Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/081L40

A Method of Learning Latent Variables Dimensionality for Bayesian Networks

*Zan Zhang**, *Hao Wang and Hongliang Yao*

School of Computer and Information, Hefei University of Technology, Hefei 230009, P. R. China

Received: 16 May. 2013, Revised: 10 Sep. 2013, Accepted: 11 Sep. 2013
Published online: 1 Apr. 2014

Abstract: Latent variables often play an important role in improving the quality of the learned Bayesian networks and understanding the nature of interactions in the model. The dimensionality of latent variables has significant effect on the representation quality and complexity of the model. The maximum possible dimensionality of a latent variable is a Cartesian product of the state space of its Markov blanket variables. In order to obtain the dimensionality of the latent variable, we need to calculate the network score for every possible dimensionality of the latent variable, and the calculations of this task are substantial. Besides, we do not know the data and condition probability table of the latent variable which makes the task difficult. In this paper, we propose a novel method to learn the dimensionality of the latent variable when the network structure is known. Firstly, we use the latent variable and its Markov blanket variables to extract a local network from original network. Then, we score the local network instead of the original network which reduces the running time of the task. Secondly, we utilize a state-clustering method to score the network for each dimensionality of the latent variable, where a simulated annealing strategy is introduced to avoid local optimum. Finally, based on the above stages, we choose the dimensionality of the latent variable which can make the network get the best score. This new method has excellent learning performance and can deal with complex networks. Extensive experiments validate the effectiveness of our method against other algorithms.

Keywords: Bayesian Networks, Latent Variable, Dimensionality Learning

1 Introduction

A serious problem in learning Bayesian Networks (short for BNs) is the presence of latent variables that are never observed, yet interact with observed variables [1]. The existence of latent variables is common, such as when we detected latent variables in fix structure (one assessed by expert) and in cases where we want to introduce new variables to network for improving the model. In order to discover the dimensionality of latent variable, we need to calculate the score of network for each possible dimensionality of the latent variable, and the calculation amount of this task is huge. We do not know the training data and parameters of the latent variable which makes the task difficult. The first method for learning the dimensionality of the latent variable is EM-based scores method [1]. Later, Elidan and Friedman propose an approach that utilizes a score-based agglomerative state-clustering, called Agglomeration algorithm [2]. Traditional methods still have their own drawbacks.

(1) The time complexity of traditional methods is too high.

The EM-based scores method applies the EM algorithm [3,4] to learn parameters for the network containing the latent variable with each possible dimensionality. This method should perform several EM runs from different random start points. So the time complexity of the EM-based scores method is too high. Agglomeration algorithm has the cubic running time [2].

(2) Traditional methods often trap in a local optimum.

Since the EM algorithm often traps in a local optimum, thus, the EM-based scores method often traps in a local optimum. Agglomeration algorithm is a hill-climbing method [2]. Agglomeration algorithm only cares about the highest score in one iteration, thus, it often traps in a local optimum like other hill-climbing methods.

In this paper, we propose a novel algorithm for learning the latent variable dimensionality based on state-clustering and simulated annealing (short for SSA algorithm). Our contribution is the learning performance and running time

* Corresponding author e-mail: zz_bns@163.com

of our method are much better than the other methods. Our method can deal with the complex network.

Our method consists of two phases, namely, the extract local network phase and the merge states phase. Based on the Markov Independencies [5,6] and decomposition of Bayesian scoring (short for BDe) metric [7], when we use BDe metric to score the network, variables which independent with the latent variable will not influence the networks score while the latent variable takes different dimensionality. In the extract local network phase, firstly, we abandon variables which condition independent with latent variable. We use the latent variable and its Markov blanket variables to extract a local network. We only need the training data of variables in local network instead of the original training data. Since we calculate the score of local network instead of original network, the score process can be accelerated and will not affect the learning results. Secondly, we determine the maximal possible dimensionality of latent variable. In the merge states phase, our method starts with the maximal possible dimensionality of latent variable. We maintain a hard assignment to the latent variable in the training data at each iteration. Thus, we can score the data by the complete data scoring functions that are orders of magnitude more efficient than standard EM-based scores method. The procedure progresses by choosing the two states whose merger will lead to the improvement in the score, where an optimization strategy based on a Metropolis rule of simulated annealing [8,9] is employed to avoid local optimum and enhance the merge effectiveness and efficiency. These steps are repeated until all the states are merged into one state. Based on the two phases, we discover the dimensionality of the latent variable. Extensive experiments show that the SSA algorithm outperforms the other methods.

