Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 9 (3): 451-468, 2015
ISSN 1819-4311 / DOI: 10.3923/jse.2015.451.468

Comparison of Effects of Different Learning Methods on Estimation of Distribution Algorithms

[1,2,3]Caichang Ding, [1,3]Lixin Ding and [2]Wenxiu Peng
[1]State Key Laboratory of Software Engineering, Wuhan University, Wuhan, 430072, China
[2]School of Computer Science, Yangtze University, Jingzhou, 434023, China
[3]School of Computer, Wuhan University, Wuhan, 430072, China

Corresponding Author: Caichang Ding, School of Computer, Wuhan University, Wuhan, 430072, China

ABSTRACT

This study investigates Estimation of Distribution Algorithms (EDAs) based Bayesian networks with KS learning method. The EDAs based Bayesian networks are used to analyze the effect of learning the best structure in the search. By using KS learning method that can learn optimal Bayesian networks, two important issues in EDAs are studied. First, we discuss that whether learning a more perfect depending model leads to a better behave of EDAs. Second, when a perfect learning is accomplished, we are able to observe that how is the problem structure transformed into the probabilistic model. Several different kinds of experiments have been conducted. The experimental results show that when the accuracy of the learning is increasing, the quality of the problem information learned by the probabilistic model can also be improved. However, the improvements in model accuracy do not mean a more efficient search at all times.

Key words: EDAs, learning structure, problem structure, probabilistic model

INTRODUCTION

EDAs (Larranaga and Lozano, 2002; Hauschild and Pelikan, 2011; Muelas *et al*., 2014) are a new kind of Evolutionary Algorithms (EAs) that use probabilistic models instead of the typical genetic operators used by Genetic Algorithms (GAs) (Reeves, 2010; Wang and Chen, 2013). Relevant features of the search space in EDAs are extracted by machine learning methods. In EDAs, employing a probabilistic model represents the collected information which is used to generate new individuals later. By this way, probabilistic models can lead the search to hopeful areas of the search space.

Mathematically, an optimization problem can be seen as the minimization or maximization of a given function. Thus, optimization problems can be formulated as:

$$x^* = \arg_x \max f(x) \quad (1)$$

where, f: S→R is called the objective function or fitness function, $x = (x_1,....,x_n) \in S$ is a candidate solution of the problem and S is named the problem space. In most cases, the optimum x^* is not unique. In this study, the problem space S is an n dimensional discrete space.

Because the EDAs (Larranaga and Lozano, 2002; Muelas *et al*., 2014) can capture the structure of the problem, EDAs are considered to be more efficient than GAs. In EDAs, the specific

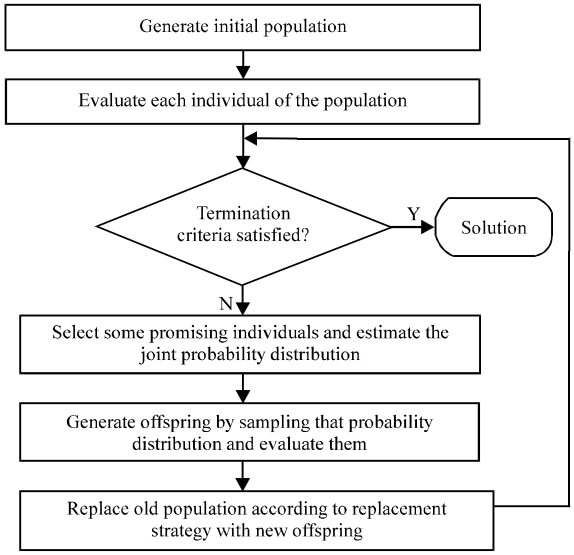

Fig. 1: Flow chart of estimation of distribution algorithms

interactions among the variables of solutions in the problem are taken into mind. In evolutionary algorithms, the interactions are displayed implicitly in mind, whereas in EDAs, the interrelations are showed explicitly through the joint probability distribution associated with the individuals of variables selected from each generation. The probability distribution is calculated according to a population of selected candidate solutions of previous generation. Then, offsprings are sampled from this probability distribution generate. Neither crossover nor mutation has been used in EDAs. Figure 1 displays the flow chart of estimation of distribution algorithms.

An EDA has such basic elements (Larranaga and Lozano, 2002). Encoding of candidate solutions, objective function, selection of parents, building of a structure, generation of offspring, selection mechanism and algorithm parameters like population size, selection size, etc.

Different EDAs (Larranaga and Lozano, 2002; Ahn *et al*., 2012) mainly differ in the kind of probabilistic models employed and the approaches used to learn, then sample from the obtained models. Bayesian networks is one of the models that has been widely used in EDAs. One of the advantages of EDAs that employ these kinds of models is that the complexity of the learned structure relies on the characteristics of the selected individuals. Moreover, the analysis of the models learned during the search can provide useful information about the problem structure.

One important issue in EDAs is to study how the selections of the probabilistic models and of the learning and sampling methods can offer the reasonable balance between exploration and exploitation. There are many studies which report about the way in which the performance of EDAs can remarkably vary according to the changes in the parameters that determine the learning of the models (Jin and Jin, 2014; Ceberio *et al*., 2014; Chang and Chen, 2014). Moreover, in the case of EDAs which use Bayesian networks, the role of the parameters which penalize the complexity of the networks has been studied, a similar analysis of the accuracy of the methods used for finding the best model and its influence in the behavior of EDAs has not been carried out yet.

How the characteristic of the search space are reflected in the learned probabilistic models is another related and important issue in EDAs. This issue has received special attention from the EDA community and is essential to understand the mechanisms that enable EDAs to efficiently

sample from the problem space during the search process. However, the question of analyzing the relationship between the problem space and the structure of the learned probabilistic models becomes more and more difficult because of the next two main reasons: The search process is random in the EDAs and the methods used when learning the models can only detect approximate, suboptimal, structures.

