population, the better strings are firstly selected by using one of the popular selection methods [16], [20]. In the second step, a BN that fits the selected set of strings is constructed. In the third step, new strings are generated according to the joint distribution encoded by the constructed network. Finally, the new strings are incorporated into the original population, replacing some of the old ones or all of them. The above four steps are repeated until some termination criteria are met.

However, constructing a BN that fits a given dataset is an NP-hard problem, and it also needs consuming mass computational resources [21]. Besides, the methods used to construct a BN have some flaws, and a rigorously theoretical analysis on these methods is needed. In this paper a parallel model is introduced that combines BOA with standard PEA [22], [23], named T-PEA. Instead of constructing a general Bayesian network, T-PEA constructs a tree model based on Bayesian Dirichlet (BD) metric [24], [25]. We can easily built a tree model compared with a general BN. This improvement saves mass computational resources used in BOA. A theoretical analysis is made to develop an effective method for constructing a tree model. Two examples are constructed to show how a tree model is built from a given data. The experimental results show that T-PEA is able to solve all tested problems with a proximate linear time. In comparison with a simple parallel evolutionary algorithm (PEA), T-PEA has the advantages of small iterations, fast and global convergence. The difference of iterations until successful

Manuscript received April 24, 2015; revised June 23 , 2015. This work was supported by the National Nature Science Foundation of China (No. 61403290, 11301408, 11401454), the Foundation for Youths of Shaanxi Province (No.2014JQ1020) and the Foundation of Baoji University of Arts and Sciences(No.ZK15081).

Youlong Yang, Wensheng Wang and Wenming Yang are with the School of Mathematics and Statistics, Xidian University, Xi'an, Shaanxi, 710126, P.R. China. e-mail:ylyang@mail.xidian.edu.cn, 291980664@qq.com, yonthweiming@163.com

Yanying Li is with the School of Mathematics and Statistics, Xidian University, Xi'an, Shaanxi, 710126 P.R. China and the College of Mathematics and Information Science, Baoji University of Arts and Sciences, Baoji, Shannxi, 721013, P.R.China. e-mail: liyanying81@163.com.

convergence between the two algorithms would significantly enlarge with the function size grows. The optimum returned by T-PEA is more precise than that returned by PEA on all tested problems. Furthermore, T-PEA succeeds BOA's advantages. T-PEA is independent of the ordering of the variables in a string, therefore changing this would not affect the performance of the algorithm.

The rest of the paper is organized as follows. The pseudo-code of the presented algorithm is described in Section 2. We introduce BD metric and make a theoretical analysis on tree models in Section 3. In this theoretical part of the paper, only binary variables that can obtain either 0 or 1 are considered. In Section 4, we discuss the algorithm and construct a tree model based BD metric. The results of the experiments are presented in Section 5. Finally, conclusions and future research direction are given in Section 6.

II. Outline of the Algorithm

The algorithm is summarized as follows.

Step 1. set $t \leftarrow 0$, randomly generate initial population $P_1(0), P_2(0), \cdots, P_N(0)$.

Step 2. parallel evolution of each population.

2.1 select a set of promising strings $S_i(t)$ from population $P_i(t)$ $(1 \leq i \leq N)$ and exchanged individuals with other populations.

2.2 construct tree model TB that fits the selected set of strings $S_i(t)$ using a chosen metric system.

2.3 generate a set of new strings $O_i(t)$ according to the joint distribution encoded by tree model TB.

2.4 create a new population $P_i(t+1)$ by replacing some strings from $P_i(t)$ with $O_i(t)$, Set $t \leftarrow t+1$.

Step 3. If termination criteria are not met, go to **step 2**.

We also draw up the flow chart of T-PEA (see Fig. 1). It dose a descending sort by using the fitness size of the individuals of population $P_i(t)$ $(1 \leq i \leq N)$, then the individuals of $P_i(t)$ are respectively assigned to the three groups $R_{i1}(t)$, $R_{i2}(t)$ and $R_{i3}(t)$. And the three groups contain of 10, 10 and 20 individuals, respectively.

III. Bayesian Networks

A. Bayesian metrics

Bayesian metrics account for the uncertainty of the network structure and its parameters by using the Bayes rule and assigning prior distribution to both the network structure and the parameters of each structure [26], [27], [28]. The quality of a particular structure is measured by computing the marginal likelihood of the structure with respect to the given data. The marginal likelihood is computed by averaging the likelihood of the models conditioned on the observed data according to a prior distribution over all possible conditional probabilities in the model:

$$p(B\,|D\,) = \frac{p(B)p(D|B)}{p(D)} = \frac{p(B)\int_{\theta} p(\theta\,|B\,)p(D\,|B,\theta\,)d\theta}{p(D)} \tag{1}$$

where B is the evaluated Bayesian network structure (without particular parameters), D is the dataset, and θ denotes a set of possible parameters specifying the conditional probabilities in the network B. Furthermore, $p(B)$ is the prior distribution of the network structures, $p(\theta\,|B\,)$ is the conditional probabilities of the parameters given the particular network structure, and $p(D\,|B,\theta\,)$ denotes the probability of D given the network structure and its parameters. Here, $p(D)$ is computed as

$$p(D) = \int_{\Theta} p(D|\theta)p(\theta)d\theta \tag{2}$$

where $p(\theta)$ be the prior probability distribution for the Bayesian network, Θ is the space of all possible parameters. Since $p(D)$ is a normalizing constant, this term is usually omitted. The computation of the marginal likelihood is intractable in general. The Bayesian-Dirichlet (BD) metric assumes that the conditional probabilities follow the Dirichlet distribution and makes a number of additional assumptions, which yields the following score:

$$\begin{aligned}\mathrm{BD}(B) = p(B)\prod_{i=1}^{n}\prod_{\pi_{X_i}}\frac{\Gamma(m'(\pi_{X_i}))}{\Gamma(m'(\pi_{X_i})+m(\pi_{X_i}))} \\ \cdot\prod_{x_i}\frac{\Gamma(m'(x_i,\pi_{X_i})+m(x_i,\pi_{X_i}))}{\Gamma(m'(x_i,\pi_{X_i}))}\end{aligned} \tag{3}$$

