Proceedings of Machine Learning Research vol 72, 225-236, 2018 PGM 2018

Bayesian Network Structure Learning with Side Constraints

Andrew C. Li ACLI@UWATERLOO.CA
Peter van Beek VANBEEK@CS.UWATERLOO.CA
Cheriton School of Computer Science
University of Waterloo

Abstract

Hybrid methods for Bayesian network structure learning that incorporate both observed data and expert knowledge have proven to be important in many fields. Previous studies have presented both exact and approximate hybrid methods for structure learning. In this paper, we propose an approximate method based on local search that is capable of efficiently handling a variety of prior knowledge constraints, including an important class of non-decomposable *ancestral* constraints that assert indirect causation between random variables. In our experiments, our proposed approximate method is able to significantly outperform an existing state-of-the-art approximate method in finding feasible solutions when *hard constraints* are imposed. Our approach is able to find near-optimal networks while scaling up to almost fifty random variables. In contrast, previous exact methods are unable to handle more than twenty random variables. Furthermore, we show that when prior knowledge is integrated, we are often able to produce a network much closer to the ground truth network, particularly when the amount of data is limited.

Keywords: Bayesian network; structure learning; local search; expert knowledge; hard constraints; ancestral constraints.

1. Introduction

A Bayesian network (BN) is a probabilistic graphical model of a probability distribution with applications in various fields including medicine (Flores et al., 2011), ecology (Pollino et al., 2007), and bioinformatics (Friedman et al., 2000).

In practice, BNs are usually either fully specified by an expert or learned directly from observed data; however, both approaches have major drawbacks. Experts usually do not have sufficient domain knowledge to produce the full network, especially as the number of random variables increases. On the other hand, the task of learning the BN structure directly from data is NP-hard, even to approximate to within a reasonable factor (Chickering et al., 2003; Dasgupta, 1999) and data is often limited or expensive.

1.1 Expert Knowledge Constraints.

In many fields, hybrid methods for BN structure learning that incorporate both expert knowledge and data have been applied to obtain superior results; e.g. Flores et al. (2011); Sesen et al. (2013); Oyen et al. (2016); Antal et al. (2004). These studies use approximate search methods with informative priors, allowing an expert to specify confidence levels in various constraints. In contrast, we focus on incorporating prior knowledge as *hard constraints*. Our method builds on MINOBS, a local search based memetic algorithm for BN structure learning by Lee and van Beek (2017). We are able to handle the following types of constraints in our method.

(1) *Existence of an arc:* assert that a directed arc xy exists. The user can also specify that an undirected arc xy exists if the direction is not known.

(2) *Absence of an arc:* assert that the directed arc xy does *not* exist.

(3) *Ordering constraint:* assert x comes before y in some ordering of the nodes in the BN. This implies that there exists no directed y, x-path (but the converse does not necessarily hold).

(4) *(Positive) Ancestral constraint:* assert the existence of a directed x, y-path.

We refer to such constraints collectively as *side constraints*. These side constraints are expressive enough to indirectly handle some other common constraints, such as undirected arc absence (no arc xy or yx), causal tiers (only arcs from a lower tier variable to a higher tier variable are permitted) and specifying root nodes and leaves.

The primary challenge of our work was incorporating ancestral constraints which cannot be handled locally (by modifying the parents of a single node), or by pruning the space of orderings. They are notoriously difficult to incorporate into the structure learning process (Chen et al., 2016). However, ancestral constraints are important for a number of reasons. First, although ancestral constraints are weaker than directed arc existence constraints, direct dependencies can only be known given a high degree of domain knowledge. Experts are more likely to provide useful information when less specific constraints can also be asserted (Flores et al., 2011). Secondly, ancestral constraints can assert that one variable is an indirect cause of another, which is an abundant form of expert knowledge in many domains (e.g. Şahin et al. (2006); Ma et al. (2017); Leu and Chang (2013); Chen and Leu (2014); Giordano et al. (2013)). Thirdly, ideally BNs built for knowledge discovery or knowledge representation should encode all known causal dependencies. In our experiments, we demonstrate that imposing ancestral constraints produces a BN much closer to the ground truth network in terms of their causal statements. In fact, BN structure learning from data alone cannot distinguish between causation and correlation.

1.2 Results.

We present a novel hybrid BN structure learning method that can incorporate expert knowledge as hard constraints, including ancestral constraints. In our experimental evaluation, we demonstrate that our method is able to find high-quality solutions to constrained problems, while scaling to more variables than other existing methods. To the best of our knowledge, we are the first to successfully incorporate ancestral constraints into a local search based BN structure learning approach.