In this study, we provide a choice that allows to investigate the influence that learning an optimal models of the population produce in the response of EDAs based on Bayesian networks. Moreover, our study provide a solution to obtain more accurate information about the relationship among the problem structure, distributions of the solutions and the probabilistic models learned during the search process.

This study is based on the use of methods for learning the best Bayesian networks. Methods which conduct best Bayesian structure learning, compute according to a set of individuals and a predefined score, the network structure that optimizes the score and the BIC score used in this study. Because learning the best Bayesian network is a NP-hard problem, it is must to set constraints on the maximum number of variables so as to they can be deal with by some methods. Usually, we use dynamic programming algorithms to learn the structure.

The learning of probabilistic models to extract the relevant information that the selected individuals can contain about the problem is a fundamental step of the algorithm. Regarding this issue, we wonder how the search and the behavior of the EDA is influenced by the accuracy of the learning method.

BAYESIAN NETWORKS

The EDAs considered in this study employ factorizations which can be represented by Bayesian networks. Bayesian networks (Darwiche, 2010; Bensi *et al*., 2013) which is called belief networks are a kind of probabilistic graphical model. This kind of probabilistic models is one of very popular paradigms that can deal with probability distributions efficiently in modeling uncertain information. The domain of expert systems is one of the most important sources for the development of Bayesian networks. Moreover, in the past few years, Bayesian networks have obtained considerable attention in domain of the machine learning. As a result of this attention, more and more study and tutorials have appeared. Thus, besides expert systems, Bayesian networks are also applied in classification problems, bioinformatics and optimization.

Bayesian networks are the product of associating probability and graph theory (Koller and Friedman, 2009), similarly with any other probabilistic graph model. The graph consist of the model encodes a number of conditional independences related to the probability distribution. Let $x = (x_1,..., x_n)$ be an n dimensional discrete random variable. The factorization of the joint probability distribution $p(X = x)$ for X can be graphically expressed by a Bayesian network, where $x = (x_1,..., x_n)$ is an assignment of the random variable X. More specifically, a Bayesian network can be expressed as a pair (s, θ_s), where, s is a directed acyclic graph that is model structure and θ_s is the set of parameters associated to the structure that is model parameters. The structure s determines the set of conditional dependences among the random variables of x. According to the structure s, the joint probability distribution p(x) can be factorized by means of marginal and conditional probability functions. Specifically, the probability distribution factorizes according to the graph as:

$$p(x) = \prod_{i=1}^{n} p(x_i \mid pa_i) \quad (2)$$

where, pa_i denotes a value of the variables Pa_i that is the parent set of x_i in the graph structure s.

The local probability distributions in the factorization are those which is induced by means of the product that appears in Eq. 3. We suppose that these local probability distributions depend on the parameters $\theta_s = (\theta_1,...,\theta_n)$. So, Eq. 3 could be rewritten by specifying the parameters:

$$p(x \mid \theta_s) = \prod_{i=1}^{n} p(x_i \mid pa_i, \theta_i) \tag{3}$$

Suppose that the variable x_i has r_i possible values, thus the local probability distribution $p(x_i \mid pa_i^j, \theta_i)$ is an unbounded discrete distribution:

$$p(x_i^k \mid pa_i^j, \theta_i) = \theta_{ijk} \tag{4}$$

where, $pa_i^1,..., pa_i^{q_i}$ represent the q_i possible values of the parent set Pa_i. The parameter θ_{ijk} denotes the probability of variable x_i which takes in its k-th value, at the same time, the set of its parents' variables takes in its j-th value. Therefore, the local parameters are determined by $\theta_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$.

Bayesian network learning: In order to look for a Bayesian network (Friedman *et al*., 1997; Hauschild *et al*., 2012) which can make us to represent and deal with the uncertain knowledge of a specific field, setting both the structure and the parameters is very necessary. The structure and conditional probabilities that is necessary for describing the Bayesian network can be provided either externally by experts, by machine learning from datasets or by mixing both of these methods. In this study, we mainly focus on the second method. Besides, when the structure has been automatically learned, it can provide us with perceptions into the interactions between the variables of the field.

The learning step can be separated into two subtasks that are structural learning and parameter learning (Daly *et al*., 2011; Buntine, 1991; Barber, 2012). Although there are different approaches to learn the structure of a Bayesian network, we mainly focus on the so-called score plus search method. This kind of methods copes with the structure learning as an optimization problem. Thus, the steps of learning a Bayesian network can be expressed as follows. Given a data set D containing N cases, $D = \{x_1,...,x_N\}$, finding the structure s* such that:

$$s^* = \arg\max_{s \in S^n} g(s,D) \tag{5}$$

where, g (s,D) is the score which measures the quality of any given structure s related to the data set D and S^n is the set of all possible Directed Acyclic Graphs (DAG) which have n nodes. A number of relevant and used heuristic techniques such as greedy search, simulated annealing, particle swarm optimization, genetic algorithms and ant colony optimization have been used in this task.

If score can be decomposed in presence of complete data sets, it is the one of the desirable character. These scores can be decomposed in sub-scores related to every node X_i and its parents Pa_i in the structure s. Formally, we can express a decomposable score as:

$$g(s,D) = \sum_{i=1}^{n} g_D(X_i, Pa_i) \tag{6}$$

where, the g_D is the sub-score function. As a result of the decomposability, it is computationally more efficient when the local search is carried out because when we add an arc into the network, it is only necessary to evaluate the set of nodes involved by this change.