The paper is organized as follows. In Section 2, we present the related work. In Section 3, we describe our preliminary. Section 4 we describe our new algorithm in detail. Section 5 reports our experimental results. Finally, we conclude the paper in Section 6.

2 Related Work

We are given training data D of samples from $X=\{x_1,...,x_n\}$, and a network structure G over X and a latent variable T. We need to discover the dimensionality of T.

Firstly, EM-scoring method assumes the maximal dimensionality of T. Secondly, this method applies the EM algorithm to learn parameters for the networks containing T for each possible dimensionality. Thirdly, it approximates the score of the network combined with parameters for each possible dimensionality. Finally, it chooses the dimensionality of T which leads to the best score. The central problem of this approach is its exhaustiveness.

Elidan propose the Agglomeration algorithm. Firstly, this algorithm initializes the algorithm with a variable T that has many states. Agglomeration algorithm maintains a hard assignment to T in the training data. The assignment is the state that holds in the instance. Secondly, Agglomeration algorithm calculates the score of the network for each possible dimensionality of T. At each stage, the algorithm chooses the pair of states which leads to the largest increase (or smallest decrease) to merge. These steps are repeated until T has a single state. The time complexity of Agglomeration algorithm is cubic and the Agglomeration algorithm is a hill-climbing method.

3 Preliminary

3.1 Bayesian Networks

Consider a finite set $X = \{X_1,...,X_n\}$ of discrete random variables where each variable X_i may takes states from a finite set, denoted by $Dim(X_i)$. A Bayesian network is a Directed Acyclic Graph (short for DAG) $G = (X,A)$, Where each arc $a_{ij} \notin A$ describes a direct dependence relationship between two variables $Dim(T)$ and X_j . Each node is annotated with a conditional probability distribution (short for CPD) that represents $P(X_i|Pa(X_i))$, where $Pa(X_i)$ denotes the parents of X_i in G. It can be proved that a Bayesian network (X,A) uniquely encodes the joint probability distribution of the domain variables $X = \{X_1,...,X_n\}$:

$$P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{1}$$

3.2 Markov Independencies

Definition 1 Each variable X_i is independent of its non-descendants, given its parents in G.

One implication of the Markov independencies is that a variable X_i interacts directly only with its Markov Blanket which includes the Xs parents, children, and spouses.

3.3 Bayesian Scoring Metric

Bayesian scoring metric is a well-known measure for learning BNs from data. This scoring metric uses a balance between the likelihood gain of the learned model and the complexity of the network structure representation [2].

The initial expression of *BDe* score is:

$$BDe(G|D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left[\log\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+m_{ij})} + \sum_{k=1}^{r_i}\log\frac{\Gamma(\alpha_{ijk}+m_{ijk})}{\Gamma(\alpha_{ijk})}\right] \tag{2}$$

The best *BDe* score is the biggest one which is related to the optimal BNs structure. D is a given training set, G is a possible network structure, r_i is the number of possible values of the X_i , q_i is the number of possible configurations(instantiations) for the variables in $Pa(X_i)$. α_{ij} and α_{ijk} are hyper-parameters of the prior distribution of parameterization, $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$. m_{ij} and m_{ijk} are data sufficient statistics. $m_{ij} = \sum_{k=1}^{r_i} m_{ijk}$, m_{ijk} is the number of cases in D where X_i has its $k'th$ value and $Pa(X_i)$ is instantiated to its $j'th$ value.

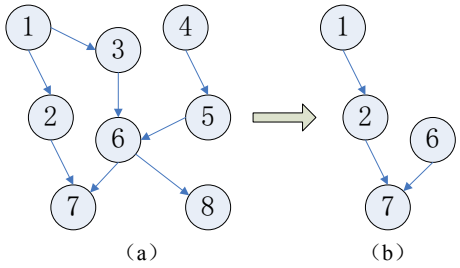

Fig. 1: The process of extracting a local network

4 SSA Algorithm

SSA algorithm consists of two phases, namely, the extract local network phase and the merge states phase. We describe the details below.

