Bayesian Substructure Learning - Approximate Learning of Very Large Network Structures

Andreas Nägele[1,2], Mathäus Dejori[1], and Martin Stetter[1]

[1] Siemens AG, Corporate Technology, CT IC 4, D-81730 Munich, Germany
[2] Dept. of Computer Science, Technical University of Munich, D-85747 Garching, Germany

Abstract. In recent years, Bayesian networks became a popular framework to estimate the dependency structure of a set of variables. However, due to the NP-hardness of structure learning, this is a challenging task and typical state-of-the art algorithms fail to learn in domains with several thousands of variables. In this paper we introduce a novel algorithm, called substructure learning, that reduces the complexity of learning large networks by splitting this task into several small subtasks. Instead of learning one complete network, we estimate the network structure iteratively by learning small subnetworks. Results from several benchmark cases show that substructure learning efficiently reconstructs the network structure in large domains with high accuracy.

Keywords: Graphical Models, Bayesian Networks, Structure Learning.

1 Introduction

Bayesian networks (BNs) are popular graphical models to describe the dependencies between a set of random variables in a probabilistic as well as graph theoretic way. They provide a consistent and intuitive graphical representation of higher-order statistics between these variables. One important task for BNs that became important in recent years is the learning of the qualitative dependency structure from data. Due to the NP-completeness of structure learning [6], many interesting domains for Bayesian network learning face the problem of high dimensionality. The computational time of learning Bayesian networks can be reduced by applying heuristic assumptions about possible network structures combined with heuristic search strategies. For example, the "Sparse Candidate" algorithm with polynomial computational complexity was introduced [11]. This algorithm restricts the number of possible parents for each variable to a small number and allows only edges between a variable and its "candidate" parents. The basic idea behind this algorithm is following heuristic argument: If variables X_1 and X_2 are almost independent in the data, they are unlikely to be connected in a Bayesian network and, thus, the search space of possible network structures can be restricted on those that have no edge between X_1 and X_2. Recently, a very competitive algorithm was introduced [16]. This so called *MMHC* algorithm uses a constraint based method to detect possible parent-child relationships and

J.N. Kok et al. (Eds.): ECML 2007, LNAI 4701, pp. 238–249, 2007.

combines all these relationships to an undirected skeleton of the network structure. Afterwards, a score-based greedy search procedure is used to learn the edge orientation based on the skeleton.

However, structure learning in domains with tens of thousands of variables is intractable for current learning methods given the available computational power except for very restricting cases with boolean variables paired with very sparsely linked graphs [12]. Areas of applications for large networks are, besides others, social networks, warehousing or biological processes. For learning in such large domains, one typically restricts the feature dimensions on a feasible subset of relevant variables that are of high interest, as it was done in [10] for the estimation of biological processes.

Here we introduce the *substructure learning* algorithm which combines the approach of restricting possible network structures with dimensionality reduction. This approach is shown to be efficient in terms of accuracy and scalability for structure learning in large domains. In a first step, we learn the skeleton of the network structure. In a second step, for each variable, we learn one small subnetwork to estimate the local structure "around" this variable. Thereby, the possible network structures are restricted, based on the undirected skeleton. By learning one subnetwork for each variable in the complete network, we systematically estimate the complete directed network structure step by step, without learning one single Bayesian network for the whole domain. Based on all subnetworks, we provide a graphical representation of the directed dependency structure using the framework of a fPDAG (feature partial directed graph) to enable a unifying representation of all small and local subnetworks.

2 Methods

2.1 Background

At first, we provide a brief summary of learning multinomial Bayesian networks (BNs) and instead direct the interested reader to [14,16] for further information. Bayesian networks can be used to describe the joint probability distribution of n random variables $\mathbf{X} = \{X_1, X_2, X_3, ..., X_n\}$. A Bayesian network $B = (\mathcal{G}, \Theta)$ consists of two parts. The first part is the network structure which is a directed acyclic graph (DAG) $\mathcal{G}$ with each variable X_i represented as a node. The edges in the DAG represent statistical dependencies between the variables. The second part is a set of parameters describing the probability density. With the independence statements encoded in the DAG the joint probability function over $\mathbf{X}$ can be decomposed into the product form

$$p(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} p(X_i | \mathbf{Pa}_i, \Theta, \mathcal{G}), \tag{1}$$

where $\mathbf{Pa}_i$ are the parents of variable X_i in the DAG $\mathcal{G}$.

