INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 11(6):889-901, December 2016.

Learning Bayesian Networks in the Space of Structures by a Hybrid Optimization Algorithm

M. Zhu, S. Liu, J. Jiang

Mingmin Zhu*, Sanyang Liu
School of Mathematics and Statistics, Xidian University
2 South Taibai Road, Xi'an, China, 710071
*Corresponding author: zmmzhu2010@126.com
liusanyang@126.com

Jiewei Jiang
School of Computer Science and Technology, Xidian University
2 South Taibai Road, Xi'an, China, 710071
jiangjw924@126.com

Abstract: Bayesian networks (BNs) are one of the most widely used class for machine learning and decision making tasks especially in uncertain domains. However, learning BN structure from data is a typical NP-hard problem. In this paper, we present a novel hybrid algorithm for BN structure learning, called MMABC. It's based on a recently introduced meta-heuristic, which has been successfully applied to solve a variety of optimization problems: Artificial Bee Colony (ABC). MMABC algorithm consists of three phases: (i) obtain an initial undirected graph by the subroutine MMPC. (ii) Generate the initial population of solutions based on the undirected graph and (iii) perform the ABC algorithm to orient the edges. We describe all the elements necessary to tackle our learning problem, and experimentally compare the performance of our algorithm with two state-of-the-art algorithms reported in the literature. Computational results demonstrate that our algorithm achieves better performance than other two related algorithms.
Keywords: Bayesian network, Artificial Bee Colony, Structural learning, Meta-heuristics, Scoring function.

1 Introduction

Bayesian networks (BNs), also known as belief networks, are becoming a popular tool for representing uncertainty in artificial intelligence. They have been applied to a wide range of tasks such as natural spoken dialog systems, vision recognition, expert systems, medical diagnosis, and genetic regulatory network inference [1-6]. With a network at hand, probabilistic inference can be performed to predict the outcome of some variables based on the observations of others [7,8]. There has been a lot of work in the last ten years on the learning of BNs for both graph structure and probability parameters. However, learning the structure is harder and, arguably, more critical. Most of these algorithms can be grouped into two different categories: constraint-based methods [9,10,11] and search-and-score methods [12,13,14]. The first one poses the learning process as a constraint satisfaction problem, and then constructs a network structure by testing the conditional independence relations. The second one poses the learning problem as a structure optimization problem. Namely, it uses a score metric to evaluate every candidate network structure, and then, finds a network structure with the best score. Though the implement of the former approach is relatively simple, the computation for high-order conditional independence tests is complex and irresponsible. Thus, the score-and-search approach gradually becomes a popular approach for learning BNs [15,16,17].

Recently, a hybrid algorithm that combines the conditional independence methodology with scoring metric optimization, max-min hill climbing (MMHC)[18], has been introduced and proven to be very competitive. In MMHC approach, after reconstructing an undirected graph encoding parent-child relationships by using the max-min parents and children (MMPC) algorithm, a greedy Bayesian scoring hill-climbing search is used to orientate the edges.

In this paper, the idea is further explored, by using the artificial bee colony (ABC)[19,20] algorithm to perform the edge orientation. We call the new hybrid algorithm max-min artificial bee colony (MMABC) algorithm. The ABC algorithm can be used to search for the near-optimal graph in the space of directed acyclic graphs (DAGs). We describe all the elements necessary to tackle our learning problem using this meta-heuristic, and experimentally compare the performance of our ABC-based algorithm with other algorithms used in the literature. The remainder of this paper is organized as follows. Section 2 describes the concepts and methods related to BNs. We introduce the ABC algorithm and our main algorithm in Section 3. Simulation studies are conducted to demonstrate the performance of our algorithm and existing algorithms in Section 4. Finally, in Section 5, we conclude and outline our future work.

2 Preliminaries

In this section we briefly review some basic concepts related to BNs and how to learn them.

A BN is a graphical representation of a joint probability distribution that includes two components. One is a directed acyclic graph (DAG) $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, the set of nodes, represents the random variables, and E is the set of edges, represents direct dependency relationships between variables. The other is a conditional probability table that quantifies the effects ($p(v_i|pa(v_i))$) that the parent set of v_i have on the variable v_i in G, where $pa(v_i)$ is the parent set of v_i in G, i.e., $pa(v_i) = \{v_j|v_j \to v_i \in E, v_j \in V \backslash v_i\}$. If v_j is the parent of v_i, then v_i is the child of v_j in G. Learning the structure of BN is equivalent to identifying the structure of the DAG G that best matches the data set D. Robinson [21] showed that the number $f(n)$ of possible structures for BN having n nodes is given by the recurrence formula:

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} \frac{n!}{i!(n-i)!} 2^{i(n-1)} f(n-i). \tag{1}$$

Searching for the best structure is difficult because the search space increase exponentially with the number of variables. It has been proven to be an NP-hard problem [22].