where the dataset D corresponds to a set of promising strings $S(t)$ from population $P(t)$, B is the network matching with $S(t)$, X_i denotes the ith vertex of the network, the product over x_i runs over all instances of X_i (in binary case these are 0 or 1), the product over π_{X_i} runs over all instances of the parents $\prod_i$ of X_i (all possible combinations of values of $\prod_i$), $m(x_i,\pi_{X_i})$ is the number of instances with $X_i = x_i$ and $\prod_i = \pi_{X_i}$, and $m(\pi_{X_i})$ is the number of instances with the parents $\prod_i$ set to the particular values given by π_{X_i}, which is computed as $m(\pi_{X_i}) = \sum_{x_i} m(x_i,\pi_{X_i})$. Terms $m'(x_i,\pi_{X_i})$ and $m'(\pi_{X_i})$ denote prior information about the values of the corresponding parameters $m(x_i,\pi_{X_i})$ and $m(\pi_{X_i})$, respectively. The cost of computing the metrics of Bayesian networks using Eq.(3) increases considerably with the number of vertices of networks increasing. However, the search procedure is used to explore the space of all possible networks in order to find the one (or a set of networks) with the metric values as high as possible, but we only need to know whether the metric value is increasing with links among vertexes adding. Therefore, it is possible to reduce computational complexity by exploring the Bayesian metric of network based on greedy algorithm. Since $\Gamma(p+1) = p!$, where p is an integer or zero, we have the following score from

$$\begin{aligned}\mathrm{BD}(B) = p(B)\prod_{i=1}^{n}\prod_{\pi_{X_i}}\frac{(m'(\pi_{X_i})-1)!}{(m'(\pi_{X_i})+m(\pi_{X_i})-1)!} \\ \cdot\prod_{x_i}\frac{(m'(x_i,\pi_{X_i})+m(x_i,\pi_{X_i})-1)!}{(m'(x_i,\pi_{X_i})-1)!}\end{aligned} \tag{4}$$

Since prior information over the evaluated populations is well-known, we consider the K2 metric which uses an uninformative assignments and assign (in binary case). $m'(\pi_{X_i}) = \sum_{x_i} m'(x_i,\pi_{X_i}) = 2$ and $m'(x_i,\pi_{X_i}) = 1$. Now we get the following K2 metric [29]:

$$p(B\,|D\,) = \prod_{i=0}^{n-1}\prod_{\pi_{X_i}}\frac{1}{(m(\pi_{X_i})+1)!}\prod_{x_i} m(x_i,\pi_{X_i})! \tag{5}$$

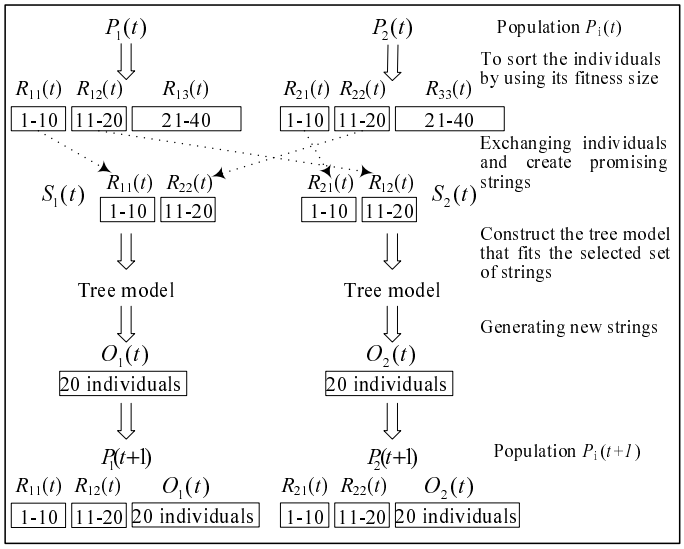

Fig. 1. The structure of parallel evolutionary algorithms based on tree model. The populations $P_i(t+1)$ are evolved from the populations $P_i(t)$ by using tree models and exchanged individuals.

B. Theoretical Analysis

In this paper we only considered BD metric. Let $x_{a=x_i}$ be the number of chromosome with the *a*th locus of chromosome in promising strings being x_i; and let $x_{b=x_i}$ be the number of chromosome with the *b*th locus of chromosome in promising strings being x_i. In binary case, x_i is either 0 or 1. Term $x_{b=x_{ib}}^{a=x_{ia}}$ denotes the number corresponding to the *a*th locus and the *b*th locus of chromosome in promising strings being x_{ia} and x_{ib}.

Theorem 1: For two independent vertexes a and b, and a binary dataset D, we consider the K2 metric, then

1. The inequation $p(B_{a\to b}\,|D) > p(B_{a\leftarrow b}\,|D)$ is equivalent to the following propositions. There is a): $x_{a=0} > x_{b=0}$ and $x_{a=0} > x_{b=1}$, or $x_{a=1} > x_{b=0}$ and $x_{a=1} > x_{b=1}$. And b): $|x_{a=0} - x_{a=1}| > |x_{b=0} - x_{b=1}|$.

2. $p(B_{a\to b}\,|D) = p(B_{a\leftarrow b}\,|D)$ if and only if there is either $x_{a=0} = x_{b=0}$ or $x_{a=0} = x_{b=1}$.

Proof: Since

$$p(B_{a\to b}\,|D) = \frac{x_{a=0}!x_{a=1}!}{(1+n)!}\cdot\frac{x_{b=0}^{a=0}!x_{b=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{b=0}^{a=1}!x_{b=1}^{a=1}!}{(1+x_{a=1})!}$$

and

$$p(B_{a\leftarrow b}\,|D) = \frac{x_{b=0}!x_{b=1}!}{(1+n)!}\cdot\frac{x_{b=0}^{a=0}!x_{b=0}^{a=1}!}{(1+x_{b=0})!}\cdot\frac{x_{b=1}^{a=0}!x_{b=1}^{a=1}!}{(1+x_{b=1})!}$$

the inequation $p(B_{a\to b}\,|D) > p(B_{a\leftarrow b}\,|D)$ gives

$$\frac{p(B_{a\to b}\,|D)}{p(B_{a\leftarrow b}\,|D)} = \frac{(1+x_{b=0})(1+x_{b=1})}{(1+x_{a=0})(1+x_{a=1})} > 1$$

That is

$$(1+x_{b=0})(1+x_{b=1}) > (1+x_{a=0})(1+x_{a=1})$$

Therefore

$$x_{b=0}x_{b=1}+x_{b=0}+x_{b=1}+1 > x_{a=0}x_{a=1}+x_{a=0}+x_{a=1}+1$$

and this inequation is equivalent to $x_{b=0}x_{b=1} > x_{a=0}x_{a=1}$. It is easy to prove the following propositions being equivalent to $x_{b=0}x_{b=1} > x_{a=0}x_{a=1}$.

So We have: a) there is $x_{a=0} > x_{b=0}$ and $x_{a=0} > x_{b=1}$, or $x_{a=1} > x_{b=0}$ and $x_{a=1} > x_{b=1}$.

b) $|x_{a=0} - x_{a=1}| > |x_{b=0} - x_{b=1}|$.

