Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Learning Bayesian Networks with Bounded Tree-Width via Guided Search

Siqi Nie
Rensselaer Polytechnic Institute
Troy NY, USA

Cassio P. de Campos
Queen's University Belfast
Belfast, UK

Qiang Ji
Rensselaer Polytechnic Institute
Troy NY, USA

Abstract

Bounding the tree-width of a Bayesian network can reduce the chance of overfitting, and allows exact inference to be performed efficiently. Several existing algorithms tackle the problem of learning bounded tree-width Bayesian networks by learning from k-trees as super-structures, but they do not scale to large domains and/or large tree-width. We propose a guided search algorithm to find k-trees with maximum Informative scores, which is a measure of quality for the k-tree in yielding good Bayesian networks. The algorithm achieves close to optimal performance compared to exact solutions in small domains, and can discover better networks than existing approximate methods can in large domains. It also provides an optimal elimination order of variables that guarantees small complexity for later runs of exact inference. Comparisons with well-known approaches in terms of learning and inference accuracy illustrate its capabilities.

Introduction

Bayesian networks (BNs) compactly represent the joint probability distribution for a multivariate domain by using a directed acyclic graph (DAG) to encode conditional independences. Learning the graph (a.k.a. structure) of a BN is a very challenging task. With the goal of decision making or probabilistic explanation, it is desirable to learn models that generalize well and at the same time have low inferential complexity. One attempt that has received growing attention recently is to learn a BN with bounded tree-width. By imposing a hard constraint on the tree-width of a BN, selecting overly complicated structures of dependences is avoided, thereby the chance of overfitting is reduced. This has been supported by empirical results (Elidan and Gould 2008). Moreover, a BN with small tree-width allows exact inferences to be performed efficiently (Kwisthout, Bodlaender, and van der Gaag 2010). This paper presents a new approach for score-based Bayesian network structure learning with a hard constraint on the tree-width of the yielded network. We show that such approach outperforms the state of the art, and can be applied to numerous problems. We employ it on the *UAI'2014 Inference Competition* and achieve promising results.

Even though research on BN structure learning has been very active (Barlett and Cussens 2013; Cussens et al. 2013; de Campos and Ji 2011; Hemmecke, Lindner, and Studený 2012; Jaakkola et al. 2010; Yuan and Malone 2013), few algorithms have been designed for learning with bounded tree-width. Elidan and Gould (2008) designed an approximate algorithm by combining several heuristics to compute the tree-width and to learn the structure of BNs. Very recently, this topic has attracted great attention. Korhonen and Parviainen (2013) proposed a dynamic programming algorithm for learning n-node BNs of tree-width at most k with time complexity $O(3^n n^{k+O(1)})$. In practice, such approach is quite slow for networks with more than 15 nodes or for tree-width bound greater than 3. Parviainen, Farahani, and Lagergren (2014) employed an integer programming approach to solve the problem. To avoid exponentially many constraints in the number of variables, it iteratively creates a cutting plane on the current incumbent solution. Berg, Järvisalo, and Malone (2014) translated the problem into a weighted maximum satisfiability problem and solved it by weighted MAX-SAT solvers. These algorithms work only with networks of at most 30 variables. Nie et al. (2014) proposed two algorithms, one based on integer programming that focuses on finding the exact solution for small domains and one based on sampling k-trees with the goal of addressing large domains. One shortcoming of the latter approach is that the sampling space for k-trees is huge and can be barely explored by the sampling technique in a reasonable amount of time.

We develop an approximate algorithm that is able to deal with domains with more than 100 variables (well beyond current methods) and still produce reasonably accurate results. We achieve significant improvements on large domains while maintaining close to optimal performance on small ones. The strategy we employ is to search for promising k-trees, which are the maximal undirected graphs of tree-width k, and then discover the optimal BN whose moral graph is a subgraph of it. By taking advantage of some underlying ideas behind the procedure to construct a k-tree, a search algorithm with pruning is introduced to find the k-tree from a root clique with maximum *Informative Score* (Nie, de Campos, and Ji 2015), which is a heuristic measure for a k-tree about the quality of the BNs that are "contained" in it. Therefore, we avoid the high complexity of an

exact method (Korhonen and Parviainen 2013) and reduce the need of so many samples over this huge space of k-trees (Nie et al. 2014), as we target *quality* of k-trees instead of *quantity*. We will show that performance can be improved significantly in both accuracy and efficiency by the new approach.