Recently, a new and quite competitive method, the MMHC algorithm [18], that combines both scoring-based and conditional independence test approaches has been introduced. It first learns the undirected graph (i.e., the edges without their orientation) of a BN using a local discovery algorithm called the max-min parents and children (MMPC) algorithm. Then, it orients the undirected graph using a greedy hill climbing search. The subroutine MMPC is sound in the sample limit. It attempts to quickly identify the set of parents and children $pc(v_i)$ of a variable v_i in two phases. In the first phase, variables that are conditionally independent on v_i can enter the set of candidate parents and children according to a heuristic function. In each iteration, variable v_j enters the candidate set that maximizes the minimum association to v_i given the current candidate set. In the second phase, the false-positive variables that possibly entered the candidate set in the first phase are removed. More detailed explanation can be found in [18].

In this paper, we first use the MMPC algorithm to learn the undirected graph of a BN, and then orientate the edges by applying the artificial bee colony (ABC) algorithm. In the following, we briefly introduce the ABC algorithm, which is a new meta-heuristic approach inspired by the intelligent foraging behavior of honeybee swarm. After that, we give the main algorithm of this paper.

3 Learning Bayesian networks using the ABC algorithm

3.1 The Artificial Bee Colony algorithm

The ABC [20] algorithm is a swarm based meta-heuristic that simulates foraging behavior of honey bees. This algorithm includes three kinds of bees considering the division of labor: employed bees, onlooker bees and scout bees. A bee that is currently exploiting a food source is called an employed bee. A bee waiting in the hive for making decision to choose a food source is named as an onlooker. A bee carrying out a random search for a new food source is called a scout. An employed bee keeps a food source in her mind when she leaves from the hive and she shares the information about her food source with onlookers on dance area. Onlookers select a food source by watching the dances of the employed bees and try to improve this source. If a food source is abandoned, its employed bee becomes a scout to explore new food sources randomly. Scouts either randomly search the environment in order to find a new food source depending on an internal motivation or based on possible external clues.

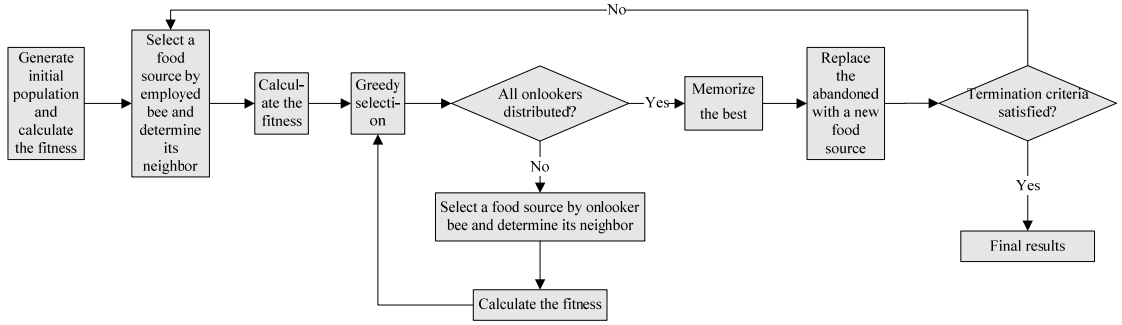

Figure 1: Flowchart of the ABC algorithm

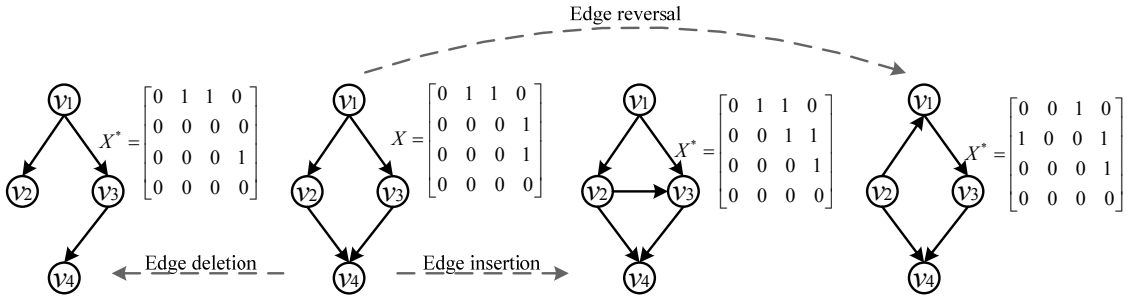

Figure 2: Three edge operators used in the MMABC algorithm

In ABC algorithm each food source represents a possible solution for the problem and the fitness value of the solution corresponds to the nectar amount of this food source. Each food source is exploited by only one employed bee. In other words, the number of employed bees is equal to the number of candidate solutions. Similar to the other swarm intelligence based approaches, the ABC algorithm is also an iterative process. It repeats a series of steps until a termination criterion is met. We show all these steps and interactions between them as a flowchart on Figure 1.

3.2 A hybrid algorithm for learning BNs using ABC optimization

The above basic ABC algorithm, originally designed for the continuous nature of optimization problems, can not directly be used for discrete problems. In this section, we develop a new ABC-based algorithm for learning BNs that performs the search in the space of directed acyclic graphs. The new algorithm is called MMABC for short. We first use the MMPC algorithm to learn the undirected graph of a BN. Based on this undirected graph UG, the initial population for MMABC is produced randomly. More specifically, the edges of the initial graph corresponding to each individual are randomly selected from the edge set of UG, which is the candidate edge set of the optimal BN structure. This greatly reduces the randomness of the initial population and improve the convergence speed. The main components of our MMABC algorithm are described below.

a. Initialization.