2. From the above proof, It is easy to show $p(B_{a\to b}\,|D) = p(B_{a\leftarrow b}\,|D)$ if and only if $x_{b=0}x_{b=1} = x_{a=0}x_{a=1}$. ■

Theorem 2: If $p(B_{a\to b}) \geq p(B_{\text{empty}})$ and $p(B_{a\to c}) \geq p(B_{\text{empty}})$, then

$$p(B_{a\to b,c}) \geq \max\{p(B_{\text{empty}}), p(B_{a\to b}), p(B_{a\to c})\}$$

Proof: Since

$$p(B_{a\to b}) = BD(B_{a\to b}\,|S(t)) = \frac{x_{a=0}!x_{a=1}!}{(1+n)!} \cdot\Big[\frac{x_{b=0}^{a=0}!x_{b=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{b=0}^{a=1}!x_{b=1}^{a=1}!}{(1+x_{a=1})!}\Big]\cdot\frac{x_{c=0}!x_{c=1}!}{(1+n)!}$$

$$p(B_{\text{empty}}) = BD(B_{\text{empty}}\,|S(t)) = \frac{x_{a=0}!x_{a=1}!}{(1+n)!}\cdot\frac{x_{b=0}!x_{b=1}!}{(1+n)!}\cdot\frac{x_{c=0}!x_{c=1}!}{(1+n)!}$$

we have

$$\frac{x_{a=0}!x_{a=1}!}{(1+n)!}\cdot\Big[\frac{x_{b=0}^{a=0}!x_{b=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{b=0}^{a=1}!x_{b=1}^{a=1}!}{(1+x_{a=1})!}\Big]\cdot\frac{x_{c=0}!x_{c=1}!}{(1+n)!} \geq \frac{x_{a=0}!x_{a=1}!}{(1+n)!}\cdot\frac{x_{b=0}!x_{b=1}!}{(1+n)!}\cdot\frac{x_{c=0}!x_{c=1}!}{(1+n)!}$$

Therefore,

$$\frac{x_{b=0}^{a=0}!x_{b=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{b=0}^{a=1}!x_{b=1}^{a=1}!}{(1+x_{a=1})!} \geq \frac{x_{b=0}!x_{b=1}!}{(1+n)!}$$

Thus

$$p(B_{a\to b,c}) = \frac{x_{a=0}!x_{a=1}!}{(1+n)!}\Big[\frac{x_{b=0}^{a=0}!x_{b=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{b=0}^{a=1}!x_{b=1}^{a=1}!}{(1+x_{a=1})!}\Big] \cdot\Big[\frac{x_{c=0}^{a=0}!x_{c=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{c=0}^{a=1}!x_{c=1}^{a=1}!}{(1+x_{a=1})!}\Big]$$

$$\geq \frac{x_{a=0}!x_{a=1}!}{(1+n)!}\cdot\frac{x_{b=0}!x_{b=1}!}{(1+n)!}\cdot\Big[\frac{x_{c=0}^{a=0}!x_{c=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{c=0}^{a=1}!x_{c=1}^{a=1}!}{(1+x_{a=1})!}\Big] = P(B_{a\to c})$$

It is similar to obtain $p(B_{a\to b,c}) \geq p(B_{a\to b})$ by proving

$$p(B_{a\to b,c}) \geq p(B_{a\to c})$$

■

Corollary 1: Let B be a network which consists of n vertexes (n variables) $X_1, X_2, \cdots, X_n$, B_{empty} be the network with no links among these vertexes, and $B_{X_i\to X_j}$ be the network with the link only from vertex X_i to X_j.

If $p(B_{X_1\to X_2}) \geq p(B_{\text{empty}})$, $p(B_{X_1\to X_3}) \geq p(B_{\text{empty}}), \cdots,$ $p(B_{X_1\to X_k}) \geq p(B_{\text{empty}})$, then

$$p(B_{X_1\to X_2,X_3,\cdots,X_k}) \geq \max\{p(B_{X_1\to X_2}), p(B_{X_1\to X_3}), \cdots, p(B_{X_1\to X_k}), p(B_{\text{empty}})\}$$

The proof is similar to the above Theorem 2.

However, if $p(B_{a\to c}) \geq p(B_{\text{empty}})$ and $p(B_{b\to c}) \geq p(B_{\text{empty}})$, we do not have $p(B_{a,b\to c}) \geq p(B_{\text{empty}})$. An example is given below to show this.

Exampl 1: Let $n = 16$, $x_{c=0} = 4$, $x_{c=1} = 12$,

$$x_{c=0}^{a=0} = x_{c=0}^{b=0} = x_{c=1}^{a=0} = x_{c=1}^{b=0} = x_{c=0}^{a=1} = x_{c=0}^{b=1} = 2$$

$x_{c=1}^{a=1} = x_{c=1}^{b=1} = 10$, $x_{000} = x_{110} = 2$, $x_{001} = x_{011} = x_{101} = 1$, $x_{010} = x_{100} = 0$, and $x_{111} = 9$, where x_{101} denotes the number of chromosome corresponding to the ath locus, the *b*th locus and the *c*th locus of chromosome in promising strings being 1, 0 and 1, respectively (the other symbols have similar meanings). Then we get

$$P(B_{\text{empty}}) = \lambda\frac{x_{c=0}!x_{c=1}!}{(1+n)!} = \lambda\frac{4!12!}{17!} = \frac{\lambda}{30940}$$

$$P(B_{\text{a}\to\text{c}}) = P(B_{b\to\text{c}}) = \lambda\frac{x_{c=0}^{a=0}!x_{c=1}^{a=0}!}{(1+x_{a=0})!}\cdot\frac{x_{c=0}^{a=1}!x_{c=1}^{a=1}!}{(1+x_{a=1})!} = \lambda\frac{2!2!}{5!}\cdot\frac{2!10!}{13!} = \frac{\lambda}{25740} > P(B_{\text{empty}})$$

and

$$P(B_{a,b\to\text{c}}) = \lambda\frac{x_{000}!x_{001}!}{(1+x_{b=0}^{a=0})!}\cdot\frac{x_{010}!x_{011}!}{(1+x_{b=1}^{a=0})!}\cdot\frac{x_{100}!x_{101}!}{(1+x_{b=0}^{a=1})!}\cdot\frac{x_{110}!x_{111}!}{(1+x_{b=1}^{a=1})!} = \frac{\lambda}{31680} < P(B_{\text{empty}})$$

Hence, we cannot get $P(B_{a,b\to\text{c}}) > P(B_{\text{empty}})$ by $P(B_{\text{a}\to\text{c}}) > P(B_{\text{empty}})$ and $P(B_{b\to\text{c}}) > P(B_{\text{empty}})$.

