Scalable Sparse Bayesian Network Learning for Spatial Applications

Thomas Liebig, Christine Körner and Michael May
Fraunhofer IAIS, Schloss Birlinghoven
53754 Sankt Augustin, Germany
{thomas.liebig, christine.koerner, michael.may}@iais.fraunhofer.de

Abstract

Traffic routes through a street network contain patterns and are no random walks. Such patterns exist for instance along streets or between neighbouring street segments. The extraction of these patterns is a challenging task due to the enormous size of city street networks, the large number of required training data and the unknown distribution of the latter. We apply Bayesian Networks to model the correlations between the locations in space-time trajectories and address the following tasks. We introduce and examine a Bayesian Network Learning algorithm enabling us to handle the complexity and performance requirements of the spatial context. Furthermore, we apply our method to German cities, evaluate the accuracy and analyse the runtime behaviour for different parameter settings.

1 Introduction

Our work is motivated by the planning of poster campaigns. Usually, a predefined number of posters is distributed over a city for a given period of time, with the main goal to reach as many people as often as possible. When planning campaigns of poster advertisements, the question for the selected poster locations arises. More precisely, the number of reached people and their average amount of poster contacts (opportunity to see) result from the dependencies between poster locations.

Locations are dependent, if they often co-occur within the trajectories of people. A trajectory of a person (i.e. a person's movement through geographic space within a certain period of time) is not a random walk through the city, but is made for a specific purpose. For example, consider the daily path of a commuter. Starting at home, it mainly passes a motorway and ends at the place of work. During the trip it is more likely for the commuter to stay on the motorway than to leave it and enter one of the villages along the motorway. Thus, different locations within a region occur not independently within one trajectory, but correlate.

Throughout this paper, we consider the streets of a city or region as domain. The NAVTEQ network [4] consisting of street segments and their neighbourhood relationships provides the discrete traffic network for our work. A trajectory through a city can be represented as sequence of street segments. However, in our application we do not need to consider the sequential order of the passed segments and can thus represent a trajectory as a set. Each set of segments can be represented binary by its characteristic function. We can thus form a vector with one position for each street segment for a town. The function assigns a one, if a segment belongs to a specific trajectory and a zero elsewise. With this representation, arbitrary traffic behaviour induces a multivariate probability distribution on street segments. Our goal is to depict this distribution.

When given a huge number of trajectories (for example in form of GPS-logs), the conditional dependencies between two or more locations can be determined by simply counting co-occurrences within the data. In case of pairwise dependencies between locations, the dependencies can be stored in a square matrix. However, this only generates useful results under the assumption that the current position just depends on the previous one (Markov assumption) [1]. Trajectory data violates this assumption, as the behaviour of a person at a crossroad depends on his or her origin. In order to obtain dependencies between more than two locations the matrix has to be extended into higher dimensions. This means to consider not only pairs of locations, but tuples of various sizes. This approach raises the problem that the matrix soon becomes huge and hard to handle in practice. Furthermore, in order to represent the dependencies of all locations in the matrix, the input data has to contain sufficient trajectories to represent the probability distribution.

Our approach tackles the first of these two problems, by using Bayesian Networks as a compact graphical representation of conditional dependencies. We circumvent the second problem by generating a sufficient artificial set of routes. Bayesian Networks are graphical models which represent a probability distribution of random variables. Variables are modelled by nodes, and edges denote the con-

978-0-7695-3503-6/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDMW.2008.124

ditional probability between the connected variables. The advance is the compact representation of the distribution, the possibility for visual analyses and inclusion of expert knowledge. Being a generative model, Bayesian Networks can also be used to draw samples according to the represented probability distribution. This allows utilization of Bayesian Networks for an iterative sampling of routes according to a known or estimated traffic distribution.

This paper is organised as follows. We give an introduction of probabilities in spatio-temporal spaces in sections 2. We present our structure learning of a Bayesian Network that models the correlations of locations within routes in section 2.2. In sections 3.1 and 3.2 we analyse and interpret the structure learning algorithm with respect to runtime, memory requirements and applicability. We apply the method to a German city in section 4 and conclude with a summary and further prospects.