Preliminaries

Let $X = \{X_i\}_{i=1}^n$ be categorical random variables. The structure of a Bayesian network $G = (V, E)$ contains vertices V corresponding to variables X and edges E representing direct dependencies between variables. Denote X_{pa_i} as the parent set of variable X_i, $i \in N = \{1, 2, \ldots, n\}$. Conditional probability tables $P(X_i|X_{pa_i})$ are given accordingly. A Bayesian network defines a joint probability distribution over X by the expression: $P(X_1, \ldots, X_n) = \prod_{i=1}^n P(X_i|X_{pa_i})$. The structure learning task of a BN is to identify the best DAG from data. In this paper we consider the score-based structure learning problem, in which a score $s(G)$ is assigned to each DAG G.

The commonly used score functions, such as BIC (Schwarz 1978) and BDeu (Buntine 1991; Cooper and Herskovits 1992; Heckerman, Geiger, and Chickering 1995), are decomposable, that is, the overall score can be written as the summation of local score functions, $s(G) = \sum_{i \in N} s_i(X_{pa_i})$. For each variable, its score is a function only of its parent set. We assume that local scores have been computed and pruned in advance (de Campos and Ji 2010; 2011) and can be retrieved in constant time. Typical scores try to maximize data log-likelihood while applying a penalty or regularization to avoid overfitting. However, this regularization avoids only local overfitting (naturally constraining the number of parents of a node), but even a graph with small in-degree can be complex in the inference stage due to a possibly large tree-width, which is not captured by usual scores.

In graph theory, the *width* of a tree decomposition of a undirected graph is the size of its largest clique minus one. The *tree-width* of an undirected graph is the minimum *width* among all possible tree decompositions of the graph. Define *tree-width* $tw(G)$ of a DAG G as the tree-width of its moral graph, which is obtained by connecting nodes with common child, and by making all edges undirected. It has been proven that the complexity of exact inference on BNs is related to its tree-width exponentially (Murphy. 2012). It is necessary to learn graphs with bounded tree-width to achieve inferential efficiency (Kwisthout, Bodlaender, and van der Gaag 2010). However even the sole computation of the tree-width of a graph is intractable (Arnborg, Corneil, and Proskurowski 1987). One way of imposing the tree-width constraint is using the so-called k-trees.

Definition 1. *The family of k-trees is defined inductively as follows.*

1. A $(k+1)$-clique is a k-tree.
2. If $G = (V, E)$ is a k-tree and $C \subseteq V$ is a k-clique, then the graph obtained by adding a new vertex v and an edge u–v for each $u \in C$ is a k-tree.

A k-clique is a set of k nodes in which every two nodes are connected to each other. k-trees are the maximal graphs with tree-width k, which means no more edges can be added to them without increasing the tree-width (see (Patil. 1986) for details). Therefore, every graph with tree-width at most k is a subgraph of a k-tree. Learning a BN whose moral graph is a subgraph of a k-tree automatically satisfies the tree-width constraint for the learned network (and every BN of tree-width k has its moral graph as a subgraph of a k-tree).

We denote by $\mathcal{T}_{n,k}$ the set of all k-trees over n nodes. One could search directly on this space (Nie et al. 2014), but the total number k-trees is in the order of $e^{n \log(nk)}$, which is extremely large for any network with more than a few tens of nodes. Therefore, the number of k-trees explored in a given limit of time is too small and there is no guarantee about their quality to produce good BNs.

Guided Search for k-trees

Due to the complexity of exactly learning a BN from a k-tree (which is linear in n but factorial in k) and the huge space of k-trees, one cannot expect to exploit too many k-trees. To avoid learning from every k-tree without distinction, Nie, de Campos, and Ji (2015) proposed the so called *Informative Score* (or I-score) function to evaluate how well a k-tree fits the data. The I-score of a k-tree T_k can be roughly defined as $I(T_k) = \sum_{e_{ij} \in T_k} I_{ij}$, where e_{ij} denotes the edge connecting node i and j, and I_{ij} is the empirical mutual information of variables i and j. Since mutual information is equivalent to the likelihood score (Koller and Friedman 2009), a high I-score for a k-tree will suggest that the best BN whose moral graph is a subgraph of the k-tree will have a high score itself. Computing $I(T_k)$ requires only mutual information of pairs of nodes with complexity at most $O(n^2)$ over all multiple runs of the algorithm (as the mutual information values can be stored for later usage). The goodness of the I-score is discussed in (Nie, de Campos, and Ji 2015). We perform some empirical study ourselves to confirm such hypothesis[1].