1.3 Related work.

We briefly discuss some related work on BN structure learning algorithms.

In a classic paper by Tsamardinos et al. (2006), greedy hill-climbing was used efficiently and accurately for standard BN structure learning. Another local search algorithm by de Campos and Castellano (2007) was able to handle precisely the constraints (1) – (3). However, this local search method is based on the space of DAGs while our proposed method primarily searches over the space of topological orderings.

A recent exact approach by Chen et al. (2016) called EC Tree was specifically designed to handle ancestral constraints. EC Tree can also handle constraints asserting the absence of a directed path and finds solutions with guaranteed optimality which our proposed method cannot. However,

their method does not scale beyond $n = 20$ random variables when using the BDeu score. GOBNILP (Bartlett and Cussens, 2017), a widely-used exact approach, is able to handle a large variety of constraints. Chen et al. (2016) demonstrated that GOBNILP can be made to handle ancestral constraints, but requires orders of magnitude more time than EC Tree.

Causal discovery via MML (CaMML) (Korb and Nicholson, 2010) uses stochastic search based on Markov Chain Monte Carlo. It is capable of handling all of the constraints (1)-(4) except for arc absence. One is also able to assert that two variables are correlated (i.e. not independent) and all constraints can be specified as a soft constraint by setting confidence levels. We compare CaMML with our proposed method in our experimental evaluation.

2. Background

We begin by reviewing preliminary details on Bayesian networks (BNs). A BN consists of a DAG G with nodes $V = \{v_1, \ldots, v_n\}$, where each node is a random variable, and a conditional probability distribution $P(v_i \mid \text{par}(v_i))$ for each random variable v_i, where $\text{par}(v_i)$ is the set of parents of v_i in G. Arcs in G from one node to another represent direct dependencies, and the full structure G encodes all conditional dependencies between the n variables. In this paper, we focus on the *score-and-search* method for BN structure learning. A scoring function $\sigma(G)$ is chosen where every DAG G on n variables is assigned a score based on how well G fits the observed data (we assume a lower score is better). A common property of most scoring functions, such as BIC/MDL (Schwarz, 1978), and BDeu (Heckerman et al., 1995), is *decomposability*, meaning the score of the DAG is the sum of its local scores at each node:

$$\sigma(G) = \sum_{i=1}^{n} \sigma(v_i,\ \text{par}(v_i))$$

It is common practice to precompute the scores for each random variable v_i and each candidate parent set $p \subseteq 2^{V \setminus \{v_i\}}$. *Parent set identification* involves pruning parent sets that never appear in optimal scoring solutions (further discussed in Section 3). *Structure optimization* is the task of assigning a parent set for each node to achieve a minimum scoring acyclic network.

In order to incorporate prior knowledge, we wish to solve the BN structure learning problem with additional hard constraints. We formally present the problem we focus on for this paper.

Definition 1 (Constrained Bayesian network structure learning problem) *Given a data set $D = \{d_1, \ldots, d_N\}$, where each d_i is a vector of discrete values over features/random variables V, a scoring function σ that measures how well a candidate structure is supported by the observed data D, and arbitrarily many constraints of one of the following forms:*

1. *$x \to y$ (directed arc existence) or x — y (undirected arc existence)*
2. *$x \nrightarrow y$ (directed arc absence)*
3. *$x < y$ (topological ordering constraint)*
4. *$x \rightsquigarrow y$ (ancestral constraint)*

The constrained Bayesian network structure learning problem *is to find a directed acyclic graph G over V minimizing the score $\sigma(G)$ subject to the following:*

- *for every constraint $x \rightarrow y$, the arc xy is in G.*
- *for every constraint x — y, one of the arcs xy or yx is in G.*
- *for every constraint $x \nrightarrow y$, the arc xy is* not *in G*
- *there exists an ordering $\mathcal{O}$ such that for all constraints $x < y$, x comes before y in $\mathcal{O}$ and $\mathcal{O}$ is a topological ordering for G*
- *for every constraint $x \rightsquigarrow y$, there exists a directed path from x to y in G*

MINOBS applies the *ordering-based search* (Teyssier and Koller, 2005) approach to BN structure learning. Given a topological ordering $\mathcal{O}$ of the nodes, the optimal DAG consistent with $\mathcal{O}$ can be found in polynomial time, assuming parent sets are restricted to some constant size at most k. This approach searches over the $O(n!)$ size space of orderings on n variables, where the score of an ordering is the minimum score of a BN consistent with that ordering.