2 Bayesian Networks

2.1 Basics

A Bayesian Network is a compact representation of the joint probability distribution of a finite set of random variables $\boldsymbol{X} = X_1, X_2, \ldots, X_n$. It consists of a network structure which is a directed acyclic graph (DAG) $G = (\boldsymbol{X}, E \subset \boldsymbol{X} \times \boldsymbol{X})$ and of local probability distributions for all variables $p(X_i = x_i | parents(X_i))$ [10]. The joint probability distribution for $\boldsymbol{X}$ is given by

$$p(\boldsymbol{X} = \boldsymbol{x}) = p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \prod_{i=1}^{n} p(X_i = x_i | parents(X_i))$$

where $parents(X_i)$ is the set of all ancestors X_j having a directed edge in G connecting X_j with X_i, i.e.

$$parents(X_i) := \{X_j | (X_j, X_i) \in E\}.$$

The lack of arcs between random variables denotes (conditional) independence among the variables [8]. Learning the structure of a Bayesian Network must solve two problems. First, the network structure must be learned. Second, the local probability distributions have to be assigned.

Introducing a metric g on the space of possible network structures M, which measures how well the joint probability distribution of $\boldsymbol{X}$ given by a dataset D is represented by an arbitrary DAG $BN \in M$, the structure search becomes an optimisation problem for the best network structure BN^* with respect to the score $BN^* = \arg\max_{BN \in M} g(BN, D)$. This problem has been proven to be NP-complete [2]. The standard approach to avoid searching the complete set M for the optimum, is a greedy search [3] which is applicable up to a few hundred variables. However, in our spatial domain we easily deal with several thousands of variables. For example the city of Hamburg contains about 36,000 street segments. To handle such large amounts of variables, the literature provides specialised algorithms, which restrict the search space due to certain heuristics [5, 7]. In addition, we expect our dataset D to be sparse, as only a small percentage of all available edges E co-occurs in one path.

2.2 State of the Art

Two approaches exist that either bound the search space by limiting the degree of vertices within the network or by limiting the set of possible edges: the Sparse Candidate algorithm [5] and the Screen Based network search [7] respectively.

The Sparse Candidate algorithm [5] limits the search space by bounding for each variable in the network the number of possible parents (the parent candidate set). It identifies the best parent candidate set for each variable using two heuristics: one for selection of appropriate parent variables and a second to evaluate the size of the parent candidate sets. Beginning with a minimal parent set size of two, the algorithm stepwise increases the number of possible parents until the stop criteria of the second heuristic applies. The advantage of this algorithm is the strongly reduced Bayesian Network structure search space. But, as a main challenge of an approach like this, Friedman, Nachman and Peér mention the ability to represent higher order XOR relations or so called mutually independence relations. If more variables than the maximal cardinality of the parent set are concerned in such a negative correlation, there is no chance for the algorithm to discover this relation during structure learning. This algorithm does also not utilise the sparseness of the given dataset.

Another state of the art structure learning algorithm is the Screen Based Network Search algorithm [7], which bounds the search space through pre-sampling the allowed edges of the Bayesian Network. It uses the sparseness within the data by processing frequent sets of random variables. The Bayesian Network learning task for a large number of random variables is divided into smaller learning tasks that are easy to handle. This requires the combination of those Bayesian Networks, encoding a partial probability distributions, to a valid Bayesian Network. The first step is called screening step. There it builds for each high frequent variable set a corresponding local Bayesian Network. Providing the most important information about the probability distribution, it is reasonable to only consider the variable sets with high frequency. The edges of these small Bayesian Networks enclosed with a score (for example the frequency of the variable set, but also other heuristics are described in [6]) are accumulated in a stack (called edgedump). This becomes the main input for the second step, the search for an all-embracing Bayesian Network. One by one edges are drawn from the top of the edgedump and are included in the resulting Bayesian Network in case that the score of the graphical model increases and its graph stays without cycles. In an optional last step, which models negative correlations, high frequent variable pairs are investigated for mutual independence (within the dataset) and additional edges are added to represent their negative correlations. Transferring this algorithm to a spatial probability space defined for an arbitrary city might cause undesirable memory or time requirements. As described in section 1 each street segment of the traffic network becomes a random variable and therefore a possible item during the frequent itemset enumeration. The number of frequent sets increases drastically by increasing the maximum frequent set lengths. One solution to reduce the number of frequent sets, is to increase the frequency threshold t, used for the decision whether a set of variables is highly frequent or not, such that we can handle the number of resulting frequent sets. But this step requires additional knowledge about the distribution, and the resulting Bayesian Network hardly describes the dataset as only a few variables get the chance to be connected.