4.1 Extract Local Network

We calculate the network score for every possible dimensionality of T . Since the *BDe* score is local decomposable, we dont need to score the original network. We can extract a local network instead of original network. The *BDe* score can be rewritten as the sum

$$Score_{BDe}(G:D) = \sum_i FamScore_{BDe}(X_i, Pa(i):D) \qquad (3)$$

Definition 2 [Heckerman et al.,1995a] Let G be a network structure and $P(\theta|G)$ be a parameter prior satisfying parameter independence and parameter modularity. Using full table CPDs and a Dirichlet prior with hyper-parameters $\alpha_{X_i|Pa(i)}$ then:

$$\begin{aligned} &FamScore_{BDe}(X_i, Pa(i):D) = \\ &\sum_{pa(i)} \left[\log \frac{\Gamma(\alpha_{pa(i)})}{\Gamma(\alpha_{pa(i)} + S[pa(i)])} + \right. \\ &\left. \sum_{x_i} \log \frac{\Gamma(\alpha_{x_i|pa(i)} + S[x_i, pa(i)])}{\Gamma(\alpha_{x_i|pa(i)})} \right] \end{aligned} \qquad (4)$$

Where Γ is the Gamma function and $\alpha_{pa(i)} = \sum_{x_i} \alpha_{x_i|pa(i)}$ and $S[pa(i)] = \sum_{x_i} S[x_i, pa(i)]$.

Theorem 1 Let G be a network over the discrete variables $\chi' = \{X_1, ..., X_n\}$ and the latent variable T.Let D be a set of M instances, where all the variables in χ are observed. Let G' be a network over the discrete variables $\chi' = \{MB(T)\}$ and T. Let D' be a set of M instances, where all the variables in χ' are observed. Let i and j be two states of T. After a merge of i and j, the value i and j are replaced with a new state denote by $i \cdot j$.

If $\Delta S = Score_{BDe}(G_{i\cdot j} : D) - Score_{BDe}(G_{i,j} : D)$, $\Delta S' = Score_{BDe}(G'_{i\cdot j} : D') - Score_{BDe}(G'_{i,j} : D')$
Then $\Delta S = \Delta S'$

That is, comparing the difference between the *BDe* score after and before the merge of states i and j of T in G , only need to compare the difference in G' with D'.

Proof. $\Delta S = Score_{BDe}(G_{i\cdot j} : D) - Score_{BDe}(G_{i,j} : D) =$
$\sum_c \sum_{pa(c)} \left[\log \frac{\Gamma(\alpha(pa(c),T=i\cdot j))}{\Gamma(S^+[pa(c),T=i\cdot j])} - \log \frac{\Gamma(\alpha(pa(c),T=i))}{\Gamma(S^+[pa(c),T=i])} - \right.$
$\log \frac{\Gamma(\alpha(pa(c),T=j))}{\Gamma(S^+[pa(c),T=j])} + \sum_c \log \frac{\Gamma(S^+[c,pa(c),T=i\cdot j])}{\alpha(c,pa(c),T=i\cdot j))} -$
$\left. \sum_c \log \frac{\Gamma(S^+[c,pa(c),T=i])}{\alpha(c,pa(c),T=i))} - \sum_c \log \frac{\Gamma(S^+[c,pa(c),T=j])}{\alpha(c,pa(c),T=j))} \right] +$
$\sum_{pa(T)} \left[\log \frac{\Gamma(S^+[t_{i\cdot j},pa(T)])}{\Gamma(\alpha(t_{i\cdot j},pa(T)))} - \log \frac{\Gamma(S^+[t_i,pa(T)])}{\Gamma(\alpha(t_i,pa(T)))} - \right.$
$\left. \log \frac{\Gamma(S^+[t_j,pa(T)])}{\Gamma(\alpha(t_j,pa(T)))} \right]$

where $alpha(t_{i\cdot j} = alpha(t_i + alpha(t_j$ and $S^+[x] = S[x] + \alpha(x)$. The first summation is over all C that are children of T and corresponds to the families of the children of and their parents and the second summation corresponds to the family of T and its parents. Thus, we get the Theorem 1.