Learning the structure and the parameters of a Bayesian network from a data set $\mathbf{D}$ can be formulated as the following problem: Given a finite data

set $\mathbf{D} = (\mathbf{d}^1, ..., \mathbf{d}^N)$ with N different independent observations, where each data point $\mathbf{d}^l = (d_1^l, ..., d_n^l)$ is an observation of all n variables, find the graph structure $\mathcal{G}$ and the parameters Θ that best match data set $\mathbf{D}$. In the Bayesian approach this implies maximizing the function

$$p(\mathcal{G}|\mathbf{D}) = \frac{p(\mathbf{D}|\mathcal{G})p(\mathcal{G})}{p(\mathbf{D})}, \tag{2}$$

where $p(\mathcal{G})$ is the prior for the structure, $p(\mathbf{D})$ a normalization constant and $p(\mathbf{D}|\mathcal{G})$ the marginal likelihood of $\mathbf{D}$ given the model graph $\mathcal{G}$.

By using an uniform prior over all possible network structures, the learning problem can be reduced by searching solely the structure with best marginal likelihood

$$p(\mathbf{D}|\mathcal{G}) = \int p(\mathbf{D}|\mathcal{G}, \Theta)p(\Theta|\mathcal{G})d\Theta, \tag{3}$$

where $p(\mathbf{D}|\Theta, \mathcal{G})$ is the likelihood of the data set $\mathbf{D}$ given the Bayesian network $(\mathcal{G}, \Theta)$ and $p(\Theta|\mathcal{G})$ denotes the prior for the local probability distributions Θ of the Bayesian network with structure $\mathcal{G}$.

The approach described so far is commonly referred to as "score-based" approach since the DAGs are rated by their score. Constraint-based methods form the second class of structure learning algorithms. Instead of searching the optimal scoring DAG, they reconstruct the network by applying conditional independence tests on the data. Recently, a new and quite competitive algorithm that combines both approaches was developed [16]. The so called MMHC (max-min hill-climbing) algorithm performs Bayesian network learning in two steps: firstly, an undirected network skeleton is estimated with MMPC (max-min parents and children) that employs constraint-based techniques. Afterwards, a score-based greedy search is performed to orient the edges in order to obtain a high-scoring DAG.

Our substructure algorithm utilizes the same approach to estimate the skeleton, thus we shortly summarize the MMPC algorithm. For more details we direct the interested reader to [16]. MMPC is a local discovery algorithm to assess the set of parents and children $\mathbf{PC}_i$ of a variable X_i. This is done in two phases. In the first phase, conditionally dependent variables can enter the set of candidate parents and children according to a heuristic function which is called Max-Min heuristic: This variable enters next the candidate set that maximizes the minimum association to X_i given the current candidate set. Thereby, the minimum association is defined as the minimal conditional dependency of a variable and X_i, tested for all possible subsets of the current candidate set. This means that this variable enters the candidate set which is most unlikely to be conditionally independent from X_i. The growing phase stops after all dependent variables have entered the candidate set. In the second phase, the false positive variables are removed which possibly entered the candidate set in the first phase. False positives are such variables that are independent of X_i given some subset of all variables. Thus, all variables that are conditionally independent given a subset

of the candidates are removed from the candidate set. The authors of MMPC have shown that under the assumption of faithfulness this algorithm will return no false negatives. It also returns no false positives if the **PC** relation is made symmetric, i.e. for all $X_j \in \mathbf{PC}_i$ it is tested whether $X_i \in \mathbf{PC}_j$; if this condition is not fulfilled, X_j is removed from $\mathbf{PC}_i$. To construct the skeleton of the Bayesian network, the MMPC algorithm is performed for all variables, and each variable is connected to all members of its set of parents and children.

It has been shown that the MMHC algorithm has a good performance in terms of quality as well as runtime and outperforms many state-of-the-art BN learning algorithms such as the Sparse-Candidate algorithm [16,11]. However, MMHC faces the problem of its computational complexity if applied in a domain with more than several thousand variables. While the skeleton reconstruction phase is quite efficient, the edge orientation phase is the limiting part in the MMHC algorithm. The authors have reported from a benchmark with 5000 variables where the first phase took 19 hours on a Pentium Xeon with 2.4 GHz, while the edge orientation took almost two weeks [16].