Let m and n denote the total number of food sources and the number of variables in a BN, respectively. Since the structure of a BN is a directed graph $G = (V, E)$, we represent each individual by a binary node-node adjacency matrix $X = (x_{ij})_{n\times n}, i, j = 1, 2, \ldots, n$. Entry (i, j) is 1 if there is a directed arc from node i to node j, and 0 otherwise. That is,

$$x_{ij} = \begin{cases} 1, & \text{if } V_i \to V_j \in E, \\ 0, & \text{if } V_i \to V_j \notin E. \end{cases} \tag{2}$$

Initial food sources (solutions) $X_k = (x_{ij})_{n\times n}, k = 1, 2, \ldots, m$ are generated randomly by selecting edges from an undirected graph $UG = (V, \overline{E})$, which is obtained by MMPC. Counters which store the numbers of trials of solutions are reset to 0 in this phase.

b. Employed Bees Phase.

Each employed bee is associated with only one food source site. An employed bee generates a neighboring solution by three operators: edge insertion, edge deletion and edge reversal. We show these three operators in Figure 2. Let X_k and ${X_k}^*$ denote the current solution and the generated one, respectively. The quality of the graph G corresponding to ${X_k}^*$ is measured by the K2 score function [23] as follows:

$$f_{K2}(G, D) = \sum_{i=1}^{n} f_{K2}(v_i, pa(v_i)|D_{v_i,pa(v_i)}), \tag{3}$$

$$f_{K2}(v_i, pa(v_i)|D_{v_i,pa(v_i)}) = \sum_{j=1}^{q_i} \left[\log\left(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \right) + \sum_{k=1}^{r_i} \log(N_{ijk}!) \right], \tag{4}$$

where D is the domain data set. $D_{v_i,pa(v_i)}$ are the statistics of the variable v_i and $pa(v_i)$ in D, i.e., the number of instances in D that match each possible instantiation of v_i and $pa(v_i)$. N_{ijk} is the number of cases in D where v_i is in its kth state and its parents are in their jth state. $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ denotes the number of cases in D where $pa(v_i)$ is in its jth state and r_i denotes the number of states of variable v_i, q_i is the number of parent configurations of v_i. Because the K2 metric is decomposable and the graphs corresponding to ${X_k}^*$ and X_k are different from one edge by each one of three operators, we can reuse most of the computations made in the food source X_k. In other words, we only need to update the statistics corresponding to the variables whose parent sets have been modified. Thus, the time performance can be greatly enhanced.

As our problem is a maximization problem and the value of $f_{K2}(\cdot)$ is negative, therefore, the fitness value assigned to the solution ${X_k}^*$ is determined using the following expression:

$$fit_k = \frac{1}{1 + abs(f_k)}, \tag{5}$$

where f_k is the K2 value of the graph G corresponding to ${X_k}^*$. A greedy selection is employed between X_k and ${X_k}^*$ such that the better one is selected depending on fitness values representing the nectar amount of the food sources at X_k and ${X_k}^*$. If X_k cannot be improved, its counter is incremented by 1, otherwise, the counter is reset to 0.

c. Onlooker Bees Phase.

An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source site with a probability related to its nectar amount. The roulette wheel selection scheme is used here in which each slice is proportional in size to the fitness value as follows:

$$p_k = \frac{fit_k}{\sum_{j=1}^{m} fit_j}. \tag{6}$$

By doing so, the onlooker bees are encouraged to visit the food sources that have higher nectar amounts. Briefly, food source selection by onlookers is based on the information provided by employed bees. A uniform random number within the range $[0, 1]$ is obtained for each source. If the probability value p_k is greater than this random number, then the onlooker bee generates a neighboring food source again by using three operators as in the case of the employed bee. After the source is evaluated, greedy selection is applied. If solution X_k cannot be improved, its counter is incremented by 1, otherwise, the counter is reset to 0. This process is repeated until all onlookers are distributed onto food source sites.

d. Scout Bees Phase.

If the value of the counter is greater than the control parameter "limit", then the food source associated with this counter is abandoned. Assume that the abandoned source is X_k, then the scout randomly discovers a new food source ${X_k}^*$ which can be achieved by randomly generating any directed acyclic graph. This procedure can increase solutions diversity and avoid falling into the local optimum. In our algorithm, there is only a single food source which can be abandoned in each cycle, and only one employed bee can be a scout.

e. Termination.

If a termination is not satisfied, go to Employed Bees Phase; otherwise stop the procedure and output the best food source found so far.

As shown in Algorithm 1, the MMABC algorithm first constructs the undirected graph UG of the underlying BN structure by the MMPC algorithm, which obtained the parents and children of each variable. It gives the best-so-far undirected graph, from which the initial population can be generated randomly by selecting $n - 1$ edges from UG (assumed the underlying BN has n variables). Figure 3 shows the process of generating initial population by a simple example. Consider the second graph in Figure 2. Assumed that its undirected graph UG obtained by MMPC is shown in Figure 3 (a). By randomly selecting 3 directed edges from UG, we can get an initial solution. Figure 3(b), Figure 3(c) and Figure 3(d) show 3 solutions of the initial population generated from Figure 3(a).