IV. Constructing Tree Models

In this part, we introduce the mechanism of constructing tree models in detail by combining two examples. Let $X_1, X_2, \cdots, X_n$ be n discrete variables, each variable correspond to one position of a gene on a chromosome which be transformed into binary coding. And let $i_1, i_2, \cdots, i_r, \cdots, i_n$ are some unknown permutations of $1, 2, 3, \cdots, n$, ϕ maps integer r to integer $\phi(r)$, where$1 \leq r \leq n$ and $0 \leq \phi(r) < r$. For all $1 \leq i \leq n$, $P(X_i\,|X_0\,)$ is by definition equal to $P(X_i)$, i.e., $P(X_i\,|X_0\,) = P(X_i)$. Therefore, we get the joint probability distribution over tree model:

$$P(X_1X_2\cdots X_n) = \prod_{r=1}^{n} P(X_{i_r}\,|X_{i_{\phi(r)}}\,) \tag{6}$$

This means we confine our models to the networks in which each node can have at most one parent. If there is $P(X_{r_i}\,|X_0\,)$ in Eq.(6) for some i_r, then X_{r_i} has no parent and X_0 is the root of the tree model. To find the optimal tree model based on K2 metric, we give some promises and definitions:

$$BD(X_i \to X_j) \triangleq \frac{x_{i=0}^{j=0}!x_{i=0}^{j=1}!}{(1+x_{i=0})!}\cdot\frac{x_{i=1}^{j=0}!x_{i=1}^{j=1}!}{(1+x_{i=1})!} \tag{7}$$

$$BD(X_0 \to X_j) = BD(X_j) \triangleq \frac{x_{j=0}!x_{j=1}!}{(1+n)!} \tag{8}$$

We then create

$$BD(TB_{X_1X_2\cdots X_n}) = \prod_{r=1}^{n} BD(X_{i_{\phi(r)}} \to X_{i_r})$$

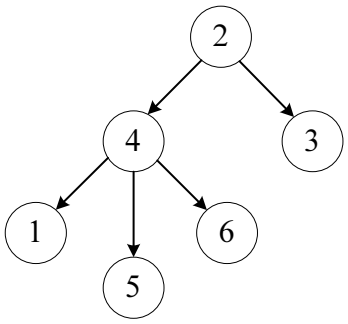

Fig. 2. A dependency tree graph with 6 nodes.

where $BD(TB_{X_1X_2\cdots X_n})$ denotes BD metric value of tree model $TB_{X_1X_2\cdots X_n}$, which consists of n vertexes $X_1, X_2, \cdots, X_n$. Our goal is to find out the tree model which has the maximum metric value in a given dataset.

Example 2: Consider a dependency tree graph $TB_{X_1X_2X_3X_4X_5X_6}$ shown in Fig. 2. Map ϕ is defined as: $\phi(2) = 0$,$\phi(4) = 2$,$\phi(3) = 2$, $\phi(1) = 4$, $\phi(5) = 4$, $\phi(6) = 4$, and its BD metric value is

$$\begin{aligned} &BD(TB_{X_1X_2X_3X_4X_5X_6}) \\ &= \prod_{r=1}^{n} BD(X_{i_{\phi(r)}} \to X_{i_r}) \\ &= BD(X_2)\cdot BD(X_2 \to X_3)\cdot BD(X_2 \to X_4) \\ &\cdot BD(X_4 \to X_1)\cdot BD(X_4 \to X_5)\cdot BD(X_4 \to X_6) \end{aligned}$$

Let I be a set which includes all variables, namely $I = \{X_1, X_2, \cdots, X_n\}$. The basic algorithm to construct the tree model with the maximum BD metric value can be described as follows.

Step 1. Select initial variable X_{root} as root node of the tree.

In our implementation we select the variable X_{root} such that

$$\begin{aligned} &|x_{X_{root}=0} - x_{X_{root}=1}| \\ &\geq \max\{\,|x_{X_i=0} - x_{X_i=1}||\; i = 1, 2, \cdots, n\} \end{aligned}$$

Step 2. Let I_{in} denote any variable already in the tree and $I_{out} = I - I_{in}$ be any variable not yet in the tree. For $X_i \in I_{in}$ and $X_j \in I_{out}$, we have maximum BD metric value $BD(X_{in-i} \to X_{out-j})$, that is

$$\begin{aligned} &BD(X_{in-i} \to X_{out-j}) \\ &= \max\{BD(X_i \to X_j)\,|X_i \in I_{in}, X_j \in I_{out}\} \end{aligned}$$

Step 3. Add X_{out-j} to the local tree I_{in}, with X_{in-i} as its parent, such that the local tree gains a leaf, i.e. $I_{in} \leftarrow I_{in} \cup \{X_{out-j}\}$, $I_{out} \leftarrow I_{out} - \{X_{out-j}\}$

Step 4. Repeat **Steps 2** and **3** until all the variables have been added to tree such that$I = I_{in}$.

Example 3: Consider constructing a tree model which has 8 variables, and dataset D is a set of promising strings $S(t)$ from $P(t)$. The dataset is shown in Fig. 3 and consists of 20 chromosomes. Firstly, we can reduce the number of all direct links among the 8 variables from $A_8^2 = \frac{8!}{(8-2)!} = 56$ to $C_8^2 = \frac{8!}{(8-2)!2!} = 28$ based on Theorem 1. Secondly, by the above algorithm, we construct the tree model having the maximum BD metric value. The initial root variable X_{root}=e is selected as root node of the tree. The first leaf is added to the local tree (see Fig. 4(A)), then the second leaf (see Fig. 4(B)) and so on until the last leaf to have the tree model which matches dataset $S(t)$.

Chromo -some	20 Chromosomes 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	$x_{i=0}$	$x_{i=1}$
ⓐ	1	0	1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	1	1	6	14
ⓑ	1	1	0	1	1	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	8	12
ⓒ	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	15	5
ⓓ	1	0	1	1	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	1	9	11
ⓔ	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	18	2
ⓗ	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	14	6
ⓕ	1	1	0	0	1	0	1	0	1	1	0	1	0	0	1	0	1	0	0	1	10	10
ⓖ	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	16	4

Fig. 3. A data set $S(t)$ from $P(t)$. $S(t)$ consists of 20 chromosomes.