Due to memory and time complexity, both algorithms were unable to find satisfying solutions or gave no solution at all in our application. We present a new approach (in section 3) that will bound both, the edgeset E and the vertex degree and therefore reduces the influence of the number of variables n and their joint probability distribution on the estimated time and memory requirements (see section 3.1). This is a strong condition in our spatial context, as only this independence guarantees the applicability of an algorithm to arbitrary large city sizes with unknown traffic behaviour as shown in the next sections.

3 Scalable Sparse Bayesian Network Structure Learning Algorithm

Our new approach (algorithm 1) bases on a heuristic that combines the ideas of both algorithms (bounding the edgeset and the number of possible ancestors). This is done in a two-step algorithm: First, we pre-sample within each route ω a set of maximal k distinct edges uniformly distributed. Afterwards, we apply a Screen Based Network Search like algorithm to the dataset as follows. We enumerate frequent variable sets on this pre-sampled data with threshold t and maximal length ml. This results in an adjustable bounded number of subsets. For each of these sets we determine in a second step a Bayesian Network that fits the original data best and collect their edges on a stack. We may order this stack by the score of the local networks or by the frequency in the local Bayesian Networks [6]. In a third step, we draw edges from the ordered stack and build a global Bayesian Network by adding the edge in any direction if it does not create any cycle in the network and increases the score. Afterwards, we scan the original dataset and recompute the common probability tables for each vertex in the Bayesian Network. This new Baysian Network learning algorithm features pre-sampling to transform an arbitrary dataset to a processable one with adjustable size and density. Furthermore, the pre-sampling is reasonable as it does not destroy too much information given by the set of routes. The information about the most significant dependencies, represented by highly frequent variable sets, remains in the input data, because the corresponding variable sets keep a high frequency.

Algorithm 1 SCALABLE SPARSE BAYESIAN NETWORK LEARNING

```
Require:  $D$       , complete dataset
          $k$       , maximal frequent set size
          $ml$      , frequent set length
          $t$       , support threshold
          $g(\cdot)$ , Bayesian Network score
Ensure:   $BN$      , a Bayesian Network
 1: for all observations $\omega \in D$ do
 2:     $\omega'$ := sample $k$ locations from $\omega$
 3:     add $\omega'$ to $D'$
 4: end for
 5: $FS$ := enumerate frequent sets ($D'$,$t$,$ml$)
 6: for all $fs \in FS$ do
 7:     $BN^* = \arg\max_{BN on fs} g(BN, D)$
 8:     add edges of $BN^*$ to $edgedump$ or if already in
        $edgedump$ increase their score
 9: end for
10: sort $edgedump$ decreasing
11: for all $edge \in edgedump$ do
12:     if $BN \cup edge$ contains no cycle then
13:         if $g(BN \cup edge) > g(BN)$ then
14:             add $edge$ to $BN$
15:         end if
16:     end if
17: end for
18: return $BN$
```

3.1 Analysis

The inclusion of the pre-sampling step to the Screen Based Network Search bounds the memory requirement of the network search independently from the observed probability distribution and reduces the influence of the number of random variables on the running time.

To prove the memory boundedness we have to prove it for the main memory consumers. These are the list of frequent sets and the edgedump. In general an expectation for the number of frequent sets of length ml within a dataset that contains n random variables and m observations is given by $\frac{n}{avglen}\binom{avglen}{ml}$ where $avglen$ is the average length of the observed paths. The pre-sampling upper bounds $avglen$ to k. Therefore the number of frequent sets is lower than $\frac{n}{avglen}\binom{k}{ml}$. To store these sets, we need $O(\frac{n\cdot ml}{avglen}\binom{k}{ml})$. To estimate the edgedump size we have to know the number of possible edges. A directed acyclic graph with n vertices may contain at most $1/2 \cdot n(n-1)$ edges. Thus, the edgedump may only contain $ml \cdot \binom{k}{ml} 1/2 \cdot ml(ml-1)$ edges. Of course, this is a rough approximation because edges co-occur in the frequent sets and the required memory becomes much smaller using an appropriate data structure for the edgedump. This worst case assumption results in case of $n \gg avglen \gg ml$ in a total memory requirement in $O(n \cdot \binom{k}{ml})$. The required time decomposes in four steps: frequent set enumeration, Bayesian Network screening, sorting the edgedump and testing for cycles. The frequent set enumeration requires $O((t_d + ml_d)(n + m + 2^{t_d+ml_d}))$ with $t_d := n - t$ and $ml_d := m - k$ [9]. Bayesian Network learning is exponential and requires every time $O(2^{ml})$. This results in a total requirement of $O(n \cdot 2^{ml}\binom{k}{ml})$. Sorting n elements is in $O(n(\log n))$. Given the maximal edgedump size this leads to $O(\binom{k}{ml} \cdot n \log n)$. Detecting cycles is in order of the numbers of ancestors of the nodes. Worst case this equals to the number of edges, so as a rough bound detecting the cycles is in $O((n \cdot \binom{k}{ml})^2)$. The sum leads to an overall time requirement in $O((n+m)2^{n+m} + n^2 + n\log n)$.