Now we introduce the formulation of the search algorithm for good k-trees. The goal is to find a k-tree with the highest I-score. We formulate this problem into a *shortest path finding* problem. According to Definition 1, let a state (of this path search space) containing the initial $(k+1)$-clique be the initial state and that with all variables be the goal state. The nodes and edges are added one by one into the existing graph until the goal state is reached. Define the cost of any given graph as its negative I-score. So moving from a state to another has the cost of all nodes/edges that are introduced for such state change. Therefore, the objective is to find a path from the initial state to the goal state with the lowest cost.

Different from typical path-finding problems, each time we add a variable to the existing graph we need also to choose a k-clique in the current graph, which will be the one connected to the new variable (this is the recursive procedure to build k-trees in Definition 1). Therefore, in each step, we choose a successor variable as well as a k-clique in the current graph to represent a change in the state of our search. The proposed algorithm is a modification of the A* search algorithm, which has been used for BN structure learning

[1] http://homepages.rpi.edu/~nies/nie2016aaai_supp.pdf

(Yuan, Malone, and Wu 2011). In this work A* search is applied over a tree.

Let C_j denote a clique of size k. The initial state has the form of $U_1 = [(X_{j1}, C_1)]$ if we represent the $(k+1)$-clique into a pair containing an arbitrary node X_{j1} in the $(k+1)$-clique and the remaining k-clique. The ith state has the form

$$U_i = [(X_{j1}, C_1); (X_{j2}, C_2); \ldots; (X_{ji}, C_i)], \quad 1 \leq i \leq n\,.$$

where C_i is a clique selected from the node set $\{C_1, X_{j1}, X_{j2}, \ldots, X_{j,i-1}\}$. The goal state U_n contains n pairs of X and C. Note that each state defines a graph.

To tackle this path-finding problem, we employ a simple breadth-first search with pruning, which uses a knowledge-plus-heuristic cost function to determine the order in which the search visits the states. The cost function $f(U)$ for node U is a sum of two functions $f(U) = g(U) + h(U)$: (1) The past path-cost function $g(U)$ for the current graph G, which is the cost from the initial state to the current state U. (2) A future path-cost function $h(U)$ to estimate the cost from current state U to the goal state.

The algorithm uses an *open* list to store the search frontier, and a *closed* list to store the explored nodes. In each step, the state with the smallest f cost from the open list is selected as the successor state, until the goal state is reached.

The definition of the g function is straightforward:

$$g(U) := -I(G) = \sum_{e_{ij} \in G} -I_{ij}\,,$$

which is the negative I-score for the current structure G. The algorithm requires the h function to be *admissible*. The function h is called *admissible* if the estimated cost will never be greater than the true cost to the goal state. Consider the following heuristic function,

$$h(U) := -\sum_{X_i \in V \backslash U} \sum_{j \in V_{ik}} I_{ij}\,,$$

where the set $V_{ik}, |V_{ik}| = k$ is defined as the set of variables that has the largest k mutual information values with variable X_i, and $V \backslash U$ denotes the set of all unexplored variables. The h function is the summation of the k largest (negative) mutual information values associated with all the unexplored variables.

Theorem 1. *h is admissible.*

Proof. For each unexplored variable X_u, the heuristic function h includes the largest k mutual information values associated with X_u. If the optimal k-tree is built, the mutual information corresponding to the edges connecting X_u cannot exceed the largest k numbers. Since the value of mutual information is nonnegative, h is a lower bound to the true cost, and thus admissible. ■

The h function has another property called *consistent*. A heuristic function h is *consistent* if for any state U and its successor W, $h(U) \leq h(W) + c(U, W)$, where $c(U, W)$ is the cost of the edges added in W. If the heuristic function h is consistent, the f function is monotonically non-decreasing on any path, and then the algorithm is guaranteed to find the optimal path as long as the goal state is reached. The consistent heuristic function allows to discard any duplicated entries in the closed set and makes the search more efficient.

Theorem 2. *h is consistent.*

Proof. Assume node X_s is added into the graph when the path travels from state U to W. Then $h(U) - h(W) = -\sum_{X_i \in V_{sk}} I_{is}$, which is the largest k mutual information values associated with variable X_s. This is obviously less than or equal to $c(U, W)$, which includes mutual information values associated with (arbitrary) k edges. Thus $h(U) \leq h(W) + c(U, W)$ and h is consistent. ■

With an admissible and consistent h function, each move the search algorithm makes is optimal, until the goal state is reached. Given the g and the h functions, the algorithm for k-trees is summarized in Algorithm 1.