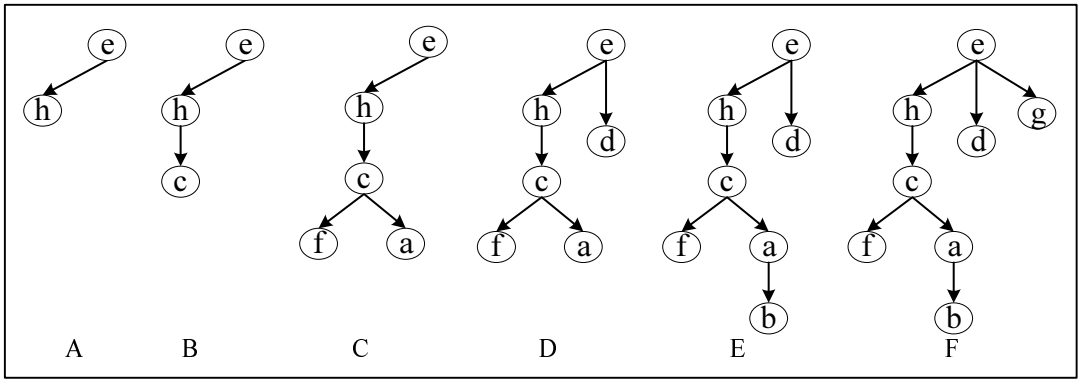

Fig. 4. Tree model generated by $S(t)$ from $P(t)$.

V. Experimental Evaluation

The experiments are designed in order to show the performance of the proposed algorithm on decomposable problems with uniformly scaled deceptive building blocks [30], [31]. To evaluate the advantage of T-PEA, we select a simple PEA based on genetic algorithms as the the contrast algorithm. In the following sections, test problems are described and the results are presented and discussed.

A. Test Function

To test T-PEA, four test problems for fixed-length binary strings were considered. The first two problems, Dec-3 function and De Jong's function 1, are uniformly scaled decomposable problems of bounded difficulty for which it is necessary that algorithms find an adequate problem decomposition; if an adequate decomposition is not found, the algorithm is expected to scale up exponentially. If T-PEA finds an adequate function decomposition to solve Dec-3 and De Jong's function 1, it indicates that PEA should be capable of solving other decomposable functions of bounded difficulty and anything easier. The last two test problems are typical multivariate multimodal function named Shubert and Fun.4. A unitation function is a function whose value depends only on the number of ones in a binary input string. The function values for the strings with the same number of ones are equal. Several functions of unitation can be additively composed in order to form a more complex function. Let us have a function of unitation defined for strings of length k. Then, the function additively composed of functions f_k is defined as

$$F(X) = \sum_{i=1}^{l} f_k(u_i) \tag{9}$$

where X is the set of n variables and for $i \in \{1, \cdots, l\}$ are subsets of k variables from X. Sets u_i can be either overlapping or non-overlapping. They can be mapped onto a string so that the variables from one set are either mapped close to each other or spread throughout the whole string. It is obviously that $F(X)$ is additively decomposable. Several simple and easily 3-bit deceptive functions are unitation functions and can be additively composed and then form a more complex Dec-3 function. For Dec-3 function, the input string is first partitioned into independent groups of 3 bits each. This partitioning does not change during the run. A 3-bit deceptive function is applied to each group of 3 bits and the contributions of all deceptive functions are added together to form the fitness of Dec-3 function. And a 3-bit deceptive function is defined as

$$f_{3deceptive}(u) = \begin{cases} 0.9 & \text{if } u = 0 \\ 0.8 & \text{if } u = 1 \\ 0 & \text{if } u = 2 \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

where u is the number of ones in an input string of 3 bits. Therefore, an n-bit Dec-3 has one global optimum in the string of all ones and $2^{\frac{n}{3}} - 1$ other local optima. In order to avoid this situation that the optimization is misled away from the optimum when each bit is considered independently, it is necessary to consider interactions among the positions in each partition [16].

De Jong's function 1 [33] is defined to be a performance index to be minimized, and is a simple n-dimensional parabola with spherical constant-cost contours. It is a continuous, convex, unimodal, low-dimensional quadratic function with a minimum of zero at the origin. For testing the performance, it is restricted to a bounded subspace of R^n of

the form $-512 \leq x_i \leq 512$, $i = 1, \cdots, n$, and is discretized by specifying a resolution factor Δx_i=1 for each axis. Therefore, the search space of the evolutionary algorithms can be used a binary representation and be searched consisted of $(1024)^n$ alternative solutions. For De Jong's function, the input string is first partitioned into independent groups of 10 bits each. This partitioning does not change during the run. De Jongs function 1 is given by:

$$F(x) = \sum_{i=1}^{n} (x_i^2) \tag{11}$$

where $x_i \in [-512, 512]$, $i = 1, \cdots, n$.

Shubert function is defined as

$$F(X) = \prod_{l=1}^{n} \sum_{i=1}^{5} i \cos[(i+1) \times x_l + i] \tag{12}$$

where $x_i \in [-10, 10]$, $i = 1, \cdots, n$.

We specify the minimum of Shubert function. When $n = 2$, $F(X)$ is the standard Shubert function. Its known that the standard Shubert function has 760 local optimal solutions. Among which there are 18 optimums and the value is -186.7310. For Shubert function, the input string is first partitioned into independent groups of 10 bits each. This partitioning does not change during the run.

The Fun. 4 is defined as

$$F(x, y) = xcos(2\pi y) + ysin(2\pi x) \tag{13}$$

where $x \in [-2, 2]$ and $y \in [-2, 2]$. The task is to search the maximum of Fun. 4. In the domain of x and y, the optimal solution is $(1.7625, -2)$ and the maximum is 3.7563. For Fun. 4, the input string is first partitioned into independent groups of 20 bits each. This partitioning does not change during the run.

B. Experimental Results

To study scalability, we consider a range of problem sizes for the first three test problems and the problem size of Fun. 4 is set 2. The results are then used to investigate the growth of the number of function evaluations until successful convergence to the global optimum with respect to the problem size. We also record the change in value of each problem. For 3-deceptive, a population is said to have converged when the proportion of some value on each position reaches 95%. However, the criterion of convergence of De Jong's function 1 is that the proportion of some value on each position of some population reaches 99%. For the Shubert function, the population is said to have converged when it contains over a half of optimal solutions. We set two convergent criterions for Fun.4. One is the maximum iterating times with 280. And the other is that the minimum is greater or equal to 0.95$\times$ maximum for some generation. The population sizes for all problem instances have been determined empirically so that the algorithms meet the criterion of convergence in all of 30 independent runs. The truncation selection has been set to be 50%.

A simple PEA based on genetic algorithms is compared to T-PEA on the three test functions. The simple PEA starts with an initial population contained several randomly subpopulation. In each iteration, the probability of crossover has been set to 60%, and it applies bit flip mutation to the current binary string where each bit is flipped with a small probability 1%.