After producing and evaluating the initial population, the main loop of the MMABC algorithm then begins for $Iter$ iterations. Step 7 and 15 show one of the edge operators can be chosen to obtain the neighbor of the current solution in Employed Bees Phase and Onlooker Bees Phase, respectively. Each of the operators can be chosen randomly with equal probability. If there is a cycle in the generated graph, another new graph can be achieved again.

4 Experimental results

In this section, we present the experimental results carried out with our algorithm and compare MMABC with two hybrid heuristic methods: MMHC [18] and MMACO [24]. In all the

Algorithm 1 The MMABC algorithm.

1. Obtain the undirected graph UG by MMPC;
2. Generate the initial population of solutions $X_k, k = 1,2,\ldots,m$, $tr_k = 0$. m is the total number of food sources. tr_k is the non-improvement number of the solution X_k, used for abandonment;
3. Evaluate the population;
4. Iter=1;
5. **repeat**
6. **for** $k = 1$ to m **do**
7. Produce a new food source $X_k{}^*$ for the employed bee of the food source X_k by using one of the three edge operators and evaluate its quality according to equation (3);
8. Apply a greedy selection process between $X_k{}^*$ and X_k and select the better one;
9. If solution X_k does not improve $tr_k = tr_k + 1$, otherwise $tr_k = 0$;
10. **end for**
11. Calculate the probability value p_k by (6), $k = 1,2,\ldots,m$;
12. $t = 0, k = 1$;
13. **repeat**
14. **if** $random < p_k$ **then**
15. Produce a new food source $X_k{}^*$ for the onlooker bee of the food source X_k by using one of the three edge operators and evaluate its quality by (3);
16. Apply a greedy selection process between $X_k{}^*$ and X_k and select the better one;
17. If solution X_k does not improve $tr_k = tr_k + 1$, otherwise $tr_k = 0$;
18. $t = t + 1$;
19. **end if**
20. **until** $t = m$;
21. **if** $max(tr_k) > limit$ **then**
22. Replace X_k with a new randomly produced solution;
23. **end if**
24. Memorize the best solution achieved so far;
25. Iter=Iter+1;
26. **until** Iter=Maximum Iteration.

Figure 3: The MMABC algorithm

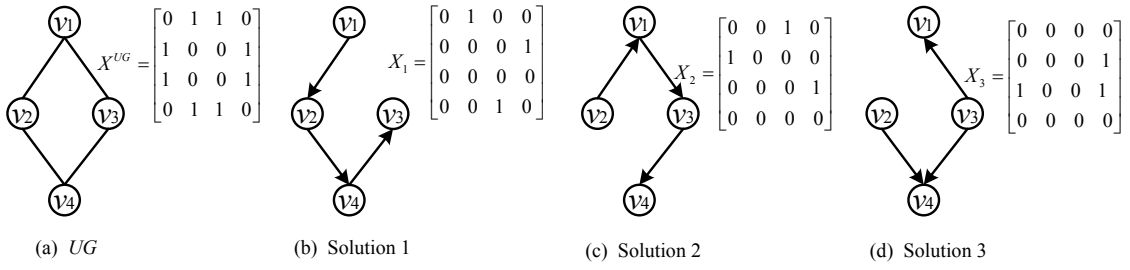

Figure 4: The process of generating initial population

cases the scoring metric used to guide the search is the K2 metric, so, the difference among implementations is minimized. All these experiments are based on the BN toolbox written by Murphy [25] and the Causal Explorer System developed by Aliferis et al.[26]. The experimental platform is a personal computer with Pentium 4, 3.06 GHz CPU, 1.0 GB memory, and Windows XP.

The standard way of assessing the accuracy of a learning method is to draw samples from a known BN, apply structure learning on the artificial data, and to compare the learned structure with the original one. To benchmark our method, we investigate its performance on artificial data with real-world characteristics focusing on the well-known Asia [27], Insurance [28], and Alarm [29] networks. The first one is a small BN using a fictitious medical example that considers the dependence of the health status of a patient (having tuberculosis, lung cancer, or bronchitis) on risk factors (visit to Asia and smoking) and clinical observations (Dyspnea and X-ray). It consists of eight nodes with eight edges connecting them. Each random variable is discrete in nature and can take two states. The second example is a network for estimating the expected claim costs for a car insurance policyholder. It consists of 27 nodes and 52 edges. The last one is a medical diagnostic system for patient monitoring. It consists of 37 nodes and 46 edges connecting them. The random variables in the ALARM network are discrete in nature and can take two, three, or four states. The characteristics of the networks are summarized in Table 1. For these three networks, data sets with 200, 500, 1000, 3000, and 5000 samples are drawn. For each of the sample sizes, ten independent data sets are generated by applying Gibbs sampling [30] on the benchmark networks, resulting in a set of 50 data sets for each network. 10 replicates are done for each of different network parameters and sample size. We will discuss our experimental results in more detail in the following two subsections.