Figure 1 shows an example of extracting a local network, where G is Asia network and G' is a local network of G . Assume the variable 2 is a latent variable. Only variable 2 and $MB(2)$ can affect the change of scores. We extract a local network with variable 2, variable 1, variable 6 and variable 7. Now we dont need the training data D anymore, we just need the data D' of those variables in local network from D.

After we extracted the local network, at each iteration we maintain a hard assignment to T in the D' . We can represent this assignment as a mapping δ_T from $1, ..., M$, to the set $Dim(T)$. $\delta_T(m)$ is the state that T holds in the instance.

We need to determine the maximum possible dimensionality of T . Recall that the Markov blanket of T separates it from all other variables. This implies that two instances in which $MB(T)$ have the same states, are identical from T's perspective. Thus, the largest number of states of latent variable that are relevant for a given data sets is the number of distinct assignment to $MB(T)$ in the data [2]. In the example of Figure 2, Markov blanket of variable 13 in Alarm network is variable 2, variable 12, variable 27 and variable 30. Their

dimensionalities are 3, 2, 3 and 3. So the maximum possible dimensionality of variable 13 is $Val(13) \leq 3*2*3*3 = 54$, specific number determined by D' . We find only 13 assignments (out of 54 possible). We then augment D' with these assignments to T . That is, for each assignment $c \in Dim\{MB(T)\}$, we have a t_v for each instance m. We set $\delta_T(m)$ to be the state t_v consistent with the Markov blanket assignment of instance m .

4.2 Merge States Based on Simulated Annealing Strategy

We merge two states of T in each step. Let i and j be two states of T. After a merge of i and j, the value i and j are replaced with a new state denoted by $i \cdot j$. We then reevaluate the network with respect to this assignment, and so on. These steps are repeated until T has a single state. We return the number of states k that receive the highest score. To overcome the local optimum and improve the quality of choosing the (i, j), we introduce a merging strategy based on a simulated annealing strategy to enhance merge efficiency in the SSA algorithm. Before conducting the merge process at each step, SSA algorithm compares the score of network in the current iteration with that of last iteration, and then determines whether to carry out the merging process. The practical Metropolis rule can be denoted as:

$$P = \begin{cases} 1 & \text{if } \Delta S \leq 0 \\ \exp(-\frac{\Delta S}{t_k}) & \text{otherwise} \end{cases} \quad (5)$$

Where ΔS is the score difference of the solution obtained at two iterations, and t_k is the annealing temperature $t_k = \frac{K-k}{K} t_0 (K = L-1, K = 1, 2, ..., K)$. If the score of the solution at the current iteration is smaller than that of the last iteration, we give up merging current (i, j), and choose other couple. On the contrary, when the score of the solution at the current iteration is better, the optimizing process is randomly carried out at a certain probability. Moreover, the annealing temperature will reduce as the merging runs, and hence the random merging process will gradually decrease. When $t_k \to 0$, this strategy only performs merging for the cases of stagnating solutions. If we meet the worst situation: we cant find any couple states to merge to raise the score of network. We merge the couple which can lead to the highest scoring network. Figure 2 shows an example of merging states process.

4.3 SSA Algorithm

With the theoretical analysis above, we propose the new approach named SSA algorithm.

1. Initialization:
Initialize: T,G ,D ,t_0

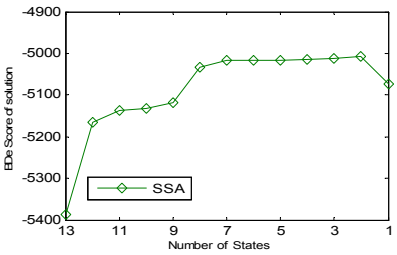

Fig. 2: Trace of the merging states process in a simple synthetic example. We sampled 500 instances from the Alarm network, and then hid the observations of the variable 13 in the data. The real dimensionality of variable 13 is 2. The result shown is for recovering the dimensionality of variable 13.