The rest of the paper focuses on the methods used to approximately solve the *constrained BN structure learning problem* .

3. Parent Set Identification

Score-and-search approaches to BN structure learning rely on effective pruning strategies to reduce the number of candidate parent sets for each variable. The problem of *parent set identification* is to determine which parent sets do not appear in optimal solutions and hence do not need to further be considered. Unfortunately, existing pruning rules preserve the optimal solution under the assumption that we are solving the problem with *no* side constraints, but can be incorrect in the presence of side constraints. For example, consider the following widely used pruning rule.

Lemma 2 (Teyssier and Koller (2005)) *Let x_i be a node, Π, Π' potential parent sets for x_i such that $\Pi \subset \Pi'$ and $\sigma(x_i, \Pi) < \sigma(x_i, \Pi')$. Then Π' is not the parent set of x_i in an optimal DAG G^*.*

In the *(unconstrained) BN structure learning problem*, this rule allows Π' to be pruned since for any DAG G' containing Π' as a parent of x_i, one can instead assign Π as the parent of x_i resulting in a DAG G with a strictly lower score than G'. However, when the addition of constraints are considered, this rule no longer holds, as G' may satisfy arc existence or ancestral constraints that G does not. Therefore this pruning technique should not be used.

Our method uses new approaches to parent set identification specific to the *constrained BN structure learning* problem. For example, the following rules use constraints to eliminate infeasible solutions.

(P1) If $x \rightarrow y$, then any parent set of y that does not contain x can be pruned.

(P2) If $x \nrightarrow y$, then we may prune all parent sets of y containing x.

(P3) If $x < y$, then we may prune parent sets of x containing y.

(P4) If $x \rightsquigarrow y$, Π is a parent set of y, and for all nodes $p \in \Pi$, $p < x$, then Π may be pruned (since assigning Π as the parent set of y immediately makes satisfaction of $x \rightsquigarrow y$ impossible).

Using these rules to prune the candidate parent sets, any DAG constructed from the remaining candidate parent sets must satisfy all constraints of the form $x \rightarrow y$ and $x \nrightarrow y$. If we wish to only consider DAGs consistent with some ordering $\mathcal{O}$, we can easily handle the constraint $x - y$ using (P1) as the direction of the arc can be deduced from $\mathcal{O}$. The next rule is *inexact* but is able to drastically reduce the number of candidate parent sets.

(P5) *Let x_i be a node, Π, Π' potential parent sets for x_i, $\lambda \geq 1$ a constant such that $\Pi \subset \Pi'$ and $\lambda\sigma(x_i, \Pi) < \sigma(x_i, \Pi')$. Then Π' is pruned from the set of candidate parent sets.*

Notice that this pruning rule is equivalent to Lemma 2 if $\lambda = 1$ is chosen. We prune a parent set Π' if its score is significantly worse than one of its subsets Π, as Π' is unlikely to appear in optimal solutions to the *constrained BN structure learning problem*. The choice of λ is important; overpruning can be detrimental to the quality of DAGs found while underpruning is inefficient. We compute an initial value of λ using the formula,

$$\lambda = 1 + \omega(\frac{n}{N})(1 - (1 - \frac{m_{\text{anc}}}{n(n-1)})^2),$$

where n is number of random variables, N is number of data points, m_{anc} is number of ancestral constraints, and ω is a constant.

This formula satisfies some key desiderata. Consider G and H, the optimal-scoring DAGs without and with ancestral constraints imposed, respectively. We expect G and H to be close in structure and score if (i) the number of data points is large (as both approach the ground truth network, or an I-equivalent network) or (ii) the number of ancestral constraints is small. Therefore, in these cases we wish to have a small λ in order to prune parent sets with poor scores more liberally.

In our approach, we restart the search with a higher value of λ if feasible solutions cannot be found. We also specify a value of ω in our experimental procedure.

4. Constraint-based Hill Climbing

In this section, we discuss how to search for an optimal solution consistent with a given ordering $\mathcal{O}$ that satisfies all constraints. Recall that $x \rightarrow y$, $x \nrightarrow y$, and $x - y$ can all be handled in parent set identification. We assume that $\mathcal{O}$ is consistent with all ordering constraints as this is handled in the next section. Here we show how to handle ancestral constraints. For convenience, we denote $\phi(\mathrm{G})$ to be the number of ancestral constraints $x \rightsquigarrow y$ that are satisfied by a DAG G.