Concerning the accomplishment of the score g (s, D), there are some choices that can be considered. For example, there are marginal likelihood, the log likelihood probability penalty and information theory based entropy which we can use. In this study, we will employ the Schwarz criterion, which is also called Bayesian Information Criterion score (BIC). This score is obtained as following steps. From a known dataset $D = \{x_1,...,x_N\}$, we could calculate for any Bayesian network structure s the maximum likelihood estimate $\hat{\theta}_S$ for the parameters set θ_s and the related maximized log likelihood:

$$\begin{aligned} \log p(D \mid s, \theta_s) &= \log \prod_{w=1}^{N} p(x_w \mid s, \theta_s) \\ &= \log \prod_{w=1}^{N} \prod_{i=1}^{n} p(x_{w,i} \mid pa_i, \theta_i) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log(\theta_{ijk})^{N_{ijk}} \end{aligned} \tag{7}$$

where, N_{ijk} represents the number of instances in dataset D in that the variable X_i takes its k-th value x_i^k and Pa_i takes its j-th value Pa_i^j. Because the maximum likelihood estimate of parameters θ_{ijk} is calculated by:

$$\hat{\theta} = \frac{N_{ijk}}{N_{ij}}$$

Where:

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk},$$

We have:

$$\log p(D \mid s, \hat{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \tag{8}$$

Usually, we do not employ the log-likelihood function to lead the search process because of two main problems. Firstly, the log-likelihood function is a monotonous increasing according to the complexity of the model structure. Thus, if we employ this score to determine the quality of the

structures during the search process, it could lead searching towards the complete Bayesian network. Secondly, when the number of parameters for each variable increases, the error of the parameter estimation increases too. For purpose of overcoming these problems, we could add a penalty term to the log-likelihood. The equation of the penalized log-likelihood takes as:

$$\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - h(N)\dim(S) \tag{9}$$

where, dim(S) denotes the dimension of the Bayesian network, i.e., dim(S) = $q_i(r_i-1)$. h(N) is a penalty function which is non-negative, it takes:

$$h(N) = \frac{1}{2}\log N$$

in the BIC. Therefore, we can write the BIC as:

$$BIC(s, D) = \log \prod_{w=1}^{N}\prod_{i=1}^{n} p(x_{w,i} \mid pa_i, \hat{\theta}_i) - \frac{1}{2}\log N \sum_{i=1}^{n} q_i (r_i - 1) \tag{10}$$

On the other hand, parameter learning is the numerical analysis of the parameters θ_s which denote the conditional and marginal probability distributions after the factorization determined by the structure s. Although, this study can be dealt by means of different methods, we employ the maximum likelihood estimation in this study. Particularly, when the structure has been learned, the parameters of model can be calculated by using the Laplace correction as follows:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + 1}{N_{ij} + r_i} \tag{11}$$

Emulation: When we have got a Bayesian network by learning, this model could provide us with detailed probabilistic information which we are interested in. Usually, the information which the researcher wants to know is the probability of some events on the basic of special observations. Generally speaking, the probabilities which we are interested in, are not reposited in the Bayesian network obviously. It is necessary to compute in order to obtain them. This course is called probabilistic inference and it is usually an NP-complete problem.

Emulation of Bayesian networks, which is also named stochastic sampling, can be regarded as an option to the exact inference. The Emulation of a given probabilistic graphical model requires to get a sample from the probability distribution for X which the model encodes. Next, the marginal and conditional probabilities involved can be calculated from the sample.

For our goals about EDAs, the intention of the emulation of Bayesian networks is to get a new population in which the probability relations among the random variables of the network are potential. Specifically, for the purpose of sampling from the Bayesian network, the sampling method which we employ is forward. The variable must be sampled after all its parent variables have been obtained. This approach is named Probabilistic Logic Sampling (PLS). Figure 2 presents a pseudo-code of this approach.

```
1    π ←Ancestral ordering of the nodes in the baysian network
2    for j = 1 to N
3       for i = 1 to n
4          X_{jπ(i)}  ←Randomly generate a value form p(x_{π(i)} |pα_{π(i)})
5    done
6    done
```

Fig. 2: Pseudo-code of the probabilistic logic sampling method

METHODOLOGY

Learning methods: When we have defined a score to assess Bayesian networks, we have to run a search process to look for the Bayesian network which can return the best score given the data. In this study, we use two different methods: Approximate learning method and exact learning method.

Learning an approximate model: In practical applications, we must to look for an suitable model structure as soon as possible. Thus, a simple learning method which can find a relatively good structure, even though it is not best, is preferred. There are a number of learning methods which can be employed for this task. However, a specific search algorithm which is called Algorithm B (Buntine, 1991) which is typically used by most of EDAs based Bayesian network.

Algorithm B is a greedy search algorithm and its pseudo-code is presented in Fig. 3, where D is a data structure which deposits the information needed to deal with the candidate arcs which should be added into the network. At the beginning, algorithm B starts from an arc-less structure which represents independently among the variables and at each iteration, an arc is added into network that can increase the score greatly. The algorithm stops when no arc that can increase the score any more, can be added into the network.

Learning an exact model: Since looking for a best Bayesian network that maximizes the score given the data is an NP-hard problem, for a long time the target of learning best Bayesian networks was restrained to problems with a very small number of variables. The first algorithm that carried out this kind of learning in less than super-exponential complexity according to n was introduced in (Koivisto and Sood, 2004) which is called algorithm KS . For the investigation carried out in this study, we employ the algorithm presented in (Koivisto and Sood, 2004) to learn Bayesian networks in the EDAs. This algorithm is effective for $n<33$.

In the next, we try to introduce the basic principles of KS learning algorithm used in this study. In this learning method, the Bayesian network structure S is defined as a vector $s = (s_1,...,s_n)$, where s_i is the subset of X which is the of parent sets of X_i Moreover, this algorithm employs an sequencing of the variables X. In this sequence, the i-th variable is represented by ord_i. The structure $s = (s_1,.., s_n)$ is considered to be corresponding to an sequence ord when all the parents of the variable locate before the node in the sequence.

Another crucial notion in this algorithm is the sink node. In a Directed Acyclic Graph (DAG), there is at least one node which has no outgoing arcs, thus at least one node is not a parent of any other node. We define these nodes as sink nodes.