We expect the resulting network structure to simplify the dependencies, as it becomes more unlikely for a relation to be recognised with increasing number of affected variables. But in case of sparse data, the most significant dependencies are the most frequent ones, and we can be sure to represent them.

Our main result that makes our Scalable Sparse Bayesian Network learning algorithm applicable to arbitrary cities and spatial regions, is the linear dependency of the memory requirement on dimensions n and m of the input data D, and a further speed up of the structure learning process compared to Screen Based Network Search.

3.2 Properties of Spatial Bayesian Networks

The interpretation of the resulting network structure may easily lead to the assumption that an edge from X_i to X_j denotes the probability of a path that first passes X_i to pass also X_j. However, our model deals with trajectories as sets of locations, omitting their temporal order (see section 1). Therefore, the existence of the given edge only containes the information that knowing whether a route passes X_i we may have an assumption for also passing X_j. In general, the direction of the edges does not represent causality.

When introducing the traffic network and routes, we did not require connectivity of either. Now, in the resulting Bayesian Network previously separable components, contained in the traffic network, stay not necessarily separable in the Bayesian Network. The reason is that routes may contain locations of multiple components (connected by streets outside of the city).

We use this property in our application. When generating our dataset, we do not only allow paths through the possibly unconnected street network G of a city, but through a sufficiently large network that includes all connected paths minimizing the cost $c(s,g)$ from any start s to any goal g within the city network $cl_{c(s,g)}(G)$. For the subsequent Bayesian structure learning we consider just the important part of the routes that passes the city. Thus, routes may become unconnected. Nevertheless, the Bayesian Network may recognise the dependencies between these locations.

The resulting spatial network also has another interesting property. Due to the independence relations of the source nodes within the DAG, we know that an arbitrary path through G passes one of the source locations. For a set of nodes, containing all sources or all of its descendants, the same property holds. Thus, the Bayesian Network represents dependencies and conditional independencies between locations graphically by its topological structure (see [1] and [10]).

4 Applicability

In this section, we demonstrate the applicability of our algorithm for the city *Brandenburg an der Havel*. The traffic network of this town contains about 4,200 street segments. We use a route generator that minimises travel time between start and destination of each route, to sample training datasets of different sizes (500, 1,000 and 200,000 routes). In this case these routes have a maximum length of 200 segments. The routes are encoded binary within a matrix. Per route there might be at most 200 ones and at least 4,000 zeros. Therefore, we call the dataset sparse. Our three datasets vary not only in the number of routes, but also in the coverage of the traffic network. Uncovered locations, will result in Bayesian Network nodes with degree zero after the stucture learning. During structure learning, we apply the scoring function BDeu [8] to evaluate the Bayesian Network.

One benefit of Bayesian Networks is their ability to create samples according to the represented distribution. In our application, we exert sample generation not only to generate trajectories accoring to the represented dependencies, but also as a method to validate the accuracy of our Bayesian Network. We applied Ancestral Sampling [1]. This algorithm first orders the random variables of the Bayesian Network topologically. Afterwards, it processes the variables in this order and samples their states according to the states sampled in previous steps. This is possible, due to the fact that the parents of each variable appear earlier within the topological ordering than the variable itself. With increasing number of samples, their distribution converges to the distribution represented by the Bayesian Network [1]. In the optimal case, the network represents the original distribution of the given dataset D. Therefore, we may recognise errors of the learning algorithm when validating the generated samples with the training set. We compare the relative frequency $p(e)$ for edges e of the training dataset with the set of samples drawn according to the Bayesian Network. Figure 1 displays the error distribution for the complete set of locations of a region. In addition it shows the error separated in three frequency classes: low $p(e) < 0.015$, medium $0.015 < p(e) < 0.1$ and high. The separation illustrates the comparably high quantity of low frequent segments and their small variance. The few errors, are caused by the inaccurate representation of mutual independence relations denoting dependencies as, for example, when a route passes a certain set of locations it unlikely passes another set of locations. We successfully applied this algorithm also to other cities, including Hamburg that contains about 36,000 street segments. This could not be achieved with [7] and [5] on our machine due to their variable memory requirements.