Algorithm 1 Find the optimal k-tree given an initial $(k+1)$-clique.

Input Mutual information I_{ij}, initial $(k+1)$-clique R;
Output a k-tree with highest I-score.

```
openset = {R}
closedset = empty
while openset is not empty, do
    Let U* be the state in openset with lowest f score.
    X_i* is the successor variable and C is the k-clique in
    the current graph to be connected to X_i*.
    if all variables are included, then
        Return the structure.
    end if
    Remove all the states containing X_i* from openset,
    add them to closedset.
    for each possible successor variable X_i, do
        for each k-clique C in the current graph, do
            Add {X_i, C} to openset.
        end for
    end for
end while
```

Given a k-tree, we can employ either an exact method (Korhonen and Parviainen 2013) or an approximate method (Nie et al. 2014) to learn a BN whose moral graph is a subgraph of the k-tree and maximizes the score function. We present Algorithm 2 for learning BNs of tree-width at most k. Algorithm 2 is an anytime algorithm with a time limit and iteration limit. When the limit is reached, the currently best solution is returned.

There are three steps in Algorithm 2: 1) Identify an initial clique; 2) Find the optimal k-tree from the clique; 3) Find a BN corresponding to the k-tree. Step 2 and 3 both have theoretical guarantees. The only uncertainty is in step 1. To minimize the effect of the randomness, the initial clique is selected as the k+1 variables with the largest I-score, similar to (Chow and Liu 1968), which selects an edge with largest mutual information as initialization. Given a time limit, it is also applicable to randomly sample a clique from all possible ones besides the clique with the highest I-score. For a network with n variables, the total number of possible $(k+1)$-cliques is $|C_{k+1}| = \binom{n}{k+1}$. Compared to the number of all possible k-trees, $|\mathcal{T}_{n,k}| = \binom{n}{k}(k(n-k)+1)^{n-k-2}$, the sample space is significantly reduced. Moreover, since

Algorithm 2 Learning a Bayesian network structure of bounded tree-width.

Input score function s_i, $\forall i \in N$, mutual information I_{ij}, $\forall i, j \in N$;
Output a DAG G^*.

```
Initialize G_i^* as an empty set for all i ∈ N;
while time or iteration limit is not reached, do
    Identify an initial (k+1)-clique;
    Find the optimal k-tree T_k from the initial clique using Algorithm 1;
    Find a DAG G that maximizes the score function and is consistent with T_k;
    if Σ_{i∈N} s_i(G_i) > Σ_{i∈N} s_i(G_i^*) then
        Update G_i^*, ∀i ∈ N.
    end if
end while
```

a clique covering a great amount of mutual information is likely to be contained in different k-trees, the initial clique does not have much impact on the k-trees found by the algorithm. This is also empirically confirmed. The worst case in Step 2 occurs when every k nodes in the existing graph can form a clique, therefore the worst case complexity is $\sum_{m=k+1}^{n-1} \binom{m}{k}(n-m)$.

Algorithm 2 automatically provides an optimal elimination ordering with *induced width* k, which is the size of the largest clique in the tree decomposition minus one. By converting the k-tree into a clique tree, we just need to eliminate the nodes from the leaves until the root. A node is eliminated when it does not appear in the parent (parent meaning the next clique in the direction of the root clique). This can be used later by inference methods on the learned network, as we will evaluate later on.

Experiments

Bayesian Network Scores

We start the experiments by comparing the BDeu scores of structures learned by the proposed method against scores from the state-of-the-art approaches. We use a collection of data sets from the UCI repository (Bache and Lichman 2013) of varying dimensionalities[2]. The numbers of samples vary from 100 to 20,000. Non-binary variables are binarized over the median value. In all experiments, we maximize the Bayesian Dirichlet equivalent uniform (BDeu) score with equivalent sample size equal to one (Heckerman, Geiger, and Chickering 1995).