Definition 3 (Parent assignment neighbour) *Two DAGs G and G' with the same nodes are* parent assignment neighbours *if and only if the parent set of a single node differs between G and G'.*

In Algorithm 2, the minimum scoring DAG consistent with $\mathcal{O}$ is computed and set as the initial DAG G_0. We then apply the hill climbing method in Algorithm 1 on G_0. At each iteration of the hill climb, we choose an *improving parent assignment neighbour* G' of G, replacing G with G'. Here, we say that G' is an *improvement* over G if and only if $\phi(G') > \phi(G)$ or $\phi(G) = \phi(G'), \sigma(G') < \sigma(G)$, i.e. satisfying the most ancestral constraints is the primary objective, with score optimization as a secondary objective. The list of candidate parent sets, *allParents*, is sorted in increasing order of score. For each candidate parent set p for node v, we consider G', the *parent assignment neighbour* of G that results from assigning p as the parent for v in G. As the number of candidate parent sets can be large, we adopt a *first improvement* neighbour selection strategy. Unfortunately, computing

Algorithm 1: hillClimbDAG(G, $\mathcal{O}$)

```
allParents ← allParentSets() ;            /* sorted by increasing score */
while true do
    for p in allParents do
        x ← child(p) ;
        if not tabu(x) and feasible(p, O) then
            G' ← G;
            G'.parentOf(x) ← p ;
            if G' improves G then
                G ← G' ;
                break ;
            end if
        end if
    end for
    if no improvement found then
        if random(0, 1) < walkProb then
            p ← random feasible parent set;
            x ← child(p) ;
            G.parentOf(x) ← p ;
            tabu x for 3 iterations ;
        else
            break ;
        end if
    end if
end while
return G ;
```

Algorithm 2: bestDAGForOrdering($\mathcal{O}$)

```
G_0 ← initialDAG(O) ;
return hillClimbDAG(G_0, O) ;
```

$\phi(G')$ can be quite expensive, performing a depth-first search to check each ancestral constraint $x \rightsquigarrow y$ in the worst case. However, a constraint $x \rightsquigarrow y$ that is satisfied in G must be satisfied in G' unless v is a descendant of x and an ancestor of y in G. Furthermore, if a constraint $x \rightsquigarrow y$ is not satisfied in G, then it is satisfied in G' if and only if v is an ancestor of y and v has a parent in G' that was a descendant of x in G. Thus, the number of depth first searches can be reduced by precomputing which ancestral constraints are satisfied, and the descendants and ancestors of nodes involved in ancestral constraints.

A problem with the *parent assignment neighbourhood* is that it is small and the hill climbing algorithm can become stuck in low-quality local minima. We solve this by introducing random walks and a tabu list. When the hill climb reaches a local minimum, it moves to a random non-improving solution with some small probability *walkProb*. The node whose parent set was affected

is then declared tabu for the next three iterations of the hill climb so the change is not immediately undone (while a node is tabu, it cannot have its parent set reassigned).

5. Ordering-based Search

In this section, we focus on hill climbing over the space of topological orderings. The score of an ordering $\mathcal{O}$ is the minimum value of $\sigma(G)$ for DAGs G consistent with $\mathcal{O}$ and satisfying all constraints. As the true minimum-scoring DAG is difficult to find, we treat the best DAG found by Algorithm 2 as the score for $\mathcal{O}$.

We follow roughly the same search strategy and neighbourhood that is outlined in MINOBS (Lee and van Beek, 2017) which uses the *insert neighbourhood* and *swap-adjacent neighbourhood*. Simply, $\mathcal{O}'$ is an *insert neighbour* of $\mathcal{O}$ if it results from selecting an element of $\mathcal{O}$ and inserting it to a new index (e.g. $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_4, x_2, x_3\}$ are *insert neighbours*). A *swap-adjacent neighbour* of $\mathcal{O}$ is an ordering resulting from swapping an element of $\mathcal{O}$ with its preceding element (e.g. $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_2, x_4, x_3\}$ are *swap-adjacent neighbours*).