2. Extract local network phase:
Given network G and latent variable T
//By Definition 1
Get $MB(T)$ from G
Extract local network G' with T and $MB(T)$
Keep the training data of $MB(T)$ in D'
$W_{MB} \leftarrow \{1, ..., K\}$, an ordering of unique assignment to $MB(T)$ in D'
for $m \leftarrow 1$ to M do
$\delta_T[m] \leftarrow W_{MB}[MB_T(M)]$//Complete data of T by assignment
end
3. Merge states phase:
for $k \leftarrow 1$ to K do
$(i, j) \leftarrow Merge(D', T, \delta_T)$
if $\Delta S = S_k - S_{k-1} > 0$ and $random > exp(-\Delta S/t_k)$
Confirm merge (i, j)
endif
if $\Delta S = S_k - S_{k-1} \leq 0$ or $random \leq exp(-\Delta S/t_k)$ then
Give up merge (i, j), choose other couple states.
endif
for each $\delta_T[m] = i$ or j do
$\delta_T[m] = i \cdot j$
end foreach
end
Return the number of states which received the highest score

4.4 Algorithm Analysis

Traditional methods have their drawbacks. Since the EM-based scores method applies EM algorithm repeatedly, it makes time complexity of this method too high and often traps in a local optimum. The Agglomeration is a hill-climbing method, also often traps in a local optimum. Agglomeration algorithm has the cubic running time. SSA algorithm extracts a local network to remove the redundant calculations, even in the worst situation the running time of SSA algorithm is less

Table 1: Data set used in experiments

Data set(D)	Original network(G)	Size of D	Nodes of G
Asia500	Asia	500	8
Asia1000	Asia	1000	8
Asia1500	Asia	1500	8
Asia2000	Asia	2000	8
Asia2500	Asia	2500	8
Asia3000	Asia	3000	8
Alarm500	Alarm	500	37
Alarm2000	Alarm	2000	37

Table 2: Variables used in our experiments from Asia

Latent Variable	Real Dimensionality
Variable1	2
Variable2	2
Variable3	2
Variable5	2
Variable6	2

than Agglomeration. SSA algorithm applies a simulated annealing strategy to control the process of merge states. Simulated annealing has been proven to be an efficient approach to avoid trap in a local optimum. So the learning results of SSA algorithm are better than the traditional methods.

5 Experimental Results

5.1 Experiment Setup

To assess the performance of the SSA algorithm, we use a common evaluation method, which is to test the algorithm on data sets generated from known networks using probabilistic logic samples. We test SSA algorithm on 8 different data sets, and compare the result with that of the EM-based scores method and Agglomeration on the same data sets. All of the data sets are generated from well-known benchmarks of BNs including the Asia network and the Alarm network [10]. Table 1 shows a summary of data sets used in our experiments. We use the method proposed by Elidan et al. [2] to choose variables for experiments. We choose 5 variables from Asia and 24 variables from Alarm. (We didnt consider variables that are either leaf or had few neighbors.) We consider the estimated dimensionality had one state more or less than the true dimensionality is near-perfect results. Table 2 and Table 3 shoes a summary of variables for experiments. The EM-based scores method is independently executed 10 times for each data set, and the running time is an average of 10 trails. The experimental platform was a PC with Pentium 4, 2.8 GHz CPU, 2G memory, and Windows XP. The algorithm was implemented by MATLAB. By large number of experiments, we set $t_0 = 3000$.

Table 3: Variables used in our experiments from Alarm

Latent Variable	Real Dim	Latent Variable	Real Dim
Variable1	2	Variable15	4
Variable2	3	Variable16	3
Variable3	2	Variable19	3
Variable4	2	Variable21	4
Variable5	2	Variable23	4
Variable7	2	Variable26	4
Variable8	2	Variable27	3
Variable 9	2	Variable28	3
Variable10	2	Variable30	3
Variable12	2	Variable31	2
Variable13	2	Variable32	3
Variable14	3	Variable35	3