As state-of-the-art methods, we compare the proposed method against five existing methods, namely the K&P algorithm[3] (Korhonen and Parviainen 2013) (results are omitted as they are always inferior to MILP in our experiments), the MILP algorithm (Nie et al. 2014), the TWILP algorithm[4] (Parviainen, Farahani, and Lagergren. 2014), the S2 algorithm (Nie et al. 2014) and the GOBNILP algorithm[5] (Barlett and Cussens 2013). Note that the GOBNILP algorithm is an exact algorithm based on integer programming that does not impose any constraint on the tree-width, therefore it can be considered as a baseline to measure how much loss in score is brought by bounding the tree-width. We denote the proposed algorithm as K&P+ and S2+, depending on the method we use to learn a BN from a k-tree (following the notation in (Nie et al. 2014)). K&P+ learns an optimal BN from a k-tree exactly using the K&P algorithm, and S2+ uses a second-stage sampling method to learn a BN from a k-tree using the S2 algorithm.

The implementation of the proposed algorithm is in Matlab on a desktop with 16GB of memory. It has been given ten minutes of running time, same that was given to all other approximate algorithms. The exact methods (MILP and GOBNILP) are given three hours of running time. The iteration limit is set to 100 with different initial cliques. The BDeu scores achieved by different methods are given in Table 1. The K&P method can only solve the smallest 4 networks with the same results as other exact methods. For small data sets with less than 30 nodes, even though the proposed algorithm is not guaranteed to return the optimal structure as the exact algorithms are, it can always find a structure that is very close to optimal in terms of scores. For the *letter*, *mushroom* and *wdbc* data set, it even found a structure better than TWILP algorithm. This is because within the time limit, the exact algorithm was not able to finish and only returned a currently best solution. Compared to the GOBNILP method which has no constraint on the tree-width, bounding the tree-width does not seem to lose too much in accuracy, while at the same time a small tree-width guarantees low complexity for exact inference (that is, in many cases the achieved network has similar BDeu to that obtained by GOBNILP without tree-width constraint but it ensures that later inferences with the network will run efficiently, which is a very important property for the practical use of the networks).

As for larger data sets, the proposed algorithm is superior to all competing methods. For the largest two data sets (*hill* and *community*), there are nearly 30% and 10% improvement over the plain S2 algorithm. All the exact methods failed due to excessive memory consumption. In all the data sets, S2+ outperformed K&P+, and the gap between these two methods has become larger as the size of the network has increased. This also suggests that finding good k-trees is more important than learning the optimal BN from a given k-tree.

Inference Accuracy and Complexity

In order to demonstrate that bounding tree-width improves the inferential complexity significantly, especially for large complex networks, while maintaining its accuracy, we first run a synthetic evaluation. We build two synthetic networks that have very large tree-width, namely *rand50* and *rand60* with 50 and 60 variables, respectively. Table 2 gives, on the left-hand side, the details about the two networks. All the

[2]Details about the datasets, http://homepages.rpi.edu/~nies/nie2016aaai_supp.pdf

[3]http://www.cs.helsinki.fi/u/jazkorho/aistats-2013/

[4]https://bitbucket.org/twilp/twilp/

[5]http://www.cs.york.ac.uk/aig/sw/gobnilp/

Table 1: Performance of different methods with tree-width limit 4. The median in 10 runs with different seeds is reported. A symbol – indicates no solution found within the time limit. Best score among approximate methods is in bold.