The *insert neighbourhood* of $\mathcal{O}$ can be traversed using a series of *swap-adjacent* moves. Each *swap-adjacent* move only affects the choice of parent sets for the two swapped elements when no additional constraints are present, but this is not the case with side constraints. We found the following strategy to be effective. Let the current ordering be $\mathcal{O} = (x_1, ..., x_i, x_{i+1}, ..., x_n)$ and the best feasible DAG found consistent with $\mathcal{O}$ be G. Suppose index i is being swapped with index $i + 1$ and we wish to find the best feasible DAG consistent with $\mathcal{O}' = (x_1, ..., x_{i+1}, x_i, ..., x_n)$. We found that using G as the initial starting point for hill climbing on $\mathcal{O}'$ improved the running time, as G satisfies all ancestral constraints and is likely of high quality. If the parent set of x_{i+1} in G contains x_i, then G is not feasible under $\mathcal{O}'$ but we fix this by replacing the parent of x_{i+1} with the lowest-scoring feasible parent under $\mathcal{O}'$.

6. Experimental Evaluation

In our experiments, we compare our proposed method, denoted by MINOBSx[1], against the CaMML[2] software. CaMML was chosen for comparison as it is widely used in application fields (see, e.g. Flores et al. (2011), Kennett et al. (2001), Sesen et al. (2013)) and is also based on a stochastic score-and-search approach. It is important to note that the learning problem is significantly easier without ancestral constraints (adding any of the other constraints also generally improves performance), therefore we did not find it useful to compare MINOBSx against methods that could not handle ancestral constraints. Unfortunately, the source code for EC Tree was not made available.

Our test instances consist of all small and medium networks in the Bayesian Network Repository[3], the largest of which is *barley* (48 nodes). We randomly sampled 6 small datasets and 6 large datasets from the associated joint probability distributions of these ground truth networks. Two sets of experiments were run: (i) only imposing ancestral constraints and (ii) imposing various constraints. For each set of experiments, we selected 4 fixed percentages and sampled 5 sets of constraints at that percentage from the ground truth network. For a percentage $p\%$, we sampled constraints in the following way.

1. Code is available at https://github.com/acliuw/MINOBS-anc
2. http://bayesian-intelligence.com/software/
3. http://www.bnlearn.com/bnrepository/

- We choose $p\%$ of all directed arcs in the ground truth network. For each arc xy we report $x \to y$ with probability $\frac{1}{2}$, otherwise we report x — y.
- We choose $p\%$ of all pairs (x, y) such that there is no directed arc xy in the ground truth network and report $x \nrightarrow y$.
- We choose $p\%$ of all pairs (x, y) such that there is a directed x, y-path in the ground truth network and report $x \rightsquigarrow y$.
- We use topological sort to retrieve an arbitrary topological ordering $\mathcal{O}$ of the ground truth network and report $p\%$ of the $\binom{n}{2}$ possible ordering constraints $x < y$ for $\mathcal{O}$.

When comparing MINOBSx and CaMML, we excluded arc absence constraints as CaMML does not accept these. For each dataset and each constraint set, we ran both MINOBSx and CaMML, providing the dataset and constraint set as input.

6.1 Setup.

All tests were run on a single core of an Intel "Broadwell" E5-2683 v4 CPU @ 2.1 Ghz with a memory limit of 512 MB for small networks and 4 GB for medium networks.

The parent set scores for MINOBSx were generated using GOBNILP (Bartlett and Cussens, 2017) with a parent limit of 3, due to limited disk space.[4] The running time for GOBNILP is included in the running time for MINOBSx in the results. MINOBSx was run for 50 generations on small networks and 10 generations on medium networks. The parameter ω described in Section 3 was tuned to a value of 50 using the *cancer*, *child*, and *barley* instances.

CaMML was run under its default settings. We allowed CaMML more running time than MINOBSx as CaMML is implemented in Java while MINOBSx is implemented in C++. All constraints passed to CaMML were specified with a confidence level of 1.0 (hard constraint) except ancestral constraints, which were assigned a confidence of 0.9999. CaMML was often unable to produce any solutions except on very small networks when the confidence level for ancestral constraints was set to 1.0. Surprisingly, by changing the confidence to 0.9999 for ancestral constraints, CaMML was able to completely solve some instances that it otherwise was not able to find any solutions to.