1 Begin with an arcless strcture

2 Compute $D[X_j \rightarrow X_i] = g_D(X_i, X_j) - g_D(X_i)$ for all distic X_i, X_j

3 do

4 find the largest $D[X_j \rightarrow X_i]$ and add an arc from X_j to X_i in the structure

5 $D[X_j \rightarrow X_i] = g_D(X_i, Pa_i \cup X_j) - g_D(X_i)$ for all disctinct X_i, X_j not belonging to Pa_i

6 $D[X_j \rightarrow X_i] = -\infty$

7 until every $D[X_j \rightarrow X_i] < 0$

Fig. 3: Pseudo-code for algorithm B

1 Calculate the local scores for all $n2^{n-1}$ diffrerent (variable, variable set) Pairs

2 Using the local scores, look for the best parents for all $n2^{n-1}$ (Variable, variable set) pairs

3 Look for the best sink for all 2^n variable sets

4 Using the results from step 3, look for a best ordering of the vatiables

5 Look for a best network using results computed in step 2 and 4

Fig. 4: Pseudo-code for algorithm KS

In this algorithm, the data set D is dealt in a special way and it employs two types of data tables. Given $W \subseteq x$, initially we use the contingency table CT(W) which is a sequence of the frequencies of different data-vectors in D^W, where D^W is the data set for W variables. However, the primary work is to computer conditional frequency tables $CFT(X_i, W)$ which record the information that how many times different values of the variable X_i appear meanwhile with different vectors $x_j^{w-\{xi\}}$ in the data.

As discussed before, many commonly used scores could be decomposed to the sum of some local scores. Therefore, the score g(s, D) can be formulated as:

$$g(s,D) = \sum_{i=1}^{n} score(CFT(X_i, s_i)) \tag{12}$$

where, the score is used to computing conditional frequency tables. Figure 4 gives the primary procedures of the method.

The first step is the primary procedure. It begins by computing the contingency table for all the variables and all smaller variable subsets. Next, for each contingency table, the conditional frequency table is computed for each variable which appears in the contingency table. These conditional frequency tables could then be used to computing the local scores for all parent set given

a variable. There are $n2^{n-1}$ local scores that are deposited in a table which will be the foundation of the algorithm.

Having computing the local scores, the best parents set for X_i are either the candidate set C or sub set of candidate sets $C\backslash\{c\} \mid c \in C$. This search process must be computed for all 2^{n-1} variable sets involved in X_i.

Step 3 of the algorithm is based on the next fact: The optimal network G* of a variable set W must have a sink node. For G* is a network which has the highest score, the sink node must have incoming arcs from its best possible set of parents. In this manner, the rest of the nodes and the arcs must form the best possible network for variables W/{s}. Thus, the optimal best sink for W, sink*(W), is the node which makes the sum max between the local score for sink node and the score for the network S without the sink node.

As we have the optimal sink nodes for all 2^n variable sets, it is feasible to produce the optimal ordering ord* in reverse order. Next, for each location from n to 1, in the ordering ord^*_i, we must reserve the optimal sink node for the set $U^n_{j=i+1}\{ord^*_j(X)\}$.

Having got a optimal ordering and a table with the optimal parents for any candidate set, it can acquire a optimal network consistent with the determined ordering. For the i-th variable in the best ordering, the optimal parents from its predecessors are selected.

Experimental design: The Estimation of Bayesian Networks Algorithm (EBNA) (Etxeberria and Larranaga, 1999; Blanco *et al.*, 2003) is one of estimation of Distribution Algorithms which is based on Bayesian networks. EBNA can employ statistics of unlimited order in the factorization of the Joint Probability Distribution (JPD). This joint probability distribution is encoded using a Bayesian network which is obtained from the database containing the selected individuals in every generation. It has been carried out with good consequence to different kinds of problems (Larranaga and Lozano, 2002). Other EDAs have been put forward in Fan *et al.* (2011), Shim *et al.* (2012) and Li *et al.* (2014). The pseudo-code of EBNA is presented in Fig. 5.

1 $BN_{t=0} \leftarrow (S_0, \theta^0_{S_0})$, where S0 is an arc-less structrue and $\theta^0_{S_0}$ is uniform

2 $D_{t=0} \leftarrow$ Generate N individuals from $BN_{t=0}$

3 Do

4 $D_t \leftarrow$ Evaluate individuals

5 $D^{Se}_t \leftarrow$ Select M<N individuals from D_t according to a selection method

6 $S_t \leftarrow$ Learn and get a network structure

7 $\theta^t_{ijk} \leftarrow$ compute θ^t using D^{Se}_t from the data set

8 $BN_t \leftarrow (S_t, \theta^t_{St})$

9 $D_{t+1} \leftarrow$ sample N individuals from the network BN_t and produce new population

10 t = t+1

11 untill stop condition is met

Fig. 5: Pseudo-code for algorithm EBNA

To investigate the influence of learning method on the performance of EDAs based Bayesian network, we have tested two different editions of EBNA. The first EBNA edition, named $EBNA_B$, implements algorithm B to search for the model which is a local search method. The second edition, named $EBNA_{KS}$, implements the KS learning algorithm to search for the model, which is an exact search method.

The two editions of the algorithms are only different in the method employed to learn the Bayesian network model at every generation. The parameters settings are explained later.

In this part, we present a number of functions that represent different classes of problems and which are used in this study to test the behavior of EDAs.

Function benchmark: Let:

$$u(x) = \sum_{i=1}^{n} x_i$$

f(x)be a function satisfying that, $\forall x, y \in \{0, 1\}^n$, f(x = f(y) if u(x) = u(y). This function is defined according to value of u(x).