DATASET	VAR.	MILP	TWILP	S+K&P	S2	K&P+	S2+	GOBNILP
nursery	9	$-7.216\cdot10^4$	$-7.216\cdot10^4$	$-7.224\cdot10^4$	$\mathbf{-7.216\cdot10^4}$	$-7.220\cdot10^4$	$\mathbf{-7.216\cdot10^4}$	$-7.216\cdot10^4$
breast	10	$-2.685\cdot10^3$	$-2.685\cdot10^3$	$-2.714\cdot10^3$	$\mathbf{-2.686\cdot10^3}$	$-2.689\cdot10^3$	$\mathbf{-2.686\cdot10^3}$	$-2.685\cdot10^3$
housing	14	$-3.159\cdot10^3$	$-3.205\cdot10^3$	$-3.459\cdot10^3$	$\mathbf{-3.196\cdot10^3}$	$-3.290\cdot10^3$	$-3.211\cdot10^3$	$-3.159\cdot10^3$
adult	15	$-2.004\cdot10^5$	$-2.007\cdot10^5$	$-2.050\cdot10^5$	$-2.010\cdot10^5$	$-2.018\cdot10^5$	$\mathbf{-2.007\cdot10^5}$	$-2.004\cdot10^5$
zoo	17	$-5.624\cdot10^2$	$-5.782\cdot10^2$	$-6.762\cdot10^2$	$-5.826\cdot10^2$	$-6.346\cdot10^2$	$\mathbf{-5.794\cdot10^2}$	$-5.615\cdot10^2$
letter	17	$-1.843\cdot10^5$	$-1.884\cdot10^5$	$-1.981\cdot10^5$	$-1.879\cdot10^5$	$-1.900\cdot10^5$	$\mathbf{-1.860\cdot10^5}$	$-1.840\cdot10^5$
mushroom	22	$-5.783\cdot10^4$	$-6.381\cdot10^4$	$-7.942\cdot10^4$	$-6.043\cdot10^4$	$-7.337\cdot10^4$	$\mathbf{-5.999\cdot10^4}$	$-5.557\cdot10^4$
wdbc	31	$-6.995\cdot10^3$	$-7.788\cdot10^3$	$-8.907\cdot10^3$	$-7.201\cdot10^3$	$-7.318\cdot10^3$	$\mathbf{-7.143\cdot10^3}$	$-6.863\cdot10^3$
audio	62	$-2.541\cdot10^3$	$-2.161\cdot10^3$	$-2.458\cdot10^3$	$-2.131\cdot10^3$	$-2.157\cdot10^3$	$\mathbf{-2.126\cdot10^3}$	$-2.013\cdot10^3$
hill	100	$-4.235\cdot10^4$	–	$-1.412\cdot10^3$	$-1.137\cdot10^3$	$-9.921\cdot10^2$	$\mathbf{-8.784\cdot10^2}$	–
community	100	–	–	$-1.129\cdot10^5$	$-9.330\cdot10^4$	$-8.756\cdot10^4$	$\mathbf{-8.384\cdot10^4}$	–

nodes are binary. The (conditional) probability distributions for each node are randomly generated from Beta(0.5,0.5). 10,000 samples are generated from these networks by a simple ancestral sampling, which we use for learning a new network with tree-width bounded in 3. Training time limit is set to 10 minutes. With both learned network of small tree-width and original network of large tree-width, we run the junction tree algorithm for exact inference. As a measurement of the inferential computational complexity, we use the total running time of 1,000 computations of marginal probability distributions, averaged over all variables (by repeating the experiment multiple times). The inference time is measured by the ratio of the running time of the learned network to that of the original network. The results are given in Table 2 (on the right-hand side). The ratios of the inference time on learned network to that of the original networks are 32.8% and 4.3%, respectively for *rand50* and *rand60*. While the inference time complexity is reduced significantly by using the proposed algorithm, accuracy has been well preserved, which can be verified by the Hellinger and max-absolute errors (details on their calculations are given in the next section). This suggests that such approach could be used as new approximate inference procedure for BNs, as we investigate in the continuation.

UAI 2014 Probabilistic Inference Competition

We now evaluate the sampling-learning-exact-inference approach (that is, we (i) take a network on which we want to run probabilistic inference, (ii) sample data from it, (iii) learn a BN of small tree-width, (iv) run exact inference on the learned network) using several networks from the UAI 2014 Probabilistic Inference Competition[6]. Given the structure and parameters of a BN (or a Markov random field), the task is to compute the marginal probability distribution of each one of the variables given some evidence (MAR task). We selected 3 networks from the *CSP* category, 6 from *DBN* and 65 from *ObjectDetection* (the reason for choosing such networks was the number of variables going up to 100). 10,000 data samples are collected from each network.

[6]http://www.hlt.utdallas.edu/~vgogate/uai14-competition/

Bear in mind that the quality of sampling is not part of the goal of this paper, so we shall assume that sampling has been effectively performed without further questions. That said, for BNs, the samples are generated using a simple and efficient ancestral sampling, in which we go through nodes in a topological order so parents have always been sampled before the current node. For Markov random fields, sampling is a more challenging task. We use a Gibbs sampling approach with restarts and a large burn-in period. In fact, one might even question whether it is worth to sample from a Markov random field to then learn a bounded tree-width BN instead of using the sampling directly to perform the desired inference. One advantage of learning a BN is that the same learned network can be used to perform multiple different queries with different observations. Moreover, the sampling from BNs is easy, so this whole scheme is certainly more appropriate in such cases. In spite of that, our goal is to evaluate the goodness of bounded tree-width learning, so we assume the samples to be appropriate.