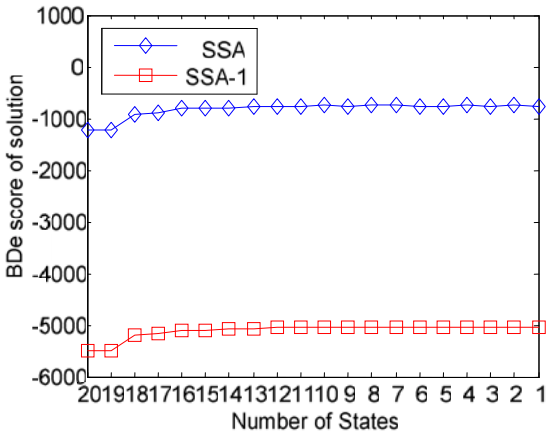

Fig. 3: Process of merging states by SSA and SSA-1

5.2 Contributions of Extract Local Network

We employ SSA-1 algorithm (dont extract a local network) and SSA algorithm to learn the dimensionality of variable10 in Alarm on Alarm500 data set. Its real dimensionality is 2. We hide the data of the variable10. The experimental results are shown in Figure 3 and the running time is shown in Table 4. In the example of only 20 states (out of 54 possible) were observed in the data. From Figure 3 and Table 4, we can see the learning results of SSA-1 and SSA are correct, but SSA-1 needs 35.765001 seconds while SSA only needs 3.104910 seconds. We can draw the conclusion that the extract local network phase can effective reduce the running time of learning the latent variable dimensionality. The time complexity of SSA algorithm will not increase with the complex of network, because we only scoring the local network of latent variables.

Table 4: Running time of SSA-1 and SSA

Running time of SSA-1(s)	Running time of SSA(s)
35.765001	3.104910

Table 5: Comparison of results on Asia network

Dataset	Statistic	Algorithm		
		EM-based	Agglomeration	SSA
Asia500	Correct	1	4	5
	Missing	0	0	0
	Extra	2	1	0
	Incorrect	2	0	0
Asia1000	Correct	0	4	4
	Missing	0	0	0
	Extra	3	0	1
	Incorrect	2	1	0
Asia1500	Correct	1	4	4
	Missing	0	0	0
	Extra	2	1	1
	Incorrect	2	0	0
Asia2000	Correct	0	5	5
	Missing	0	0	0
	Extra	3	0	0
	Incorrect	2	0	0
Asia2500	Correct	0	5	5
	Missing	0	0	0
	Extra	2	0	0
	Incorrect	3	0	0
Asia3000	Correct	0	3	4
	Missing	0	0	0
	Extra	3	1	0
	Incorrect	2	1	1

5.3 Comparison the Predictions of Dimensionality

Table 5 report the detailed results of the SSA algorithm against EM-based scores method and Agglomeration algorithm on the 6 datasets of Asia network. SSA algorithm gets better results than EM-based scores method on all datasets. SSA algorithm gets better results than Agglomeration algorithm on Asia500, Asia1000 and Asia3000. On the other three datasets, SSA algorithm and Agglomeration algorithm are all get perfect prediction of dimensionality. The learning results of SSA algorithm are better than the other methods.

Comparison of learning results with SSA algorithm and traditional methods on the 2 data sets of Alarm network is shown in Table 6. On the Alarm500 data set, SSA algorithm recovered the correct dimensionality for 8 variable and get near-perfect prediction of dimensionality for 8 variables. EM-based scores method recovered the correct dimensionality for 1 variable and get near-perfect prediction of dimensionality for 4 variables. Agglomeration recovered the correct dimensionality for 5 variables and get near-perfect prediction of dimensionality for 10 variables. On the Alarm2000 data set, we can see that SSA algorithm recovered the correct dimensionality for 4 variables and get near-perfect prediction of dimensionality for 11 variables. EM-based scores method only get near-perfect prediction of dimensionality for 4 variables. Agglomeration recovered the correct dimensionality for 4 variable and get near-perfect prediction of dimensionality for 6 variables. SSA algorithm is better than the other methods. Alarm network is a complex standard BNs with 37 variables. The experiment results show that the SSA algorithm can lead to excellent performances on the complex network.

Table 6: Comparison of results on Alarm network

Dataset	Statistic	Algorithm		
		EM-based	Agglomeration	SSA
Alarm500	Correct	1	5	8
	Missing	2	3	5
	Extra	2	7	3
	Incorrect	19	9	8
Alarm2000	Correct	0	4	4
	Missing	2	2	7
	Extra	2	6	4
	Incorrect	20	12	9

5.4 Comparison of the Running Time

Figure 4 to Figure 9 report the average running time on Asia network of SSA algorithm against traditional methods. SSA algorithm is faster than the other methods on all the 6 datasets. Comparison of running time over Alarm network of SSA algorithm against traditional methods are shown in Figure 10 and Figure 11. We can see that SSA algorithm is faster than Agglomeration algorithm. Some variables like variable 16, variable 28 and variable 32 which Markov blankets are much complex. The running time of Agglomeration on these variables are much more than the other variables. But the running time of SSA algorithm on every variable is very little. From figure 10, we can see that the running time of SSA algorithm is less than EM-based scores method on all variables except variable23, variable31 and variable32. From figure 11, we can see that the running time of SSA algorithm is less than EM-based scores method on 16 variables.