TABLE I
CONTROL PARAMETERS

	Function size(n)	No. of subpop.	Subpop. size
Dec-3	(30,60,90,120,150)	(4,8,12,16,20)	200
De Jong	(10,15,20,25,30)	(2,6,12,20,30)	100
Shubert	(2,6,10,14,18)	(2,6,10,14,18)	200
Fun. 4	(2)	(4,8,12,16,20)	60

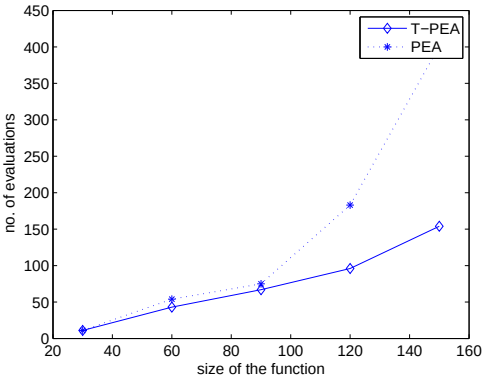

Fig. 5. Results of Dec-3 Function: the average number of iterations in 30 independent runs with respect to the function size.

In following simulations, two algorithms utilize the same control parameters listed in Table 1. In each line, the numbers of the two brackets are in accordance with the corresponding sequence.

The results of the Dec-3 function are presented in Fig.5-Fig.6. The building blocks are non-overlapping, and they which are mapped tightly onto a string are not likely to be disrupted by one-point crossover. Since it is independent of the variable ordering in a string, T-PEA with the BD metric performs better than the compared simple PEA algorithm with one-point crossover in terms of the number of iterations until successful convergence as the problem size grows. The population sizes for the PEA the problem size grows. The population sizes for the PEA and T-PEA ranged from N = 800 for n = 30 to N = 4000 for n = 150. At the same time, the number of each population had been set to 200 to implement the parallel evolutionary. Fig. 5 shows the average number of iterations in 30 independent runs with respect to the function size. In Fig. 6, a simulation results for Dec-3function with function size =120 are presented. It is obviously that T-PEA gives a much better performance in comparison to PEA on Dec-3 function.

Fig. 7-Fig. 8 show the results of the De Jong's function 1. The results of this function are similar to those of the Dec-3 function. The population sizes for the PEA and T-PEA ranged from N = 200 for n = 10 to N = 3000 for n = 30. The number of each population had been set to 100 to implement the parallel evolutionary. In Fig. 7, the overwhelming superiority with fewer iterations of T-PEA is shown. Fig. 8 shows that T-PEA can converge rapidly when function size takes 20.

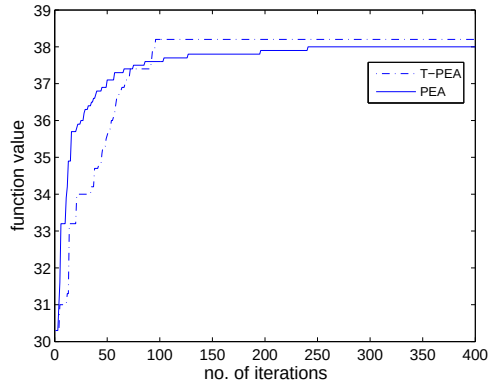

Fig. 6. Results of Dec-3: the change of the average minima in 30 independent runs when the function size is 120.

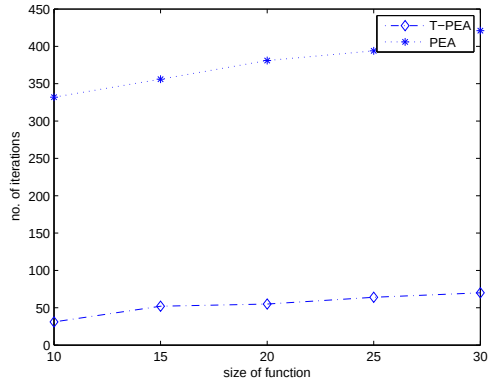

Fig. 7. Results of De Jong's Function 1: the average number of iterations in 30 independent runs with respect to the function size.

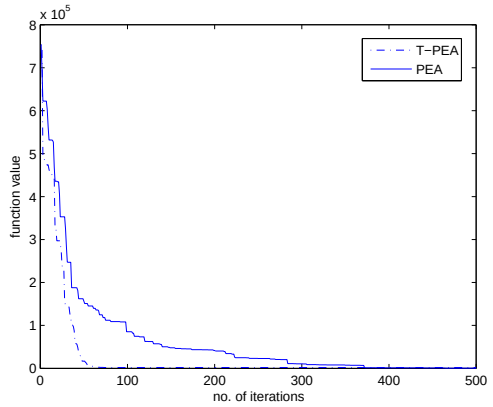

Fig. 8. Results of De Jong's Function 1: the change of the average minima in 30 independent runs when the function size is 20.

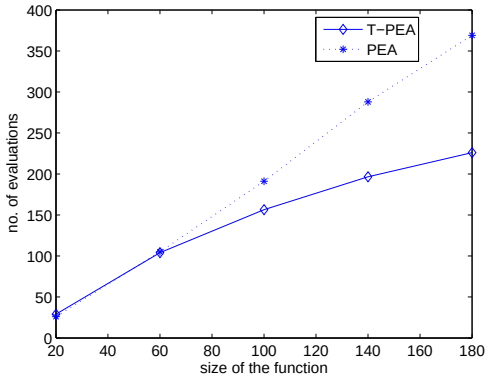

Fig. 9. Results of Shubert function: the average number of iterations in 30 independent runs with respect to the function size.

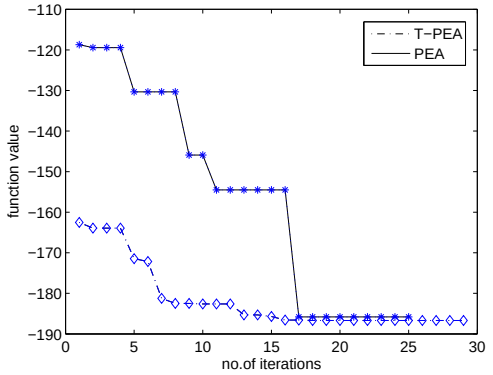

Fig. 10. Results of Shubert function: the change of the average minima in 30 independent runs when the function size is 2.

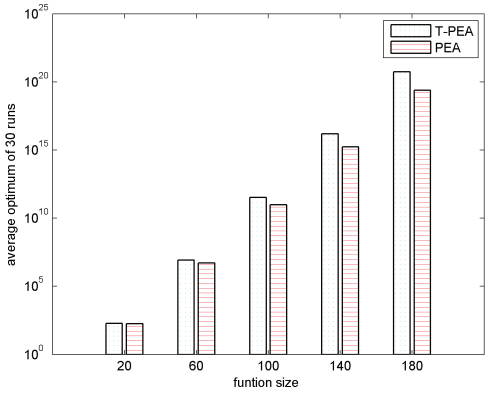

Fig. 11. Comparison Results: T-PEA vs PEA in terms of average minima of 30 independent runs with respect to the function size .The vertical axis displays the absolute values of the average minima in logs.