From the data we have produced, a BN with tree-width at most 4 is learned using the proposed method. The junction tree algorithm is then used to perform exact inference, to which we provide the corresponding optimal elimination order obtained from the k-tree (Section).

To be consistent with the procedure during the inference competition, we employ the following evaluation criteria for the MAR task: the Hellinger error and the max-absolute error. The Hellinger error is defined as

$$H_{err} = \frac{1}{n}\sum_{i=1}^{n} Hell\left(P^*(X_i), P(X_i)\right) ,$$

where $Hell\left(P^*(X_i), P(X_i)\right)$ is the Hellinger distance between the true probability distribution $P^*(X_i)$ corresponding to the i^{th} variable and the approximate one $P(X_i)$ from the learned network. For two discrete probability distributions $P = (p_1, \ldots, p_m)$ and $Q = (q_1, \ldots, q_m)$, their Hellinger distance is defined as

$$Hell(P, Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{m}(\sqrt{p_j} - \sqrt{q_j})^2} .$$

Table 2: On the left-hand side, two synthetic networks used for the evaluation of the inference complexity and accuracy. Fan-in and fan-out are maximum number of parents and children. Width is the induced tree-width. On the right-hand side, the inference accuracy and time, which is the ratio of the inference time of learned network to that of the original network.

NET	VAR.	ARCS	FAN-IN	FAN-OUT	WIDTH	H_{err}	A_{err}	Time
Rand50	50	283	7	22	18	0.0367	0.0017	32.8%
Rand60	60	517	10	41	22	0.0476	0.0023	4.3%

Table 3: The Hellinger and max-absolute errors for the MAR task of test cases in the UAI 2014 inference competition. OB stands for the category of ObjectDetection. Average error is reported.

	Hellinger error			Max-absolute error		
NET	LBP	SampleSearch	S2+	LBP	SampleSearch	S2+
DBN_{ave}	0.2095	0.0422	0.0572	0.1886	0.0479	0.0541
CSP_{ave}	0.0324	0.0031	0.0037	0.0299	0.0020	0.0035
OB_{ave}	0.0209	0.0202	0.0134	0.0087	0.0087	0.0057
Average	0.0367	0.0213	**0.0165**	0.0242	0.0116	**0.0095**

The max-absolute error is computed as

$$A_{err} = \frac{1}{n}\sum_{i=1}^{n} \max_{t} |P^*(X_i = t) - P(X_i = t)| .$$

The proposed method can be viewed as taking the complicated network, sampling from it, building a simpler network with small tree-width, and finally using an exact inference method. As baseline methods, we compare the results with loopy belief propagation (LBP) and SampleSearch[7] (Gogate and Dechter 2007) with iterative join graph propagation (IJGP) for approximate inference, both applied to the original competition networks. LBP was run until completion, while the other methods were given two minutes per instance. In our experiment, the junction tree algorithm has an average running time of 0.2 second per instance, so time was mostly spent during learning (which is also accounted in the two minutes). The average errors are given in Table 3. Compared to the approximate inference algorithms, the proposed method reached comparable accuracy in the MAR task. In most cases, it outperformed LBP and SampleSearch in terms of both Hellinger error and max-absolute error. Some exceptions are in the DBN and CSP categories. One reason is that these networks assign extreme marginal probabilities to some nodes. For example, node 2 in DBN_12 network has ground truth probability of 6.11×10^{-6} to take state 1. Our algorithm estimated it as 8.22×10^{-5} and SampleSearch estimated it as 6×10^{-6}. Intuitively, they all indicate this node always takes state 2. Our sample based method requires a large number of samples to estimate the extreme probabilities. In contrast, SampleSearch directly computes them from the ground truth network. This explains why we have problems with such networks. In general, the approach has achieved good accuracy, and meanwhile the complexity of exact inference is reduced due to the small tree-width. As an advantage over the others, multiple different inferences could be queried very efficiently using the learned BN without the need of further learning (even though this is not accounted for in the competition, so we have not exploited such fact).

[7] http://graphmod.ics.uci.edu/group/IJGPSampleSearch

Conclusion

In this paper we present an algorithm for learning Bayesian networks with bounded tree-width out of potentially good k-trees. Using the informative score, we introduce a search method that can efficiently find these k-trees of high promise, which in return are able to produce Bayesian networks with high BDeu scores (as empirically seen). Multiple experiments indicate the effectiveness of this procedure. In particular, the proposed method achieved close to optimal performance for small networks and was able to scale up to larger networks having better performance than the current state of the art. Because it provides an optimal elimination order that guarantees inference efficiency, using the learned network for reasoning and extracting information is greatly facilitated. Inspired by that, we created a sampling plus learning plus exact inference procedure and compared to well-known approaches in an important inference task of UAI 2014's competition. With bounded tree-width, inference complexity is significantly reduced with respect to the original networks (especially for large complex networks) while inference accuracy is usually well maintained.