2.2 Substructure Learning

In this section we introduce substructure learning as an efficient and scalable method for estimating the structural dependencies in large and sparse domains. The general idea behind this algorithm is that subparts of very large networks can be learned by omitting unimportant variables, as it is done e.g. for the estimation of the genetic regulatory network where the large amount of about 30,000 genes (variables) is reduced to a small set of relevant genes [10]. How the selection of important network nodes for one subnetwork can influence the learned structure is exemplarily shown in Fig. 1, that is taken from [4]. The removal of one important node (X_7 in this example) can disrupt the structure of the BN. The direct relationships that pass originally over X_7 in left hand network must be represented by indirect relationships between the remaining nodes in the subnetwork on the right hand side, which leads to a massive appearance of false positive and false negative edges, thus leading to an entirely wrong set of relationships.

Based on the idea of dimensionality reduction and instead of learning the whole network, we learn a set of small subnetworks that together resemble the original global structure with high accuracy. The algorithm itself is a two-step process (see Table 1): Firstly, the skeleton $\mathcal{S}$ of the complete network structure is reconstructed. Secondly, small subnetworks are learned independently of each other for an estimation of the complete network structure. The new approach of substructure learning is to estimate the complete network structure by learning several subnetworks, one for each variable in the complete network.

In the first step, our algorithm (lines 2 – 5) is identical to the first phase of MMHC and determines the set of parents and children $\mathbf{PC}_i$ of each variable X_i to reconstruct the skeleton $\mathcal{S}$ of the complete network. In the second step (lines 6 – 12), we introduce a variable selection component by leaving, for each variable X_i, a subnetwork that is centered "around" the variable. This estimation starts with

Table 1. Substructure learning algorithm

```
1:  procedure SUBSTRUCTURE(D)
      Input: data D
      Output: set of Bayesian subnetworks B
      // skeleton reconstruction
2:    for each variable X_i ∈ X do
3:      PC_i := MMPC(X_i,D);
4:    end for
5:    create skeleton S by all PC_i
      // structure learning
6:    for each variable X_i ∈ X do
7:      M_i := NEIGHBOURHOOD(X_i,S);
8:      D_{M_i} := restrict data D on variables in M_i
9:      B_i := LEARN_BN(M_i, D_{M_i}, S);
10:     B_i := restrict B_i on the Markov blanket of X_i and X_i;
11:     B := B ∪ B_i
12:   end for
13:   return B;
14: end procedure
```

the selection of variables $\mathbf{M}_i$ for learning one BN (line 7). For that purpose we utilize the skeleton $\mathcal{S}$ to determine the set $\mathbf{M}_i$ of structurally important variables which is calculated by NEIGHBOURHOOD(X_i,$\mathcal{S}$). This procedure returns a set of variables which includes X_i, the neighbours of X_i in $\mathcal{S}$ and their neighbours. Thus, the central variable X_i of the local structure, the parents and children of X_i and their parents and children are all put together for learning one single BN. This variable selection is the first crucial step since a suboptimal selection with missing variables which are structurally important can lead to false positives as well as false negatives, as it was shown before examplarily.

The second crucial step is the learning of the local Bayesian subnetworks (line 9). As done for MMHC, we restrict edges in the subnetwork to edges that also appear (as undirected edges) in the skeleton, this means an edge between two variables can only be added during structure search if the variables are also connected in the

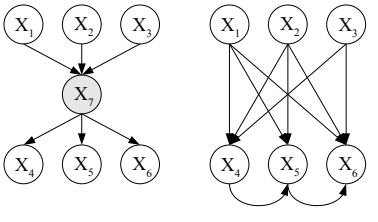

Fig. 1. The left Bayesian network is an example for a complete Bayesian network, the network on the right-hand side is the simplest Bayesian network that encodes the same probability distribution, but without node X_7. Note that the nodes X_4, X_5 and X_6 are no longer independent given their parents.

skeleton. To increase the quality of the network estimation we afterwards restrict the learned subnetwork to the Markov blanket of X_i by removing all variables and edges that do not belong to the Markov blanket or X_i itself. The Markov blanket of a variable is a subset of variables that render this variable independent from all others. In a BN, the Markov blanket of a variable consists of its parents, its children and the parents of its children. The result of the substructure algorithm is the set $\mathbf{B}$, containing all local Bayesian subnetworks B_i, one for each variable. All the local subnetworks allow a structural estimation of the complete DAG and, as well, build a quantitative model for each single variable given its Markov blanket, encoded as a BN.