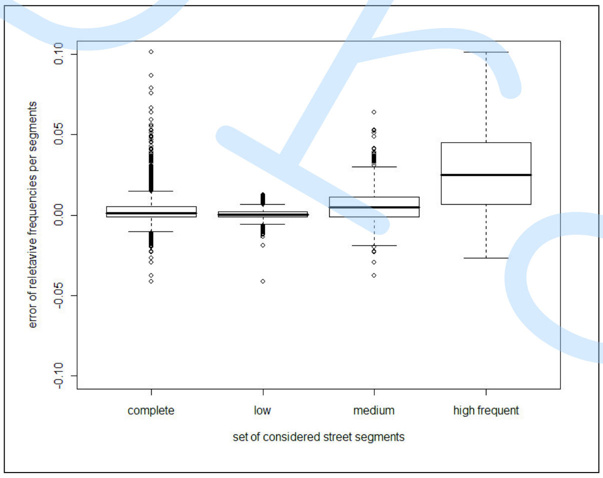

Figure 1. Difference of relative frequency of locations in the dataset and in sets of samples

5 Evaluation

As none of the other presented Bayesian Network learning algorithms returned a network structure for our spatial domain, our validation focuses on comparison with other baseline models that model individual mobility. These are: drawing random segments from the street segments without regard for connectivity, and Hidden Markov Models [11]. Being generative models both approaches return a set of segments. We perform our experiment as follows. For a town (in this case we use the German city Rodgau) we create 300,000 artificial routes and build a Bayesian Network to represent their probability distribution. We also use these trajectories to learn a Hidden Markov Model (HMM). Like Bayesian Networks, Hidden Markov Models are graphical models. The main difference is that Markov Models do not represent a probability distribution but a stochastic process, a so called Markov process. A system is assumed to act in discrete states. A first order Hidden Markov Model (as we will use here) keeps the transition probabilities between consecutive states within a transition matrix. This implies that each state only depends on its ancestor and not on the whole state history. Such Markov Models may be used to sample Markov chains of arbitrary length. In our case a state is defined as being at a certain segment, this is similiar to the random variable definition in section 1.

The other baseline model is the simplest generative model that creates sets of street segments. It utilises the street network as an urn to draw segment sets from. We compare also this method (weighted random sampling) to our Bayesian Network based algorithm. In this case, the similarity of both models is that each does not guarantee the resulting sample to be connected but to converge to the underlying frequency distribution of the original data. Both baseline methods request for a certain sample length. We draw this value according to the distribution specified by the routeset.

All three models are used generatively to create a set of 10,000 trajectory samples. Afterwards we draw 500 pairs of locations $(segment_1, segment_2)$ from the street network and compare their conditional probability $p(segment_1|segment_2)$ with the probability of the original routes. We repeat the drawing 10,000 times to avoid random effects. One criterion for comparison is the correlation between the predicted probabilities and the original probability. In our experiment the mean correlations were 75% for our Spatial Bayesian Networks, 71% for Hidden Markov Models and 47% for weighted random sampling. Figure 2 shows the complete correlation distribution over all 10,000 sets of location pairs. The high correlation of the Hidden Markov Models is misguiding, because in most cases the Markov chain prediction equals zero. Due to the relative small original conditional probabilities it correlates well, though. Figure 3 displays the mean deviation between predicted and original probabilities. It clearly visualises the negative bias of the HMM. To summarise, the use of Spatial Bayesian Networks to represent traffic behaviour results in a