Table 1: Bayesian networks

Network	Variables	Edges	Max In/Out-degree	Domain range
Asia	8	8	2/2	2
Insurance	27	52	3/7	2-5
Alarm	37	46	4/5	2-4

Table 2: The K2 values for the true networks with different samples

Network	$n = 200$	$n = 500$	$n = 1000$	$n = 3000$	$n = 5000$
Asia	-501.79	-1194.9	-2350.1	-6881.9	-11458
Insurance	-5291.2	-9851.6	-17105	-45406	-73717
Alarm	-3259.2	-6364.4	-11162	-29894	-48597

4.1 Comparison of the solution quality

In this subsection, we compare the performance of the different algorithms on all of the data sets. Our main objective in these experiments is to determine whether MMABC is more efficient and effective than the other two hybrid algorithms. For MMABC and MMACO, the population size is set to 20, and the maximum number of iterations is set to 100. The significant level used

in these algorithms is set to 0.05. Other parameter settings of the MMACO algorithm can be found in [24]. The parameter "limit" for MMABC is set to 6, namely, if the solution associated with an employed bee does not improve for 6 times, then that solution is replaced by some randomly generated solution. In order to provide an objective measure of network structure reconstruction behavior, we compare the underlying DAG with the learned DAG and record the structural hamming distance (SHD), where SHD is defined as the total number of operations to verify the learned DAG to the underlying DAG. The K2 scoring function was also presented to measure the learned network structure, which could be used to compare the quality of the structures obtained by different algorithms. The K2 values for the true network structures with different data sets can be seen in Table 2.

Table 3: The results for three algorithms with different datasets

Network	Algorithm	$n = 200$	$n = 500$	$n = 1000$	$n = 3000$	$n = 5000$
	MMABC	3.0	2.1	1.0	0.7	0.3
		-500.53±0.7	-1191.6±1.1	-2351.6±3.1	-6877.0±3.5	-11454.0±5.2
Asia	MMACO	4.8	2.9	1.2	0.6	0.3
		-488.8±1.3	-1179.3±3.2	-2346.5±7.1	-6868.0±9.1	-11478.0±11.9
	MMHC	6.1	3.3	2.0	1.1	0.9
		-507.6±1.1	-1209.2±1.8	-2351.9±3.7	-6896.3±5.7	-11593.0±6.8
	MMABC	15.8	11.6	8.7	6.6	4.1
		-5292.1±1.2	-9853.8±1.8	-17116.1±2.6	-45408.3±4.7	-73716.9±5.6
Insurance	MMACO	16.9	12.4	9.4	6.7	5.3
		-5295.8±3.1	-9857.6±5.6	-17115.3±6.9	-45412.1±8.2	-73716.7±10.1
	MMHC	36.2	27.9	19.3	12.4	8.9
		-5361.3±1.5	-9911.5±3.2	-17190.4±3.4	-45815.6±5.3	-73850.3±6.8
	MMABC	18.4	11.3	10.0	6.9	3.9
		-3198.3±1.2	-6363.5±1.7	-11149.9±2.8	-29883.9±4.1	-48600.9±5.0
Alarm	MMACO	19.1	12.1	11.2	7.2	4.4
		-3072.2±3.1	-6344.3±5.5	-11085.7±6.9	-29891.1±9.2	-48605.9±11.1
	MMHC	35.8	27.6	19.2	13.8	9.6
		-3085.8±1.4	-6435.1±3.1	-11210.2±3.3	-29849.3±5.3	-48694.0±6.7

Table 3 provides the mean and standard deviation of the K2 score and the mean SHD values obtained for the learned networks with different data sets. $\mu \pm \sigma$ indicates the mean and the standard deviation over the executions carried out. We have carried out 10 executions of each algorithm and for each domain considered. In Table 3, the first six rows display the results of three algorithms on Asia data sets, and the rest twelve rows is the results on Insurance and Alarm data sets. For example, figures in the second row are the SHD values obtained by the MMABC algorithm on Asia data sets. Figures in the third row are their corresponding mean and standard deviation of the K2 scores. On the whole, we can see that MMACO algorithm is a little better than MMHC algorithm on the Asia network, and our algorithm is best. However, in the case of Insurance and Alarm networks, the advantages of MMABC and MMACO are more evident, and MMABC performs better than MMACO, which means our algorithm has better performance with more complicated networks. The K2 score of our algorithm is always closer

to that of the original network. The reason is that the construction process of a BN for an ant is dependent much too on the pheromone and heuristic information of candidate arcs, which is prone to overfitting. Although it can be seen that the K2 score of MMABC and MMACO becomes similar as the sample size increase, MMABC has a lower SHD value. We also note that in some cases, the score obtained by MMACO is better than that of MMABC, but MMABC has lower SHD values. This indicates that a better score does not necessarily mean a better SHD value and vice-versa. It can occur because of small size and because of the parameters given to the scoring function, which have been shown to produce differences in scoring function behavior. It is worth mentioning that the standard deviation of the MMABC is lower than that of the other two algorithms, which means that the MMABC guarantees good quality networks with less fluctuation than other algorithms.