Acknowledgements This work is supported in part by the grant N00014-12-1-0868 from the Office of Navy Research and by the grant W911NF-12-1-0473 from the Army Research Office.

References

Arnborg, S.; Corneil, D. G.; and Proskurowski, A. 1987. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods* 8(2):277–284.

Bache, K., and Lichman, M. 2013. UCI machine learning repository.

Barlett, M., and Cussens, J. 2013. Advances in Bayesian Network Learning using Integer Programming. In *Proc. 29th Conf. on Uncertainty in AI*, 182–191.

Berg, J.; Järvisalo, M.; and Malone, B. 2014. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, 86–95.

Buntine, W. 1991. Theory refinement on Bayesian networks. In *Proc. 7th Conf. on Uncertainty in AI*, 52–60.

Chow, C., and Liu, C. 1968. Approximating discrete probability distributions with dependence trees. *Inf. Theory, IEEE Trans. on* 14(3):462–467.

Cooper, G. F., and Herskovits, E. 1992. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learning* 9(4):309–347.

Cussens, J.; Bartlett, M.; Jones, E. M.; and Sheehan, N. A. 2013. Maximum Likelihood Pedigree Reconstruction using Integer Linear Programming. *Genetic Epidemiology* 37(1):69–83.

de Campos, C. P., and Ji, Q. 2010. Properties of Bayesian Dirichlet scores to learn Bayesian network structures. In *AAAI Conference on Artificial Intelligence*, 431–436. AAAI Press.

de Campos, C. P., and Ji, Q. 2011. Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research* 12:663–689.

Elidan, G., and Gould, S. 2008. Learning Bounded Treewidth Bayesian Networks. *J. of Mach. Learning Res.* 9:2699–2731.

Gogate, V., and Dechter, R. 2007. Samplesearch: A scheme that searches for consistent samples. In *International Conference on Artificial Intelligence and Statistics*, 147–154.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learning* 20(3):197–243.

Hemmecke, R.; Lindner, S.; and Studený, M. 2012. Characteristic imsets for learning Bayesian network structure. *Int. J. of Approx. Reasoning* 53(9):1336–1349.

Jaakkola, T.; Sontag, D.; Globerson, A.; and Meila, M. 2010. Learning Bayesian network structure using LP relaxations. In *Proc. 13th Int. Conf. on AI and Stat.*, 358–365. JMLR W&CP 9.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models*. MIT press.

Korhonen, J. H., and Parviainen, P. 2013. Exact Learning of Bounded Tree-width Bayesian Networks. In *Proc. 16th Int. Conf. on AI and Stat.*, 370–378. JMLR W&CP 31.

Kwisthout, J. H. P.; Bodlaender, H. L.; and van der Gaag, L. C. 2010. The Necessity of Bounded Treewidth for Efficient Inference in Bayesian Networks. In *Proc. 19th European Conf. on AI*, 237–242.

Murphy., K. P. 2012. *Machine learning: a probabilistic perspective*. MIT Press.

Nie, S.; Mauá, D. D.; de Campos, C. P.; and Ji., Q. 2014. Advances in learning Bayesian networks of bounded treewidth. In *Advances in Neural Information Processing Systems*, 2285–2293.

Nie, S.; de Campos, C. P.; and Ji, Q. 2015. Learning bounded tree-width Bayesian networks via sampling. In *13th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 387–396.

Parviainen, P.; Farahani, H. S.; and Lagergren., J. 2014. Learning bounded tree-width Bayesian networks using integer linear programming. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, 751–759.

Patil., H. P. 1986. On the structure of k-trees. *J. Combin. Inform. System Sci* 11(2-4):57–64.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Stat.* 6(2):461–464.

Yuan, C., and Malone, B. 2013. Learning optimal Bayesian networks: A shortest path perspective. *J. of Artif. Intell. Res.* 48:23–65.

Yuan, C.; Malone, B.; and Wu, X. 2011. Learning Optimal Bayesian Networks Using A* Search. In *Proc. 22nd Int. Joint Conf. on AI*, 2186–2191.