Fig. 9-Fig.11 show results of Shubert function. The population sizes for the PEA and T-PEA ranged from N = 400 for n = 2 to N = 3600 for n = 18. The number of each population had been set to 200 to implement the parallel evolutionary. Fig. 9 shows the average number of iterations with respect to the function size. Fig. 10 shows the change of the average minima in 30 independent runs when the function size is 2. As Shubert has many local minima, the results show that PEA can not gain the real minima under the convergent condition and falls into local optimum. Fig. 11 presents the average minima of T-PEA and PEA of 30 independent runs with respect to the five function size. The vertical axis displays the absolute values of the average minima in logs. Obviously, as the function size grows, the difference between the optimums returned by PEA and T-PEA increases markedly.

Fig. 12-Fig.14 show results of Fun. 4. The function size of Fun. 4 is 2, that is the two independent variables x and y. The population sizes for the PEA and T-PEA ranged from 4 to 20. And each population contains 60 individuals which is formed by binary strings of length 40. T-PEA can find the maximum 3.7563 for all population sizes and PEA can do this only for sizes 8, 12, 16, 20. However, T-PEA only need a few times of iteration. That is, within a short time, the minimum is greater or equal to 0.95 $\times$ maximum at some generation. With the same data sets, PEA requires iterations as many as 280 times on population sizes 8 to 20. Although on population size 4, there are only average 34.33 times of

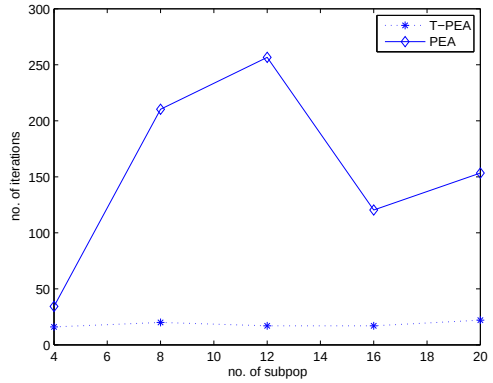

Fig. 12. Comparison Results: T-PEA vs PEA in terms of average number of iterations to achieve optimal value of 30 independent runs for each different size of population.

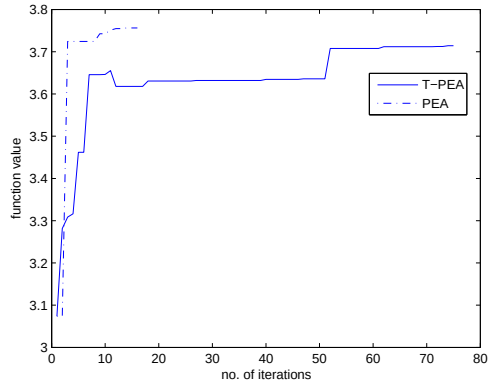

Fig. 13. Comparison Results: T-PEA vs PEA in terms of average maximum of 30 independent runs with respect to the population size 4.

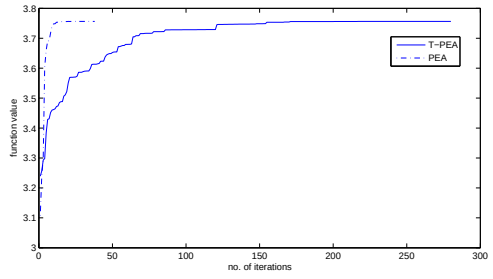

Fig. 14. Comparison Results: T-PEA vs PEA in terms of average maximum of 30 independent runs with respect to the population sizes 8, 12, 16, 20.

iteration for PEA, the convergent criterion has been reached without finding a satisfactory solution. The optimal obtained by PEA on population size 4 is 3.7142 which is less than the global maximum 3.7563. Fig.12 shows the average number of iterations to achieve optimal value for each different size of population. It is obviously, T-PEA is a faster convergence algorithm than PEA. Fig. 13 presents the average maximum of T-PEA and PEA of 30 independent runs with respect to the population size 4. PEA obtains the optimal 3.7142 with 75 iterations and T-PEA obtains the optimal 3.7563 with 16 times of iteration. Fig. 14 presents the average maximum of T-PEA and PEA of 30 independent runs with respect to the other four population sizes (8,12,16,20). These results once again indicate T-PEA's superiority.

VI. Conclusions and Future Work

In this paper we have shown that learning tree model based on BD metric is not expansive, and given a theoretical analysis to create tree model from data set. At the same time, we described the frame of parallel evolutionary algorithms based-on tree models. The experiments have shown that the proposed algorithm outperforms the simple PEA on n-dimensional parabola with spherical constant-cost contours and even on decomposable problems with tight building blocks as the function size grows. The gap of the number of iterations until successful convergence between the two algorithms would significantly enlarge with the function size grows. In addition, T-PEA succeeds BOAs advantages that T-PEA is independent of the ordering of the variables in a string and therefore changing this would not affect the performance of the algorithm. In our future research we will try given some empirical results about parallel evolutionary algorithms based-on tree models. And in addition, we will try to describe difference to construct tree models between based-on minimum description length metric and BD metric.

References

[1] J. L. Andrews and P. D. McNicholas, "Using evolutionary algorithms for model-based clustering, *Pattern Recognition Letters*, vol. 34, no. 9, pp. 987-992, 2013.

[2] Jinxing Che, "Support vector regression based on optimal training subset and adaptive particle swarm optimization algorithm," *Applied Soft Computing*, vol. 13, no. 8, pp. 3473-3481, 2013.

[3] E. G. Bekele, C. L. Lant, S. Soman and G. Misgna, "The evolution and empirical estimation of ecological-economic production possibilities frontiers, *Ecological Economics*, vol. 90, pp. 1-9, 2013.

[4] S. Nesmachnow, H. Cancela and E. Alba, "A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling, *Applied Soft Computing*, vol. 12, no. 2, pp. 626-639, 2012.

[5] J. Toutouh, S. Nesmachnow, E. Alba, "Fast energy-aware OLSR routing in VANETs by means of a parallel evolutionary algorithm, *Cluster Computing*, vol. 16, no. 3, pp. 435-450, 2012.

[6] L. Wang, Q. K. Pan, P. N. Suganthan, W. H. Wang and Y. M. Wang, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems, *Computers & Operations Research*, vol. 37, no. 3, pp. 509-520,2010.

[7] E. Cantú-Paz, "A Survey of Parallel Genetic Algorithms, *Calculateurs paralleles, reseaux et systems repartis*, vol. 10,no. 2, pp. 141-171,1998.

[8] E. Cantú-Paz and D. E. Goldberg, "Efficient Parallel Genetic Algorithms: Theory and Practice, *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 221-238, 2000.