There are a kind of functions where the difficulty is given by the interactions that arise among subsets of variables. One example of this class of functions are deceptive functions:

$$\text{Deceptive3}(x) = \sum_{i=1}^{\frac{n}{3}} f_{3dec}(x_{3i-2}, x_{3i-1}, x_{3i}) \tag{13}$$

where, f_{3dec} is defined according to the function u:

$$f_{3dec} = \begin{cases} 0.9 & \text{for} \quad u = 0 \\ 0.8 & \text{for} \quad u = 1 \\ 0.0 & \text{for} \quad u = 2 \\ 1.0 & \text{for} \quad u = 3 \end{cases} \tag{14}$$

$$u(x) = \sum_{i=1}^{n} x_i \tag{15}$$

Another function used in this study is the function SixPeaks, which is a variant of the FourPeaks problem and it can be defined as:

$$\text{SixPeaks}\ (x, t) = H(x, t) + R(x,t) \tag{16}$$

Where:

$$H(x, t) = \max\{\text{tail}(0, x), \text{head}\ (1, x), \text{tail}(1, x), \text{head}(0,x)\} \tag{17}$$

- Tail (b, x) is number of contiguous trailing b in x
- Head (b, x) is number of contiguous trailing b in x

The target is to maximize the function. When the number of variables is even, there are 4 global optima for this function, which is located at the points:

$$(\overbrace{0,\ldots,0}^{t+1},1,\ldots 1),\ (0,\ldots,0,\overbrace{1,\ldots,1}^{t+1})$$

$$(\overbrace{1,\ldots,1}^{t+1},0,\ldots,0),\ (1,\ldots,1,\overbrace{0,\ldots,0}^{t+1})$$

These solutions are very difficult to obtain because they are isolated. On the other hand, two local optima which is (0, 0,...,0) and (1, 1,...,1), are very easily got.

Here, we set the value of t equal to:

$$\frac{n}{2}-1$$

Parameter settings: According to the primary steps of the EDA (Hauschild *et al.*, 2009; Armananzas *et al.*, 2011; Martins and Delbem, 2014). It deals with a population with N individuals. At the beginning, the original population is sampled from a uniform distribution, therefore, all the individuals in the population have the same sampling probability. Each generation begins with choosing a subset of hopeful individuals from the population. In this procedure, we employ truncation selection in which the threshold is 50%. Therefore, we get half of the individuals whose fitness value is best. The next procedure is to learn and get a probabilistic model from the population of selected individuals. This is the only procedure where the EDAs that we will find differ. As soon as the probabilistic model is built (Griffiths *et al.*, 2010; Abdollahzadeh *et al.*, 2012; Bonawitz *et al.*, 2014), we can generate the new population. For purpose of doing that, we sample from the probabilistic model to get N new solutions and then we add them into the current population. The new population is comprised of the N best individuals which is selected from the 2N individuals available (Chen *et al.*, 2010; Boussaid *et al.*, 2013; Lima *et al.*, 2007).

As previously said, each EDA investigated in this study employs factorizations which could be encoded by means of Bayesian networks. Thus, the same methods can be employed both to get the corresponding parameters and to generate the new solutions. As explained above, we estimate the parameters by maximum likelihood and the new individuals is sampled by PLS.

RESULTS

In this part, the experimental results by $EBNA_B$ and $EBNA_{KS}$ solving for function Deceptive3 with n = 15 and function Sixpeaks with n = 16 are showed. We compare the frequency matrices calculated by $EBNA_B$ and $EBNA_{KS}$ to solving the functions and study some patterns found in the structures of the models which are learned.

DISCUSSION

We begin by using a population size of 350. Figure 6 and 7, respectively present the frequency matrices calculated from the models learned by $EBNA_B$ and $EBNA_{KS}$ solving for the function Deceptive3 with n = 15. Both of the algorithms are able to get the dependencies of variables corresponding to interactions in the problem.

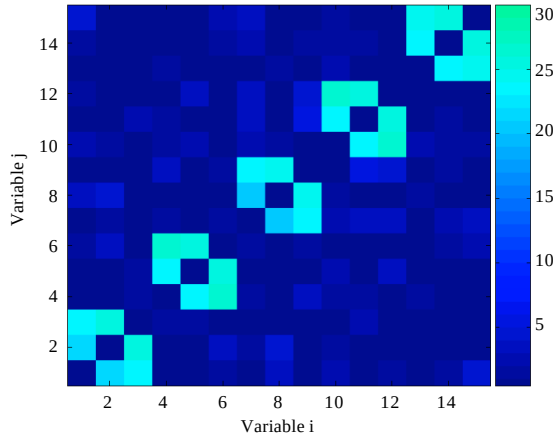

Fig. 6: Frequency matrices learned from the models by $EBNA_B$ for function Deceptive3 with N = 350

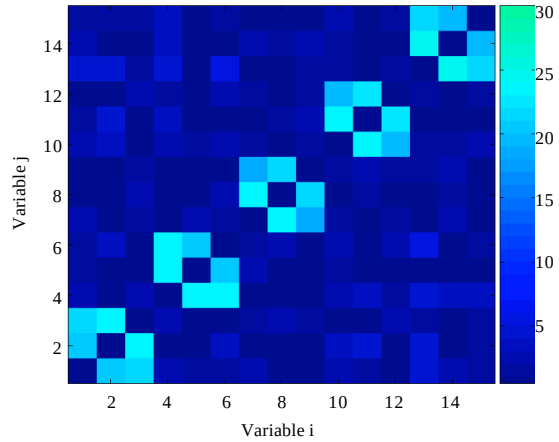

Fig. 7: Frequency matrices learned from the models by $EBNA_{KS}$ for function Deceptive3 with N = 350

It can be seen that the models with some dependences which are not obviously found from the function structure. This is especially obvious for the $EBNA_{KS}$ algorithm. The reason for this phenomenon is that KS learning method is more vulnerable to the over fitting of the data which takes place with the small population size. Thus, we increase the population size and set it to N = 1000, then do the same experiment for this function again. The frequency matrices calculated from the models learned by $EBNA_B$ and $EBNA_{KS}$ are shown in Fig. 8 and 9, respectively, when solving for the function Deceptive3 with n = 15. According to the two figures, we can see the influence of increasing the population size on the relationship of learned dependencies. It can be seen that false correlations have nearly disappeared in the learned models. Both of the two algorithms are able to learn more exact models with a population size which is 1000.