6.2 Results.

MINOBSx was able to find feasible solutions incorporating all constraints imposed *for all tests*. CaMML was able to consistently incorporate most constraints on small networks, but performed poorly on all larger cases. Results for 3 of the 11 networks are explicitly specified in Table 1. [5]

We also compare the quality of the networks learned by MINOBSx against the ground truth networks. We do not compare against the networks learned by CaMML as the two methods use different scoring functions, which can have a large impact on results. For instances with only ancestral constraints, we directly analyze the results produced in the previous group of experiments; however, for instances with various constraints, we re-ran MINOBSx on constraint sets that included arc absence constraints. We report the number of missing/extra/reversed arcs and the structural intervention distance (SID) (Peters and Bühlmann, 2015) (which measures the difference in causal

4. For the largest case, *barley*, the number of parent sets generated is $48 * (\binom{48-1}{3} + \binom{48-1}{2} + \binom{48-1}{1} + \binom{48-1}{0}) =$ 832,512 with a parent limit of 3. If the parent limit is increased to 4, the number of parent sets generated is 9,394,032.
5. Extended versions of Table 1 and Table 2 containing full results are available at github.com/acliuw/MINOBS-anc.

instance	N	%	MINOBSx			CaMML		
			% feasible	% sat	t (seconds)	% feasible	% sat	t (seconds)
asia 8 variables 18 params	250	10 / 5	100 / 100	100 / 100	1.1 / 0.5	100 / 100	100 / 100	5.8 / 5.4
		25 / 10	100 / 100	100 / 100	1.3 / 0.5	100 / 100	100 / 100	6.3 / 5.5
		50 / 15	100 / 100	100 / 100	0.9 / 0.3	100 / 100	100 / 100	6.8 / 5.9
		100 / 20	100 / 100	100 / 100	0.5 / 0.2	100 / 100	100 / 100	8.3 / 5.7
	1000	10 / 5	100 / 100	100 / 100	0.9 / 0.4	100 / 100	100 / 100	5.5 / 5.0
		25 / 10	100 / 100	100 / 100	1.1 / 0.4	100 / 100	100 / 100	6.0 / 5.3
		50 / 15	100 / 100	100 / 100	0.7 / 0.3	100 / 96.7	100 / 99.7	6.3 / 5.6
		100 / 20	100 / 100	100 / 100	0.4 / 0.2	100 / 100	100 / 100	7.3 / 5.6
insurance 27 variables 984 params	500	10 / 5	100 / 100	100 / 100	180.5 / 104.9	50.0 / 70.0	95.9 / 99.0	439.5 / 325.3
		25 / 10	100 / 100	100 / 100	318.9 / 56.5	56.7 / 50.0	98.0 / 98.9	723.9 / 385.0
		50 / 15	100 / 100	100 / 100	328.7 / 52.8	30.0 / 33.3	98.5 / 98.1	1165.6 / 485.2
		100 / 20	100 / 100	100 / 100	292.5 / 37.4	0 / 53.3	98.1 / 99.6	2052.3 / 571.7
	2000	10 / 5	100 / 100	100 / 100	124.0 / 88.3	0 / 53.3	78.8 / 98.1	438.5 / 309.8
		25 / 10	100 / 100	100 / 100	236.3 / 49.6	16.7 / 23.3	92.4 / 96.8	748.6 / 393.2
		50 / 15	100 / 100	100 / 100	251.5 / 48.2	3.3 / 10.0	94.4 / 96.6	1175.2 / 487.9
		100 / 20	100 / 100	100 / 100	233.1 / 33.4	0 / 10.0	95.6 / 98.8	1956.8 / 571.1
barley 48 variables 114005 params	2000	10 / 5	100 / 100	100 / 100	2321.4 / 5866.8	0 / 0	70.3 / 89.3	19824.8 / 11666.1
		25 / 10	100 / 100	100 / 100	4228.9 / 2941.3	0 / 0	74.4 / 93.6	37034.3 / 14864.9
		50 / 15	100 / 100	100 / 100	7163.0 / 3518.9	0 / 0	73.9 / 95.0	70433.0 / 20339.2
		100 / 20	100 / 100	100 / 100	7246.6 / 1806.3	0 / 0	80.9 / 96.3	114036.7 / 22366.9
	8000	10 / 5	100 / 100	100 / 100	4761.1 / 6032.7	0 / 0	44.0 / 82.8	18092.6 / 10759.6
		25 / 10	100 / 100	100 / 100	4063.8 / 3620.0	0 / 0	52.5 / 91.8	35308.3 / 14477.5
		50 / 15	100 / 100	100 / 100	5137.8 / 3022.8	0 / 0	54.0 / 95.1	64893.5 / 18142.0
		100 / 20	100 / 100	100 / 100	5675.6 / 1638.8	0 / 0	53.4 / 95.1	111338.4 / 21184.4

Table 1: Performance results for ancestral constraints only (first number in each pair) and with various constraints (second number in each pair). N is number of observations, $\%$ is the fixed percentage used to sample constraints, $\%$ feasible is the percentage of cases where the solution satisfied *all* constraints imposed, $\%$ sat is the percentage of satisfied constraints out of those constraints imposed, t is the running time required by the program. Highlighted cells indicate that not all constraints were able to be satisfied.

statements) between the learned and ground truth networks. We also report the percentage difference between the BDeu score of the learned network and the optimal score produced by GOBNILP[6] on the instance *without side constraints*. The results for 4 networks of varying sizes are displayed in Table 2; however, the other networks produced similar results.