[9] E. Alba, J.M. Troya, "A Survey of Parallel Distributed Genetic Algorithms, *Complexity*, vol. 4, no. 4, pp. 31-52, 1999.

[10] S.Yussof, R. A. Razali,. , Ong Hang See, A. A. Ghapar, M. M Din, "A Coarse-Grained Parallel Genetic Algorithm with Migration for Shortest Path Routing Problem, in *11th Conf. Rec. 2009 IEEE High Performance Computing and Communications*, pp. 615-621,2009.

[11] E. Alba, J. M. Troya, "An Analysis of Synchronous and Asynchronous Parallel Distributed Genetic Algorithms with Structured and Panmictic Islands, in *Parallel and Distributed Processing, J. Rolim et al. (eds.), Springer Berlin Heidelberg, 1999*, pp. 248-256.

[12] Udo Kohlmorgen. FeinkÄornige, "parallele genetische Algorithmen, *Ph. D. thesis, Univ. of Karlsruhe, Germany, German*, 1999.

[13] H. Mühlenbein and T. Mahnig, "Evolutionary optimization using graphical models, *New Generation Computing*, vol. 18, no. 2, pp. 157-166,2000.

[14] S. Baluja and S. Davies, "Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space, *Carnegie Mellon University Technical Report* No. CMU-CS-97-107, 1997.

[15] M. W. Hauschild, M. Pelikan, K. Sastry, D. E. Goldberg, "Using Previous Models to Bias Structural Learning in the Hierarchical BOA, *Evolutionary Computation*, vol. 20, no. 1, pp. 135-160, 2012.

[16] Martin Pelikan, David E.Golaberg and Erick Cantú-Paz, "The Bayesian optimization algorithm, *Univ. Illinois, Urbana-Champaign, IlliGAL Report* No. 98013, 1998.

[17] M. Pelikan, D. E. Goldberg and E. Cantú-Paz, "The Bayesian optimization algorithm, population sizing, and time to convergence, *Lawrence Livermore National Lab., CA (US)*, 2000.
[18] M. Pelikan and D. E. Goldberg, "Hierarchical Bayesian Optimization Algorithm, *Studies in Computational Intelligence*, vol. 33, pp. 63-90, 2006.
[19] C. F. Lima, M. Pelikan, F. G. Lobo, D. E. Goldberg, "Loopy Substructural Local Search for the Bayesian Optimization Algorithm, in *Lecture Notes In Computer Science*, LNCS 5752, 2009, pp. 61-75.
[20] L. Claudio, L. Fernando, P. Martin, G. David, "Model accuracy in the Bayesian optimization algorithm, *Soft Computing - A Fusion of Foundations, Methodologies & Applications*, vol. 15, no. 7, pp. 1351-1371. 2011.
[21] D. Heckerman, D. Geiger and D. H. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Machine learning*, vol. 20, no. 3, pp. 197-243, 1995.
[22] D. M. Chickering, D. Heckerman and C. A. Meek, "A Bayesian approach to learning Bayesian networks with local structure," in *Proc. Thirteenth conf. on Uncertainty in artificial intelligence. Morgan Kaufmann, 1997*, pp. 80-89.
[23] E. Alba and M. Tomassini, "Parallelism and Evolutionary Algorithms, *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 443-462, 2002.
[24] C. K. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees, *IEEE Transactions on Information Theory*, vol. IT-14, no. 3, pp. 462-467,1968.
[25] E. Cantú-Paz, "Markov chain models of parallel genetic algorithms, *IEEE Transactions on Evolutionary Computation*, vol. 4, no.3, pp. 216-226, 2000.
[26] Meilă M, "An Accelerated Chow and Liu Algorithm: Fitting Tree Distributions to High-Dimensional Sparse Data,"*Proceedings of the Sixteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. 1999*,pp. 249-257.
[27] Bouckaert R R, "Properties of Bayesian belief network learning algorithms," in *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc. 1994,* pp. 102-109.
[28] Li Yanying, Yang Youlong, Zhu Xiaofeng, Dou Xiaoxia, "Towards fast and efficient algorithm for learning markov blankets," *IAENG International Journal of Computer Science*, vol. 42, no. 1, pp. 17-22, 2015.
[29] Cooper G F, Herskovits E, "A Bayesian method for the induction of probabilistic networks from data," *Machine learning*, vol. 9, no. 4, pp. 309-347, 1992.
[30] D. Goldberg, "Simple Genetic Algorithms and the Minimal,Deceptive Problem, in *Genetic Algorithms and Simulated Annealing, Morgan Kaufmann, 1987*, pp. 74-88.
[31] L. D. Whitley, "Fundamental principles of deception in genetic search, in *Foundations of genetic algorithms, G. J. Rawlins, Ed. Morgan Kaufmann, San Mateo, 1991*, pp. 221-241.
[32] H. A. Simon, The Sciences of the Artificial. *Cambridge, CA: MIT*, 1968.
[33] K. A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems, *Ph. D. Thesis, University of Michigan, 1975*.

Youlong Yang was born in Pucheng, Shaanxi, China in 1967. He received his B.S. degree, M.S. degree in the Department of Mathematics from Shaanxi Normal University, in 1990 and 1993, respectively. And received the Ph.D. degree at Northwestern Polytechnical University in 2003. In 2006, he was out bounded post-doctoral mobile stations in Xidian University, and then was sent to the United States University of Rochester as a national public school student in 2007. His scientific interest is Probabilistic Graphical Models and Time Series.

Youlong is full professor, PhD supervisor in Xidian University and executive director of the Mathematical Association of Shaanxi Province.

Prof. Yang has published more than 40 papers. His papers were published as the first-named author in journals such as Information Sciences, International Journal of Approximate Reasoning, Acta Application Mathematic, Control Theory and Applications. He has received one reward of Shaanxi Provincial Science and Technology and two Bureau departmental level research awards and so on.

Yanying Li was born in Songyuan, Jilin Province, China in 1981. She received her B.S. degree, M.S. degree in the Department of Mathematics from Jilin Normal University in 2004 and 2007 respectively. And began work for a doctorate at Xidian University in 2011. She is interested in Probabilistic Graphical Models and Time Series.

Yanying is a Ph.D in Xidian University and a lecturer in Baoji University of Arts and Sciences.

Dr. Li has published 17 papers in English and in Chinese.

Wensheng Wang received his M.S. degree in School of Mathematics and statistics at Xidian University in 2012. His scientific interest is Probabilistic Graphical Models and Time Series.

Wenming Yang begins to work for a M.S. degree in School of Mathematics and statistics at Xidian University in 2014. His scientific interest is Probabilistic Graphical Models and Time Series.