We carry out a similar investigation for the function Sixpeaks with n = 16. Figure 10 and 11, respectively present the frequency matrices learned by $EBNA_B$ and $EBNA_{KS}$ with a population size

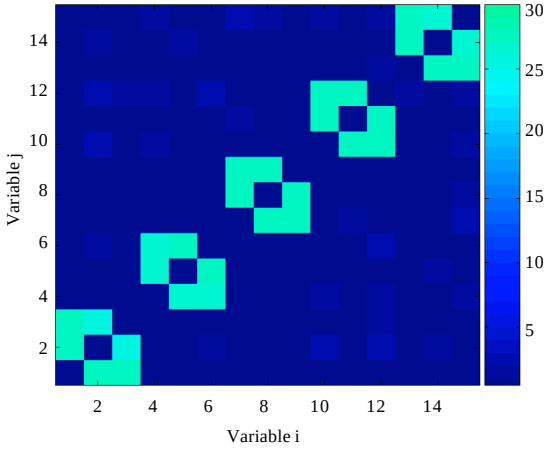

Fig. 8: Frequency matrices calculated from the models learned by $EBNA_B$ for function Deceptive3 with N = 1000

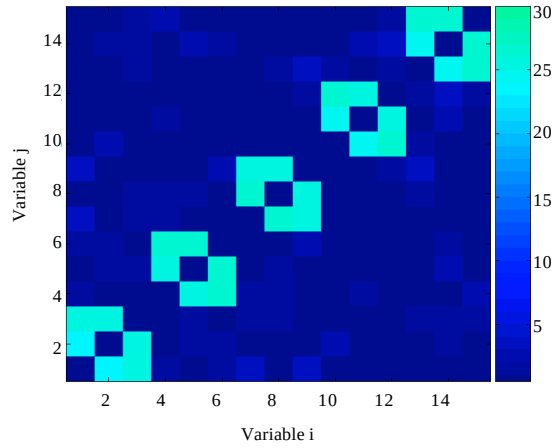

Fig. 9: Frequency matrices calculated from the models learned by $EBNA_{KS}$ for function Deceptive3 with N = 1000

of 350 at the end of running. It can be noticed that both of the two algorithms can not learn the correct structure. As in the situation of the Deceptive3 function, $EBNA_{KS}$ seems to learn more false dependencies than $EBNA_B$.

Population size which is too small may be the primary reason to explain the bad performance in the transforming the function structure into the model structure. Thus, we do the experiment again using a bigger population size with N = 1000. The experiment results obtained at the end of running are presented in Fig. 12 and 13, respectively, which are the frequency matrices calculated from the models learned by $EBNA_B$ and $EBNA_{KS}$ solving for the function Sixpeaks with n = 16.

The figures indicate that, by increasing the population size, $EBNA_{KS}$ can get a very exact structure. The model which is got by learning, receives total the short-order dependencies of the function. On the other hand, $EBNA_B$ does not accomplish a similar improvement. Moreover, the

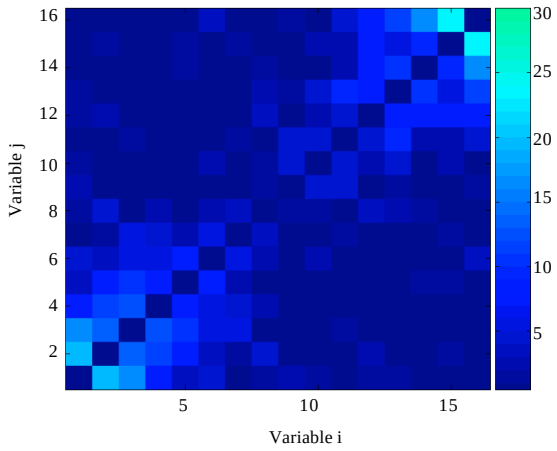

Fig. 10: Frequency matrices calculated from the models learned by $EBNA_B$ for function SixPeaks with N = 350

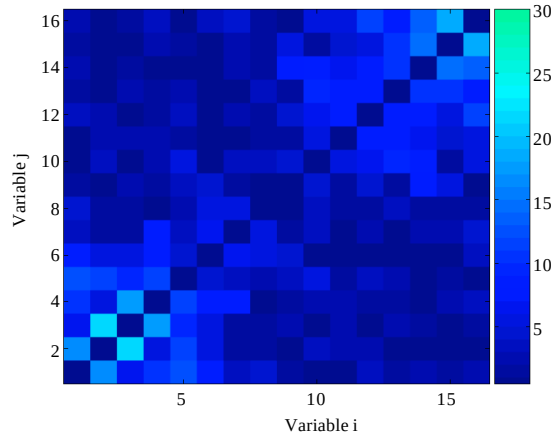

Fig. 11: Frequency matrices calculated from the models learned by $EBNA_{KS}$ for function SixPeaks with N = 350

precision of the model is lower than when we used the population size N = 350 before, which can be noticed by comparing Fig. 10 and 12. Here, we argue that the local approximate technique might be encountering certain learning limits.

In the literature (Larranaga and Lozano, 2002), the B algorithm was introduced to EDAs based on Bayesian networks, which is $EBNA_B$ in this study and the preliminary results were showed.

In this study, we investigate the connections between the true structure of problems and structure obtained from the searching for the probabilistic models by $EBNA_B$ and $EBNA_{KS}$ and it is an ongoing research direction of EDAs. According to the figures obtained from the experiments, we can investigate the most likely dependencies from the probabilistic models in EDAs and study their relationship with the true structure of functions. We can also employ the dependency displayed from the probabilistic model to establish a function which has a ideal interactions in some extends.