From these results, MINOBSx appears much more robust than CaMML, especially as the number of variables increases. Even when it spends significantly more time than MINOBSx, CaMML is often unable to find feasible solutions (e.g. see the *barley* case). Also, the percentage difference between the BDeu scores to the constrained problem and the unconstrained problem is small and never over 4%. Hence, solutions produced by MINOBSx are within a 4% error margin of the optimal *constrained* solution in the worst case.

When we analyze the affect of ancestral constraints on solution quality, we see that the number of missing and reversed arcs tends to decrease while the number of extra arcs does not change or worsens. However, in some cases, the number of missing arcs does not improve, even when 100% of ancestral constraints are added. A small number of various constraints seems to offer more noticeable and consistent improvements to missing and extra arcs. A likely reason is that arc existence

6. A parent limit of 8 was imposed for small networks, 6 on medium networks (except for *mildew* and *barley*), and 4 for *mildew* and *barley* due to time constraints.

instance	N	%	Missing	Extra	Reversed	SID	Score (BDeu)
		0*	1.5	1.7	1.0	12.2	0%
		10 / 5	1.4 / 1.4	1.6 / 1.6	0.7 / 0.9	10.3 / 11.1	0.0% / 0.0%
	250	25 / 10	1.2 / 1.4	1.8 / 1.6	0.5 / 0.6	7.2 / 8.9	0.1% / 0.1%
asia		50 / 15	1.0 / 0.8	2.0 / 1.4	0.3 / 0.4	4.3 / 4.4	0.2% / 0.3%
		100 / 20	0.5 / 0.7	1.7 / 1.1	0.0 / 0.2	1.8 / 3.9	0.3% / 0.3%
8 variables		0*	0.8	0.3	1	9.0	0%
18 parameters		10 / 5	0.7 / 0.8	0.4 / 0.4	0.9 / 1.1	7.3 / 9.0	0.0% / 0.0%
	1000	25 / 10	0.4 / 0.5	0.5 / 0.3	0.4 / 0.8	4.2 / 6.4	0.0% / 0.0%
		50 / 15	0.2 / 0.3	0.4 / 0.4	0.0 / 0.5	0.4 / 3.6	0.0% / 0.0%
		100 / 20	0.0 / 0.2	0.3 / 0.3	0.0 / 0.4	0 / 3.1	0.0% / 0.0%
		0*	5.3	1.0	3.0	115.7	0%
		10 / 5	4.8 / 4.8	1.1 / 0.8	1.9 / 1.3	91.4 / 82.6	0.1% / 0.2%
	500	25 / 10	4.7 / 4.3	1.9 / 1.6	1.6 / 1.8	93.9 / 79.4	0.2% / 0.3%
child		50 / 15	3.9 / 4.2	1.4 / 1.2	1.7 / 0.7	76.1 / 69.9	0.3% / 0.4%
		100 / 20	2.2 / 3.6	2.2 / 0.8	0.0 / 0.3	35.8 / 57.6	0.9% / 0.4%
20 variables		0*	1.7	0.2	3.5	79.2	0%
230 parameters		10 / 5	1.5 / 1.3	0.2 / 0.2	0.5 / 0.1	26.0 / 20.8	0.0% / 0.0%
	2000	25 / 10	0.7 / 1.0	0.3 / 0.3	0.3 / 0.5	12.1 / 18.0	0.0% / 0.0%
		50 / 15	0.7 / 1.0	0.3 / 0.3	0.5 / 0.0	12.2 / 16.2	0.0% / 0.0%
		100 / 20	0.2 / 1.0	0.3 / 0.1	0.0 / 0.0	2.5 / 16.6	0.1% / 0.0%
		0*	2.2	5.8	1.3	45.7	0%
		10 / 5	2.0 / 1.6	6.2 / 5.5	1.1 / 1.6	34.7 / 46.4	0.0% / 0.1%
	1000	25 / 10	2.0 / 1.8	6.3 / 5.4	0.7 / 0.8	27.7 / 32.2	0.1% / 0.1%
alarm		50 / 15	2.0 / 1.5	6.1 / 5.0	0.3 / 0.6	22.4 / 27.7	0.1% / 0.2%
		100 / 20	2.0 / 1.5	6.2 / 4.4	0.0 / 0.1	18.0 / 18.3	0.1% / 0.2%
37 variables		0*	2.0	3.2	1.8	39.5	0%
509 parameters		10 / 5	2.0 / 1.6	4.6 / 4.5	0.6 / 1.2	24.9 / 38.8	0.0% / 0.0%
	4000	25 / 10	2.0 / 1.8	4.5 / 4.3	0.3 / 0.6	20.1 / 28.0	0.0% / 0.0%
		50 / 15	2.0 / 1.5	4.1 / 4.2	0.0 / 0.4	18.0 / 22.3	0.0% / 0.1%
		100 / 20	1.7 / 1.4	4.2 / 3.7	0.0 / 0.3	12.3 / 20.4	0.0% / 0.1%
		0*	32.3	8.2	9.7	949.5	0%
		10 / 5	33.3 / 31.7	14.4 / 12.2	5.1 / 4.4	792.9 / 756.7	0.6% / 1.4%
	2000	25 / 10	31.9 / 31.0	15.6 / 13.8	5.3 / 4.8	802.6 / 741.5	0.9% / 1.7%
barley		50 / 15	31.0 / 27.5	17.4 / 11.3	3.0 / 3.2	699.4 / 666.9	1.4% / 2.9%
		100 / 20	30.2 / 26.4	19.5 / 11.8	0.0 / 1.5	619.3 / 628.1	2.4% / 3.9%
48 variables		0*	25.5	3.7	9.7	794.7	0%
114005 parameters		10 / 5	25.6 / 24.6	8.9 / 8.0	5.3 / 5.2	636.6 / 625.4	0.3% / 0.8%
	8000	25 / 10	25.0 / 24.0	10.4 / 8.8	5.5 / 4.3	644.4 / 584.7	0.4% / 0.9%
		50 / 15	22.1 / 20.5	9.2 / 7.6	3.3 / 1.9	541.7 / 487.4	0.6% / 1.4%
		100 / 20	20.8 / 19.3	12.0 / 7.0	0.0 / 1.6	457.2 / 507.2	0.9% / 1.8%