2.3 Structure Representation

The set of partially overlapping subnetworks $\mathbf{B}$ lacks of a unifying representation of the network structure. While the structure of a single subnetwork forms a DAG, the edges of all subnetworks together need neither to be acyclic nor to have all edges with conforming orientations in all subnetworks. For example, it is likely that there exists an edge between two variables in one subnetwork that has the opposite direction in another subnetwork, or it does not occur in the other subnetwork at all. For a unifying representation of such structural uncertainties we use the framework of feature partial directed graphs (fPDAG) [8,9], which assigns confidences to edge features. The features of an edge between two variables X_i and X_j can be described by a probability distribution with four states, that is

$$p_{i \leftrightarrow j} = \{p_{i \rightarrow j}, p_{i - j}, p_{i \leftarrow j}, p_{i \perp j}\} \ . \tag{4}$$

$p_{i \rightarrow j}$ denotes the probability of a directed edge from X_i to X_j, $p_{i \leftarrow j}$ the probability of a directed edge from X_j to X_i, $p_{i - j}$ the probability of an undirected edge and $p_{i \perp j}$ the probability that there is no edge between the two variables. We estimate the confidence of a feature as the empirical mean of the confidences in the subnetworks:

$$p_{i \leftrightarrow j}(k) = \frac{1}{\alpha} \sum_{g=1}^{n} f_{i \leftrightarrow j}(B_g)(k), \tag{5}$$

where $f(B_g)$ is the truth-value of the feature in network B_g: it is one if the feature appears in the network, otherwise it is zero. The normalization constant α denotes the number of networks that can make a statement about the feature. In more detail, the normalization constant of an edge feature with X_i and X_j as endpoints is the number of networks that contain both variables X_i and X_j. If there is no network that contains both variables, the probability of the edge is set to zero. As the direction of edges that do not belong to a collider structure can be ambiguous [18], the features are not calculated directly from the structure of a Bayesian network but from the PDAG (partial directed acyclic graph) representation of its network structure [7].

The fPDAG of Bayesian networks is a graph which contains all variables that are present in the Bayesian networks. Each edge between two variables X_i and

X_j is weighted with its feature $p_{i \leftrightarrow j}$. Thus, unlike Bayesian networks or PDAGs, the structure of a fPDAG is neither an acyclic directed nor a partially directed acyclic graph. Instead, it is a weighted graph that has edges between related variables, and these edges are labeled with $p_{i \leftrightarrow j}$.

2.4 Time Complexity of Substructure Learning

In the first phase of the substructure algorithm, the skeleton of the underlying dependency structure is reconstructed using MMPC. Each single call on MMPC has a computational complexity of $O(|\mathbf{X}||\mathbf{PC}|^{l+1})$ with l as the maximum size of all conditioning subsets. Thus, the overall cost for reconstructing the whole skeleton is $O(|\mathbf{X}|^2|\mathbf{PC}|^{l+1})$, where $|\mathbf{PC}|$ is the largest set of parents and children over all variables in $\mathbf{X}$ (we refer to [16] for more details). So far, the substructure algorithm does not differ from MMHC. However, in the edge orientation phase substructure learning splits the structure search problem into several small problems.

We now estimate the influence of this splitting on the number of possible network structures. Given a skeleton where each variable has at least two neighbours (or parents in the case of the sparse candidate algorithm), finding the best DAG is NP-hard in the number of variables [11,6]. Thus, learning one subnetwork is NP-hard in $|\mathbf{PC}|^2$, since $|\mathbf{PC}|^2$ is an upper bound for the number of variables in one subnetwork. This means, if $|\mathbf{PC}|$ is much smaller than the number of all variables, the substructure approach dramatically reduces the number of possible network structures. This affects the performance of heuristic search strategies like hill climbing, as well. For an estimation of the impact, we define the cost of a search strategy, depending on the maximum number of parents and children and the size of the domain, as $F(|\mathbf{PC}|, |\mathbf{X}|)$. For one subnetwork, the cost becomes $f(|\mathbf{PC}|, |\mathbf{PC}|^2)$. Thus, the overall cost for the second phase of substructure learning is $|\mathbf{X}|f(|\mathbf{PC}|, |\mathbf{PC}|^2)$. For large networks with small $|\mathbf{PC}|$ we expect the substructure algorithm to perform faster than approaches that learn the complete DAG. If we restrict $|\mathbf{PC}|$ on a fixed value, the second phase performs even linearly in the number of variables.