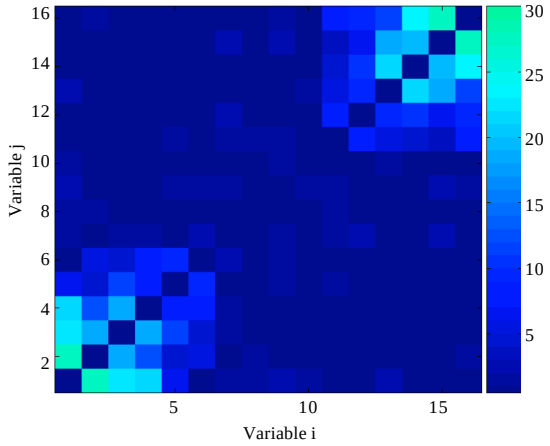

Fig. 12: Frequency matrices calculated from the models learned by $EBNA_B$ for function SixPeaks with N = 1000

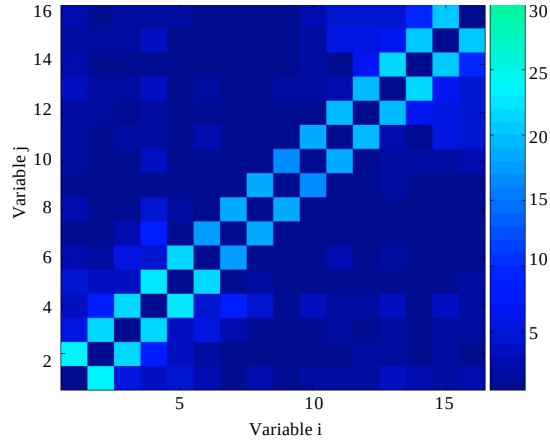

Fig. 13: Frequency matrices calculated from the models learned by $EBNA_{KS}$ for function SixPeaks with N = 1000

We argue that the approximate learning method such as B algorithm, could obtain models which only approximately display the true dependencies that appeared in the population. When the dependencies are calculated, the mistakes produced by the learning method should also be avoided.

We compare the decisive population size needed by the $EBNA_{KS}$ and $EBNA_B$ to achieve a predefined convergence rate. In the experiments, the decisive population size is the minimum population size which enable the $EBNA_{KS}$ and the $EBNA_B$ to find the optimum in 50 consecutive running. We compare the decisive population size for the function Deceptive3 (n = 9, 12, 15) and the function SixPeaks (n = 10, 12, 14, 16) obtained by $EBNA_{KS}$ and $EBNA_B$.

Table 1 gives the mean and standard deviation of the decisive population size found by $EBNA_{KS}$ and $EBNA_B$, it indicates that there is big difference between function SixPeaks and function Deceptive3. We can see that $EBNA_{KS}$ needs a less population size than $EBNA_B$. Another observation is that the standard deviation of $EBNA_B$ is always higher than that of $EBNA_{KS}$.

Table 1: Mean and standard deviation of the decisive population size found by $EBNA_{KS}$ and $EBNA_B$

		$EBNA_{KS}$		$EBNA_B$	
Functions	No.	Mean	Std	Mean	Std
SixPeaks	10	152.59	51.25	213.03	107.10
SixPeaks	12	208.91	108.10	386.11	246.18
SixPeaks	14	311.28	130.63	601.15	315.96
SixPeaks	16	859.47	107.63	Null	Null
Deceptive3	9	132.62	37.39	166.95	59.93
Deceptive3	12	167.95	59.93	261.11	86.50
Deceptive3	15	20.10	56.75	295.26	93.98

When the number of variable (n) exceed 16, the $EBNA_B$ can no longer solve the function SixPeaks, so the mean and standard deviation is Null. Since the only difference between $EBNA_{KS}$ and $EBNA_B$ is in the type of method used to learn the models, the difference of behaviors is due to the ability of $EBNA_{KS}$ to learn a more accurate model of the dependencies.

Therefore, for function SixPeaks and function Deceptive3, learning a more accurate model determines a better performance for EDAs based on Bayesian networks. It is to say that whenever the scale of the problem is easy to deal, it is a more suitable choice for theoretical analysis of the probabilistic models to use KS learning method for EDAs based on Bayesian networks.

CONCLUSION

In the comparison to the obtained structural models, we can observe that the KS method tends to learn some dependences which are not explicitly described in the formulation of the function than the B method, particularly when we use the population size which is small. It shows that the structures got by KS learning could describe the random errors of the samples as well as the underlying relationships among the problem variables. However, the effect of this phenomenon is determined by the characteristics of the problem. Although, the phenomenon of over fitting is studied in the field of EDAs, we argue that we should do more work in order to understand this issue. In particular, when a KS structural learning is carried out, the over fitting which takes place, could be a research subject in the future.

The experiment results also indicate that the kind of learning method might produce substantial differences in the probabilistic models learned and in the behavior of the EDAs based on Bayesian networks. Specially, when the KS learning method is provided with sufficient information that the selected population contains, it could learn an exact structural model which is close to the true structure of the problem. Nevertheless, the approximate method could not obtain exact structures at all times even when a big population size is employed. This fact indicates the existence of certain learning limits involved to the approximate method. This problem should be investigated more in-depth in the future.

ACKNOWLEDGMENTS

This study is supported in part by National Natural Science Foundation of China (Grant No. 60975050) and Fundamental Research Funds for the Central Universities (Grant No. 6081014).

REFERENCES

Abdollahzadeh, A., A. Reynolds, M. Christie, D.W. Corne, B.J. Davies and G.J. Williams, 2012. Bayesian optimization algorithm applied to uncertainty quantification. SPE J., 17: 865-873.