4.2 Comparison of the convergence and time performance

In order to measure the convergence speed and the runtime complexity to learn the structure, we compared with the typical runs of the MMABC and MMACO algorithms on the same data set by large number of experiments.

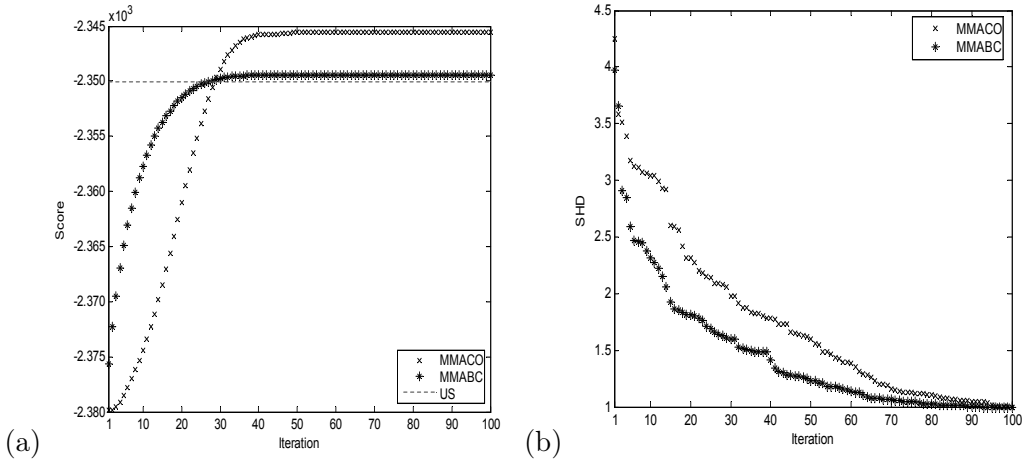

Figure 5: Comparison of the solution quality for MMACO and MMABC on Asia data sets: (a) Scores for Asia; (b) SHD for Asia

Figure 4 and Figure 5 show the curves of the convergence performance on two data sets, Asia-1000 and Alarm-5000, respectively. In these figures, the abscissas depict the number of iterations, and the Y-coordinates depict the K2 score or SHD value of the learned graph. US specifies the underlying score of the original network. From a general point of view, MMABC achieve better quality networks in the initial stages of the optimization and has a rapidly converging learning curve, which means that MMABC require a significantly lower number of iterations than MMACO to achieve networks of better quality. Although we are testing on two very different data sets, we obtain similar observation that as the score is improving over iterations, the SHD value is deteriorating. It provides strong evidence that MMABC is performing well against the MMACO algorithm.

In Figure 6, we test MMABC with 4 different population sizes on Asia-1000: 10 (MMABC1), 20 (MMABC2), 30 (MM ABC3), 40 (MMABC4). From the curves in Figure 6(a) and Figure 6(b), it can be found that the population size m can have an effect on the convergence speed of the algorithm. In other words, MMABC takes a comparatively large number of iterations to converge to an acceptable solution with a small number of bees. Because a large population

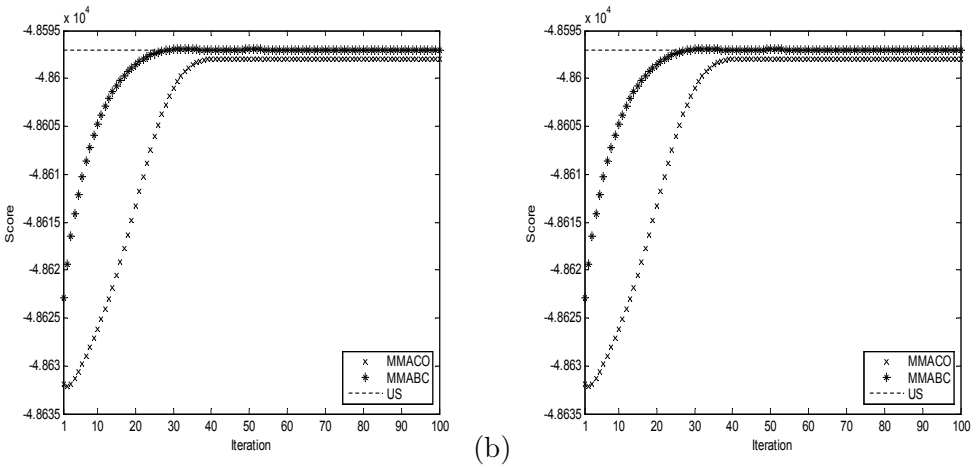

Figure 6: Comparison of the solution quality for MMACO and MMABC on Alarm data sets: (a) Scores for Alarm; (b) SHD for Alarm