3 Results

In this section, we empirically benchmark the substructure algorithm by a comparison to MMHC. We use only MMHC for comparison as it has been shown in [16] to outperform many other structure learning algorithms in terms of accuracy and time efficiency. For the benchmark, we sample training data from known benchmark networks and request both algorithms to reconstruct the original network structures. These reconstructed networks are then compared to the original network to assess the quality of the learned structures. As benchmark networks we have chosen the Alarm [3] and the Insurance network [4]. Both networks are relatively small and have only a few variables (Alarm: 37; Insurance: 27). However, we are particularly interested in the performance in large domains.

Thus, we used the tiling method described in [17], which uses one network as tile and puts several tiles together, to enlarge both networks in size. We generated several large networks with the 10-fold, 20-fold and 30-fold size of the original network using the Causal Explorer software package [2]. The resulting networks are denoted as Alarm_10, Alarm_20, Alarm_30, Insurance_10, Insurance_20 and Insurance_30 (or abbreviated: A._10, ..., I._30). From each of the benchmark networks we sampled data sets of different sizes (100, 200, 500, 1000 and 5000 samples).

For the structure learning part of the substructure algorithm we use random hill climbing as heuristic search method. This means we select randomly two variables, calculate the scores for arc addition, arc removal and arc reversal and apply the highest scoring local change until no action can improve the total score. As scoring function that solves (3), we use the Bayesian Dirichlet equivalent (BDeu) score [14] with an equivalent sample size of ten. For the MMHC algorithm we used the implementation of the original authors from the Causal Explorer software package [2]. For the DAG search, they implemented Greedy Hill Climbing and used the BDeu score with an equivalent sample size of ten, as well. The here presented approach is focused on optimally reconstructing the original structure. Thus, we assess the accuracy by using evaluation measures that are based on structural features only. Other quality measures that take the density distribution into account are not considered here. As first evaluation measure we use the structural hamming distance (SHD) which is defined as the number of the following operations to make two PDAGs match [16]: (1) insert or remove an undirected edge or (2) insert, reverse or remove a directed edge. For feature graphs (fPDAGs), we extend the definition in such a way that each operation counts not as one but as the confidence of the corresponding feature. Additionally, we report the number of false positives (FP) and false negatives (FN) defined as the number of operations to remove all false positive or false negative edges. For runtime comparisons we use the real-time of both algorithms in seconds on a computer with an Intel Pentium M processor, 2 GHz, and two GB working memory.

Table 2. Performance results for different networks and sample sizes

	500		1000		5000	
	Runtime	SHD	Runtime	SHD	Runtime	SHD
A.	1.22 (5.43)	1.25 (26.2)	1.23 (7.31)	1.03 (16.4)	1.30 (26.4)	1.85 (18.5)
A._10	**0.64** (162.8)	1.01 (382.4)	**0.73** (228.3)	**0.99** (314.5)	**0.85** (862.1)	1.10 (253.9)
A._20	**0.40** (582.9)	**0.91** (742.8)	**0.45** (802.4)	**0.89** (620.9)	–	–
A._30	**0.24** (1265)	**0.88** (1066)	**0.33** (1741)	**0.85** (867.0)	–	–
I.	1.18 (5.1)	1.00 (42.1)	1.09 (7.66)	**0.90** (36.1)	1.11 (57.2)	**0.92** (34.1)
I._10	**0.78** (129.3)	1.11 (405.0)	**0.88** (199.6)	1.04 (327.1)	**0.98** (1348)	1.08 (201.6)
I._20	**0.54** (398.5)	1.03 (757.0)	**0.65** (598.0)	1.01 (592.3)	–	–
I._30	**0.39** (815.7)	1.02 (1137)	**0.45** (1202)	**0.97** (885.2)	–	–