Ahn, C.W., J. An and J.C. Yoo, 2012. Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs. Inform. Sci., 192: 109-119.

Armananzas, R., Y. Saeys, I. Inza, M. Garcia-Torres, C. Bielza, Y. Van de Peer and P. Larranaga, 2011. Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms. IEEE/ACM Trans. Comput. Biol. Bioinform., 8: 760-774.

Barber, D., 2012. Bayesian Reasoning and Machine Learning. Cambridge University Press, Cambridge, MA., USA., ISBN-13: 9780521518147, Pages: 697.

Bensi, M., A.D. Kiureghian and D. Straub, 2013. Efficient Bayesian network modeling of systems. Reliab. Eng. Syst. Saf., 112: 200-213.

Blanco, R., I. Inza and P. Larranaga, 2003. Learning Bayesian networks in the space of structures by estimation of distribution algorithms. Int. J. Intell. Syst., 18: 205-220.

Bonawitz, E., S. Denison, T.L. Griffiths and A. Gopnik, 2014. Probabilistic models, learning algorithms and response variability: Sampling in cognitive development. Trends Cognit. Sci., 18: 497-500.

Boussaid, I., J. Lepagnot and P. Siarry, 2013. A survey on optimization metaheuristics. Inform. Sci., 237: 82-117.

Buntine, W., 1991. Theory refinement on Bayesian networks. Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence, July 13-15, 1991, Los Angeles, CA., USA., pp: 52-60.

Ceberio, J., E. Irurozki, A. Mendiburu and J.A. Lozano, 2014. A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. IEEE Trans. Evol. Comput., 18: 286-300.

Chang, P.C. and M.H. Chen, 2014. A block based estimation of distribution algorithm using bivariate model for scheduling problems. Soft Comput., 18: 1177-1188.

Chen, T., K. Tang, G. Chen and X. Yao, 2010. Analysis of computational time of simple estimation of distribution algorithms. IEEE Trans. Evol. Comput., 14: 1-22.

Daly, R., Q. Shen and S. Aitken, 2011. Learning Bayesian networks: Approaches and issues. Knowledge Eng. Rev., 26: 99-157.

Darwiche, A., 2010. Bayesian networks. Commun. ACM, 53: 80-90.

Etxeberria, R. and P. Larranaga, 1999. Global optimization using Bayesian networks. Proceedings of the 2nd Symposium on Artificial Intelligence, March 1999, Habana, Cuba, pp: 332-339.

Fan, J., Y. Liang, Q. Xu, R. Jia and Z. Cui, 2011. EDA-USL: Unsupervised clustering algorithm based on estimation of distribution algorithm. Int. J. Wireless Mobile Comput., 5: 88-97.

Friedman, N., D. Geiger and M. Goldszmidt, 1997. Bayesian network classifiers. Mach. Learn., 29: 131-163.

Griffiths, T.L., N. Chater, C. Kemp, A. Perfors and J.B. Tenenbaum, 2010. Probabilistic models of cognition: Exploring representations and inductive biases. Trends Cognit. Sci., 14: 357-364.

Hauschild, M., M. Pelikan, K. Sastry and C. Lima, 2009. Analyzing probabilistic models in hierarchical BOA. IEEE Trans. Evol. Comput., 13: 1199-1217.

Hauschild, M. and M. Pelikan, 2011. An introduction and survey of estimation of distribution algorithms. Swarm Evol. Comput., 1: 111-128.

Hauschild, M.W., M. Pelikan, K. Sastry and D.E. Goldberg, 2012. Using previous models to bias structural learning in the hierarchical BOA. Evol. Comput., 20: 135-160.

Jin, C. and S.W. Jin, 2014. Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms. Applied Soft Comput., 15: 113-120.

Koivisto, M. and K. Sood, 2004. Exact Bayesian structure discovery in Bayesian networks. J. Mach. Learn. Res., 5: 549-573.

Koller, D. and N. Friedman, 2009. Probabilistic Graphical Models: Principles and Techniques. 1st Edn., The MIT Press, New York, USA., ISBN-13: 9780262013192, Pages: 1280.

Larranaga, P. and J.A. Lozano, 2002. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluewer Academic Publishers, Boston, MA., USA., ISBN-13: 978-0792374664, Pages: 382.

Li, X., S. Mabu and K. Hirasawa, 2014. A novel graph-based estimation of the distribution algorithm and its extension using reinforcement learning. IEEE Trans. Evol. Comput., 18: 98-113.

Lima, C.F., M. Pelikan, D.E. Goldberg, F.G. Lobo, K. Sastry and M. Hauschild, 2007. Influence of selection and replacement strategies on linkage learning in BOA. Proceedings of the IEEE Congress on Evolutionary Computation, September 25-28, 2007, Singapore, pp: 1083-1090.

Martins, J.P. and A.C. Delbem, 2014. Efficiency enhancement of estimation of distribution algorithms by a compressed tournament selection. J. Intell. Fuzzy Syst., 26: 2537-2545.

Muelas, S., A. Mendiburu, A. LaTorre and J.M. Pena, 2014. Distributed estimation of distribution algorithms for continuous optimization: How does the exchanged information influence their behavior? Inform. Sci., 268: 231-254.

Reeves, C.R., 2010. Genetic Algorithms. In: Handbook of Metaheuristics, Gendreau, M. and J.Y. Potvin (Eds.). Springer, New York, USA., ISBN-13: 9781441916631, pp: 109-139.

Shim, V.A., K.C. Tan and C.Y. Cheong, 2012. A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. IEEE Trans. Syst. Man Cybern. C: Applic. Rev., 42: 682-691.

Wang, F.S. and L.H. Chen, 2013. Genetic Algorithms. In: Encyclopedia of Systems Biology, Dubitzky, W., O. Wolkenhauer, K.H. Cho and H. Yokota (Eds.). Springer, New York, USA., ISBN-13: 9781441998620, pp: 819-820.