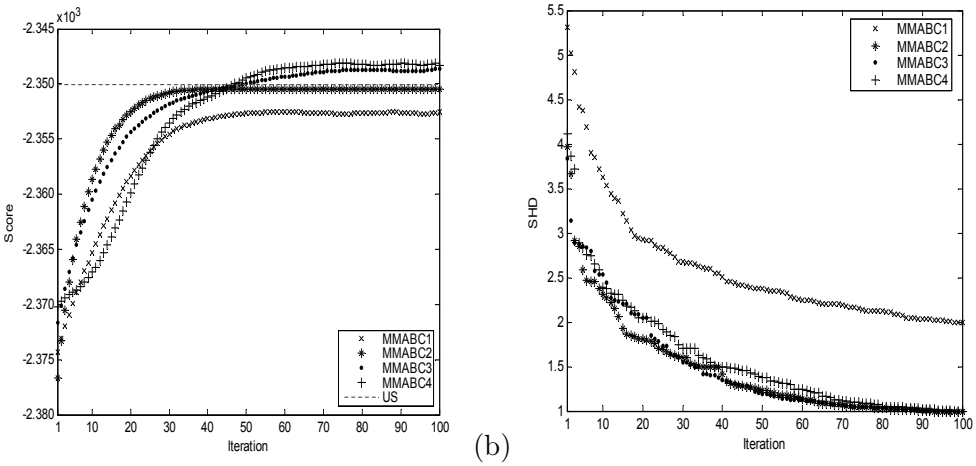

Figure 7: Comparison of the solution quality for different population sizes on Asia data sets: (a) Scores for Asia; (b) SHD for Asia

means that more food sources are employed, there are more different solutions examined than in a smaller population. On the other hand, the time consumed also increased proportionally with increasing population size. It should be noted that tiny changes to the score value can lead to large structural changes as an algorithm converges towards the optimum network.

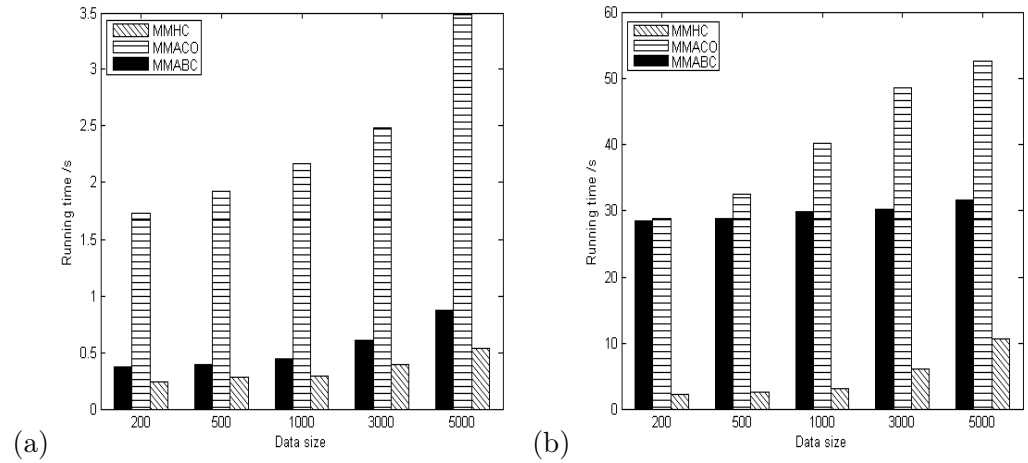

Figure 8: Comparison of the time performance on two networks: (a) Running time for Asia data sets; (b) Running time for Alarm data sets

Figure 7 gives the average running time to reach the best solution for three algorithms on two networks. Based on Figure 7, we can see that the proposed MMABC has a considerable better time performance than MMACO, and MMHC has the best time performance. However, from the results of Table 3, we can see that MMHC obtains worse K2 score and SHD values. Moreover, the advantage is very obvious when the data set is large on the same network, namely, the bigger the sample size, the more obvious the improvement. We also noted that the running time of MMABC is affected less by the size of data but more by the problem dimension. It means that MMABC is not sensitive to the increase of the sample capacity and able to handle very large data sets.

Conclusions

In this paper we have designed and implemented a hybrid ABC-based algorithm for the BN structure learning problem. Ideas from constraint based local learning and score-based ABC were combined into effective learning processes. Its application to several benchmark networks yielded better results than the previously used MMHC and MMACO algorithms.

This paper demonstrates the capability of ABC algorithm in learning a probability graphical model. Ideas presented in this paper can be applied to many other probability graphical problems also, such as dynamic BN, chain graph, etc. Moreover, we have applied the basic three edge operators for determining the neighboring solutions in MMABC. It will be interesting to see whether there exists some alternative approach to serve the same purpose here. We also intend to expand this analysis to larger benchmark networks and make further conclusions on the usefulness of the constraint-based and score-based learning approaches. Finally, although we assume in this paper that the data are completely observed, missing data or data with latent variables may arise in practice. Generalization of the proposed algorithm to incomplete data is of great research interest.

Acknowledgment

The authors would like to thank the editor and the anonymous reviewers for their insightful comments and suggestions. This work has been partially supported by the National Natural Science Foundation of China (Grant No. 61373174, 11401454), the Natural Science Foundation of Shannxi Province, China (Grant No. 2014JQ1031) and the Fundamental Research Funds for the Central Universities (Grant No. JB140711).