Table 3. Average performance results

Network	Size	Edges	Runtime	SHD	FP	FN
Alarm	37	46	1.32	1.37	**0.98**	1.31
Alarm_10	370	570	**0.74**	1.21	**0.81**	1.06
Alarm_20*	740	1101	**0.43**	**0.90**	**0.32**	1.09
Alarm_30*	1110	1580	**0.29**	**0.86**	**0.31**	1.08
Insurance	27	52	1.31	1.00	1.04	1.09
Insurance_10	270	556	1.27	1.15	1.40	1.04
Insurance_20*	540	1074	**0.59**	1.02	**0.86**	1.03
Insurance_30*	810	1619	**0.42**	**0.99**	**0.87**	1.01

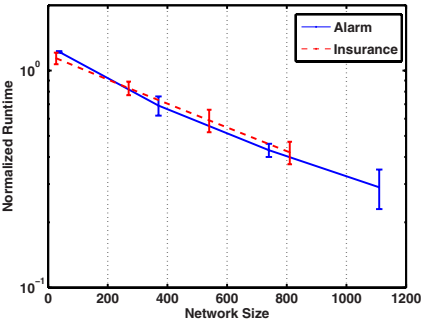

Fig. 2. Speed-up of substructure learning increases with the size of the network

Table 2 shows the relative performance of substructure learning compared to MMHC for different sample sizes (500, 1000 and 5000 samples) and networks by means of runtime (in seconds) and SHD. The numbers denote the normalized performance of substructure learning. This means that we divided each measure for substructure learning by the corresponding measure for MMHC. Thus, a value smaller than one denotes that substructure learning performs better than MMHC. The number in brackets denote the original, unnormalized measures for substructure learning. Additionally, in table 3 we report the network-wise averaged values over all sample sizes (100, 200, 500, 1000 and 5000 samples). Some of the networks (denoted by an asterisk in the table) are only learned with 500 and 1000 samples due to the large amount of time needed for one network reconstruction. As we can see, the substructure algorithm generally shows a good performance in terms of runtime and network quality compared to MMHC, especially for large networks. There is only one prominent outlier: the relatively small Alarm network is reconstructed poorly for 5000 samples with a normalized hamming distance of 1.85. For all other cases, however, the structural hamming distances are comparable for both approaches, in some cases substructure learning even outperforms MMHC. Besides, the number of false

positives are even less in most substructure networks, while there are slightly more false negatives (see table 3).

In Fig. 2, the normalized and averaged runtimes for 500 and 1000 samples are plotted against the size of the network. As MMHC shows better runtime results for small networks, the reduced complexity of substructure learning shows its advantage for larger networks: For the Insurance_30 benchmark case, substructure learning needs only about 40 % of MMHCs runtime, while for the largest Alarm network only about 30 % of the runtime is needed. We also tried to learn larger networks, thus we created a tiled Alarm network with 1850 variables and 2853 egdes. However, MMHC failed to learn the complete network within one day (we interrupted the algorithm because of time issues). In contrast, substructure learning reconstructed the whole network within 255 minutes with a hamming distance of 1378, 88 false positives, 661 false negatives and 1564 correctly identified edges. Thereby, the network learning phase of substructure learning took only 10 minutes, while the skeleton reconstruction phase took the rest.

4 Discussion

Many other approaches for efficient network learning optimize the search procedure to find a good DAG by utilizing the sparseness of the structure. Recently, an algorithm that deals with domains up to hundreds of thousands of variables was introduced [12]. However, it restricts on binary variables paired with very sparsely linked graphs. Another approach that is closely related to MMHC was introduced in [5]. While, in the worst case, the skeleton reconstruction phase using MMPC can have an exponential cost, they developed an polynomial algorithm (called PMMS) for learning the skeleton. Empirically results on benchmark cases have shown that this algorithm significantly improves the runtime with a comparable quality of the reconstructed skeleton. In future we plan to include this algorithm in our substructure learning.

Since substructure learning detects the Markov blanket for each variable and thus renders this variable independent from all other variables given the Markov blanket, it can also be seen as a feature selection algorithm. In [15] a variation of MMPC is developed that estimates the Markov blanket using conditional independency tests. A comparison of different other approaches can be found in [1]. However, these methods return only the set of variables that belong to the Markov blanket, without discovering the probability distribution and its underlying network structure.