6 Conclusion

Learning the dimensionality of latent variables is a challenging issue. In this paper, we analysis the decomposable of score metric and extract a local network based on Markov blanket of latent variable. Then an optimization strategy based on a Metropolis rule of simulated annealing is employed to improve the process of merging states. This new method can not only get better learning performances, but also can deal with the

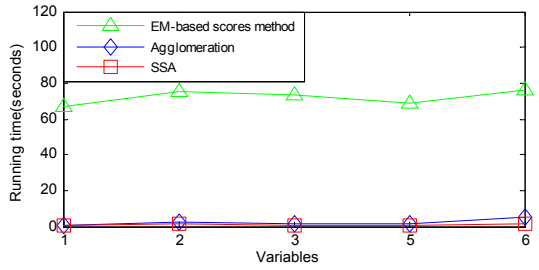

Fig. 4: Comparison of running time on Asia500

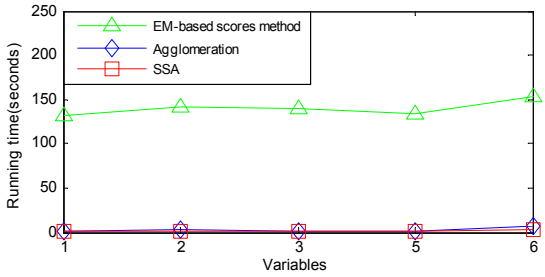

Fig. 5: Comparison of running time on Asia1000

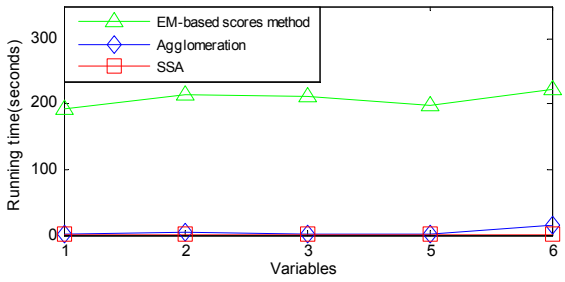

Fig. 6: Comparison of running time on Asia1500

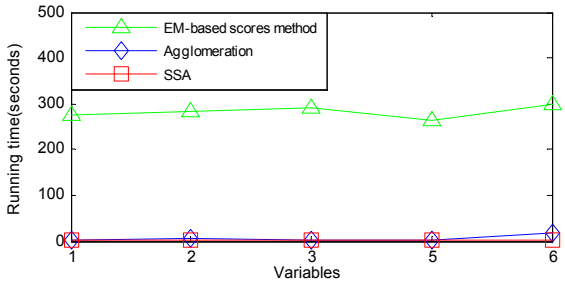

Fig. 7: Comparison of running time on Asia2000

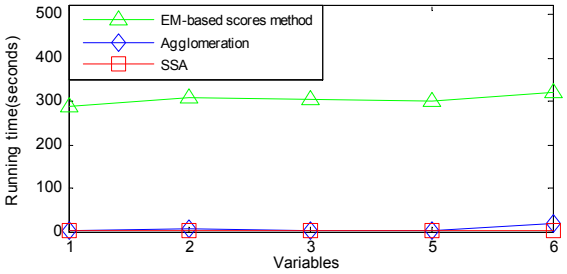

Fig. 8: Comparison of running time on Asia2500

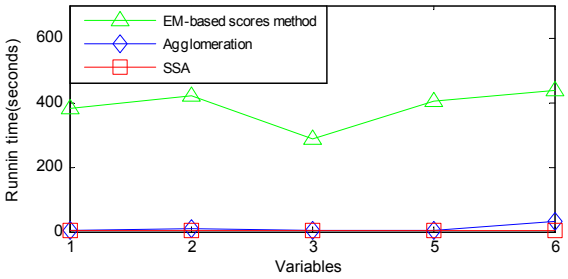

Fig. 9: Comparison of running time on Asia3000

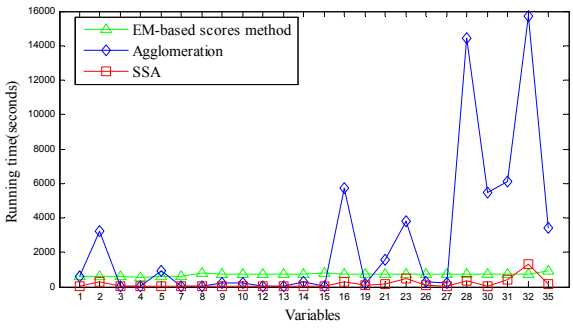

Fig. 10: Comparison of running time on Alarm500

complex networks. Experimental results demonstrate that SSA algorithm can lead to better performances of learning the latent variables dimensionality.

Acknowledgement

This work is supported by the National Natural Science Foundation of China 61070131, 61175051 and 60975034.

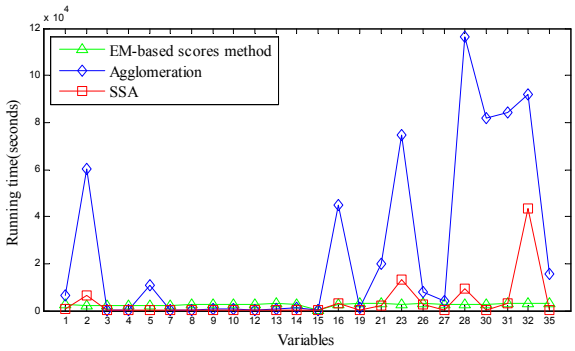

Fig. 11: Comparison of running time on Alarm2000

References

[1] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman, AutoClass: a Bayesian classification system, in the Proceedings of the Fifth International Conference on Machine Learning, **27**, 54-64 (1988).

[2] Gal Elidan, Nir Friedman, Learning the Dimensionality of Hidden Variables in the Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence,144-151 (2001).

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society. Series B (Methodological), **39**, 1-39 (1977).

[4] S. L. Lauritzen, The EM algorithm for graphical association models with missing data, Computational Statistics and Data Analysis, **19**, 191-201 (1995).