Bibliography

[1] Z. Cai, S. Sun, S. Si, B. Yannou, Identifying product failure rate based on a conditional Bayesian network classifier, *Expert Systems with Applications*, 38(5): 5036-5043.

[2] Y. Sun, Y.Y. Tang, S.X. Ding, S.P. Lv, Y.F. Cui, Diagnose the mild cognitive impairment by constructing Bayesian network with missing data,*Expert Systems with Applications*, 38(1): 442-449.

[3] V. Aquaroa, M. Bardoscia, R. Bellotti, A. Consiglio, F.D. Carlo, G. Ferri, A Bayesian Networks approach to Operational Risk, *Physica A*, 389(8): 1721-1728.

[4] D.C. Kim, X. Wang, C.R. Yang, J. Gao, Learning biological network using mutual information and conditional independence, *BMC Bioinformatics*, 11(3): S9.

[5] S. Nikolajewa, R. Pudimat, M. Hiller, M. Platzer, R. Backofen, BioBayesNet: a web server for feature extraction and Bayesian network modeling of biological sequence data, *Nucleic Acids Research*, 35: 688-693.

[6] R. Wei, J. Zhen, L. Bao, Study on mining big users data in the development of HuBei auto-parts enterprise, *Mathematical modelling of engineering problems*, 2(4): 1-6.

[7] K. Burns, Bayesian inference in disputed authorship: a case study of cognitive errors and a new system for decision support, *Information Science*, 176(11): 1570-1589.

[8] S. Lee, Y. Son, J. Jin, Decision fild theory extentions for behavior modelling in dynamic environment using Bayesian belief network, *Information Science*, 178(10) 2297-2314.

[9] J. Cheng, R. Greiner, J. Kelly, D. Bell, W. Liu, Learning Bayesian networks from data: an information theory based approach, *Artificial Intelligence*, 137: 43-90.

[10] J.P. Pellet, A. Elisseef, Using Markov Blankets for Causal Structure learning, *Journal of Machine Learning Research*, 9: 1295-1342.

[11] C. Borgelt, A conditional independence algorithm for learning undirected graphical models, *Journal of Computer and System Sciences*, 76: 21-33.

[12] E. Perrier, S. Imoto, S. Miyano, Finding Optimal Bayesian Network Given a Super-Structure, *Journal of Machine Learning Research*, 9: 2251-2286.

[13] L.M. de Campos, A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests, *Journal of Machine Learning Research*, 7: 2149-2187.

[14] L.M. de Campos, J.M. Fernández-Luna, J.A. Gámez, J.M. Puerta, Ant colony opyimization for learning Bayesian networks, *International Journal of Approximate Reasoning*, 31: 291-311.

[15] R. Daly, Q. Shen, Learning Bayesian Network Equivalence Classes with Ant Conoly Optimization, *Journal of Artificial Intelligence Research*, 35: 391-447.

[16] L. Bouchaala, A. Masmoudi, F. Gargouri, A. Rebai,Improving algorithms for structure learning in Bayesian Networks using a new implicit score, *Expert Systems with Applications*, 37: 5470-5475.

[17] J. Ji, H. Wei, C. Liu, An artificial bee colony algorithm for learning Bayesian networks, *Soft Computing*, 17: 983¨C994.

[18] I. Tsamardinos, L.F. Brown, C.F. Aliferis, The max-min hillclimbing BN structure learning algorithm, *Machine Learning*, 65: 31-78.

[19] D. Karaboga, An idea based on honey bee swarm for numerical optimization, *Technical Report-TR06*, Erciyes University, Engineering Faculty, Computer Engineering Department.

[20] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing*, 8: 687-697.

[21] R.W. Robinson, Counting unlabeled acyclic digraphs, *Combinational Mathematics*, 622: 28-43.

[22] D. Chickering, D. Heckerman, C. Meek, Large-Sample Learning of Bayesian Networks is NP-hard, *Journal of Machine Learning Research*, 5: 1287-1330.

[23] G. Cooper, E. Hersovits, A bayesian method for the induction of probabilistic networks from data, *Machine Learning*, 9: 309-347.

[24] P.C. Pinto, A. Nägele, M. Dejori, T.A. Runkler, J.M.C. Sousa, Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning, *IEEE transactions on evolutionary computation*, 13(4): 767-779.

[25] K. Murphy, The Bayes net toolbox for Matlab, *Computing Science and Statistics*, 33: 331-350.

[26] C.F. Aliferis, I. Tsamardinos, A. statnikov, L.E. Brown, Causal Explorer: A causal probabilistic network learning tooltik for biomedical discovery, *In Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, 371-376.

[27] S.Lauritzen, D.Spiegelhalter, Local Computations with Probabilities on Graphical Structures and Their Application on Expert Systems, *J.Royal Statistical Soc.*, 50: 157-224.

[28] I. Beinlich, G. Suermondt, R. Chavez, G.Cooper, The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks, *Proc. Second European Conf. Artificial Intelligence in Medicine.*

[29] J. Binder, D. Koller, S.J. Ruddell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, *Machine Learning*, 29: 213-244.

[30] T. Hrycej, Gibbs sampling in BNs, *Artificial Intelligence*, 46(3): 351-363.