Another approach that is somehow related to our work is the framework of dependency networks [13]. There, the joint distribution is defined by a set of conditional probabilities. Unlike BNs where the conditional probability of a variable is defined given its parents, the conditional probability for each variable is determined by the complete Markov blanket. Subnetworks, resulting from substructure learning, can be easily transformed into a dependency network: The conditional probability of variable X_i is given by the joint distribution of subnetwork B_i, conditioned on the Markov blanket of X_i. For inference in

a dependency network, the original authors have introduced a Gibbs-sampling method. Since subnetworks can be transformed into dependency networks, this inference method can also be applied to subnetworks.

5 Conclusion

The problem of learning the best scoring Bayesian network from data is NP-hard. In this paper, we have introduced the substructure algorithm that efficiently estimates the features of the underlying network structure by independently learning small subnetworks. Results from benchmark cases show that structural features of large networks can be learned with high accuracy, comparable to the results of MMHC. However, substructure learning scales much better for large domains, if the network is only sparsely linked. We have also shown that the framework of dependency networks can be utilized to perform inference on subnetworks.

References

1. Aliferis, C.F., Tsamardinos, I., Statnikov, A.: HITON: a novel Markov Blanket algorithm for optimal variable selection. In: AMIA: Annual Symposium proceedings, American Medical Informatics Association (2003)
2. Aliferis, C.F., Tsamardinos, I., Statnikov, A., Brown, L.E.: Causal Explorer: A Causal Probabilistic Network Learning Toolkit for Biomedical Discovery. In: Valafar, F., Valafar, H. (eds.) Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Scienes, METMBS '03, June 2003, pp. 371–376. CSREA Press (2003)
3. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In: Second European Conference on Artificial Intelligence in Medicine, London, Great Britain, vol. 38, pp. 247–256. Springer, Berlin (1989)
4. Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive Probabilistic Networks with Hidden Variables. Machine Learning 29(2-3), 213–244 (1997)
5. Brown, L.E., Tsamardinos, I., Aliferis, C.F.: A Comparison of Novel and State-of-the-Art Polynomial Bayesian Network Learning Algorithms. In: AAAI, pp. 739–745 (2005)
6. Chickering, D.M., Geiger, D., Heckerman, D.: Learning Bayesian Networks is NP-Hard. Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA, USA (November 1994)
7. Chickering, D.M.: A Transformational Characterization of Equivalent Bayesian Network Structures. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 87–98 (1995)
8. Dejori, M.: Inference Modeling of Gene Regulatory Networks. PhD thesis, TU München, Garching, Germany (2005)
9. Friedman, N., Goldszmidt, M., Wyner, A.J.: On the Application of The Bootstrap for Computing Confidence Measures on Features of Induced Bayesian Networks. In: Seventh International Workshop on Artificial Intelligence and Statistics (January 1999)

10. Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using Bayesian networks to analyze expression data. In: RECOMB, pp. 127–135 (2000)
11. Friedman, N., Nachman, I., Pe'er, D.: Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm. In: UAI 99, pp. 206–215 (1999)
12. Goldenberg, A., Moore, A.: Tractable Learning of Large Bayes Net Structures from Sparse Data. In: ICML '04: Proceedings of the twenty-first international conference on Machine Learning, p. 44. ACM Press, New York (2004)
13. Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R., Kadie, C.: Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research 1, 49–75 (2001)
14. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. Machine Learning 20(3), 197–243 (1995)
15. Tsamardinos, I., Aliferis, C.F., Statnikov, A.: Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 673–678. ACM Press, New York (2003)
16. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning 65(1), 31–78 (2006)
17. Tsamardinos, I., Statnikov, A., Brown, L.E., Aliferis, C.F.: Generating Realistic Large Bayesian Networks by Tiling. In: 19th International Florida Artificial Intelligence Research Society (FLAIRS) Conference (May 2006)
18. Verma, T.S., Pearl, J.: Equivalence and synthesis of causal models. In: Bonissone, P.P., Henrion, M., Kanal, L.N., Lemmer, J.F. (eds.) Uncertainty in Artificial Intelligence, pp. 255–268. North Holland, Elsevier Science Publishers B.V, Amsterdam (1991)