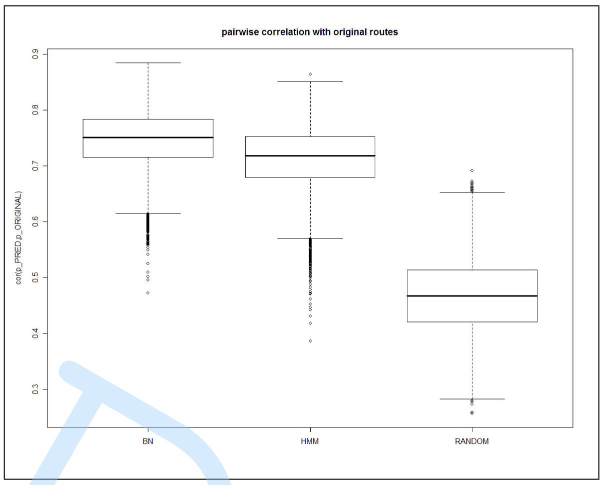

Figure 2. Distribution of correlation for 10,000 sets of 500 location pairs

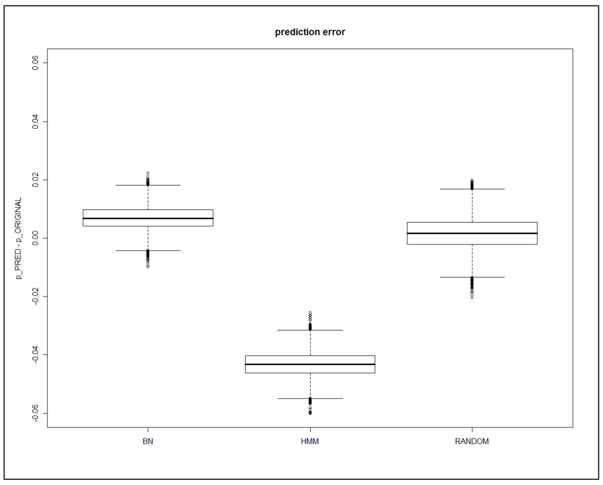

Figure 3. Distribution of prediction error

more accurate description than the two baseline approaches.

6 Conclusion

This work shows that recent advances in Bayesian Network learning increase the applicability of those to challenging real-world domains. We introduce a new method for Bayesian structure learning in presence of sparse data that combines the features of state of the art algorithms (Sparse Candidate algorithm [5] and Screen Based Network Search [7]) by limiting the size of the maximum frequent sets in the data. This enables the usage of Bayesian Networks also in spatial applications that have serious demands on the structure learning method. Unknown traffic behaviour (which leads to unpredictable probability distributions) and arbitrary city sizes (resulting in many random variables and a large dataset) are no obstacles for the structure learning method we presented, neither in time nor space requirements.

Bayesian Networks enrich a street map with additional information about the traffic behaviour within a city. In spatial applications, Bayesian Networks provide new possibilities compared to other representations of the conditional probabilities among the locations. For example, a Bayesian Network offers a graphical representation of independence relations [10] among poster locations.

In future work, we will investigate the usage of Spatial Bayesian Networks to plan poster advertisement campaigns.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[2] D. M. Chickering. Learning Bayesian networks is NP-Complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

[3] G. F. Cooper and E. Herskovits. A bayesian method for constructing bayesian belief networks from databases. In *Proceedings of the 7th Annual Conference on Uncertainty in Artificial Intelligence (UAI'91)*, pages 86–94, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[4] DDS Digital Data Services GmbH. NAVTEQ atlas. http://www.dds.ptv.de, 2006.

[5] N. Friedman, I. Nachman, and D. Peér. Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 206–215. Morgan Kaufmann, 1999.

[6] A. Goldenberg. Scalable Graphical Models for Social Networks. Technical Report CMU-ML-07-109, Carnegie Mellon Universtity, May 2007.

[7] A. Goldenberg and A. W. Moore. Tractable Learning of Large Bayes Net Structures from Sparse Data. In *Proceedings of the twenty-first International Conference on Machine learning (ICML'04)*, pages 44–52. ACM Press, 2004.

[8] D. Heckerman. A Tutorial on Learning With Bayesian Networks. Technical report, Microsoft Research, March 1995.

[9] R. C. M. Hamilton and T. Wareham. The Parameterized Complexity of Enumeration Frequent Itemsets. In *Proceedings of the International Workshop on Parameterized and Exact Computation (IWPEC '06)*, pages 227–238. Springer, 2006.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[11] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, Jan 1986.