[5] Constantin F. Aliferis, Alexander R. Statnikov, Ioannis Tsamardinos, Subramani Mani, Xenofon D.Koutsoukos, Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification Part I: Algorithms and Empirical Evaluation, Journal of Machine Learning Research, **11**, 171-234 (2010).

[6] Fu, S.-K. and M.C. Desmarais, Tradeoff Analysis of Different Markov Blanket Local Learning Approaches, Advances in Knowledge Discovery and Data Mining, **5012**, 562-571 (2008).

[7] D. Heckerman, D. Geiger, and D. M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data, Machine Learning, **20**, 197-243 (1995).

[8] D. Fouskakis, Bayesian variable selection in generalized linear models using a combination of stochastic optimization methods, European Journal of Operational Research, **220**, 414-422 (2012)

[9] Vincent Granville, Mirko Kvanek, and Jean-Paul Rasson, Simulated annealing: A proof of convergence. IEEE Transactions on Pattern Analysis and Machine Intelligence, **16**, 652-656 (1994).

[10] www.ai.nit.edu/murphyk/Software/BNST/BNst.html.

Zan Zhang is a MS student in school of Computer and Information, Hefei University of Technology. He received the BS degree in the department of Economic management, Northeast Dianli University in 2009. His research interests are in the areas of machine learning and data mining.

Hao Wang Hao Wang received the MS degree in Computer science from Hefei University of Technology in 1989, and the PhD degree in Computer science from Hefei University of Technology in 1997. He is currently a professor in Hefei University of Technology. His main research interests artificial intelligence.

Hongliang Yao received the MS degree in Mathematics from Anhui University in 1997 and the PhD degree in Computer science from the school of Computer and Information, Hefei University of Technology in 2007. He is currently an associate professor in Hefei University of Technology. His research interests are in the areas of artificial intelligence and knowledge engineering.