Table 2: Results for medium networks, with ancestral constraints only (first number in each pair) and with various constraints (second number in each pair). Rows marked 0^* under the $\%$ column indicate optimal solutions produced by GOBNILP *without side constraints*.

and absence constraints can directly fix a missing or extra arc whereas ancestral constraints are ambiguous. If an incorrect path is used to satisfy an ancestral constraint, this may create additional extra arcs while not improving the number of missing arcs. On the other hand, ancestral constraints alone appear to significantly improve the SID. This is expected, since ancestral constraints assert indirect causes and the SID measures differences in causal statements.

One pitfall we note is that due to the low parent limit of 3, MINOBSx is unable to recover the ground truth network in some cases, while CaMML does not explicitly set a parent limit. This is observed in the results for *alarm* which contains a node with 4 parents. However, this is generally insignificant as large parent sets result in overly complex models that are unlikely to appear in optimal solutions, especially when the amount of data is small.

7. Conclusion

We present a novel method for incorporating prior knowledge constraints into a local search algorithm for Bayesian network structure learning, including non-decomposable *ancestral constraints*. While previous exact methods could handle up to twenty random variables, we demonstrate that our stochastic search method is able to scale up to nearly fifty random variables while producing high quality networks. When compared against CaMML, a software widely used by researchers in applied fields, our proposed method shows much more robust and consistent performance on networks with more than twenty nodes.

For future work, similar techniques to those presented in this paper may prove useful for efficiently handling other non-decomposable constraints. One such example is d-separation, which can be used to assert conditional independence relationships between variables.

7.1 Acknowledgements.

This research was supported in part by WestGrid[7], Compute Canada[8], and an NSERC USRA Award.

References

P. Antal, G. Fannes, D. Timmerman, Y. Moreau, and B. De Moor. Using literature and data to learn Bayesian networks as clinical models of ovarian tumors. *Artificial Intelligence in Medicine*, 30 (3):257–281, 2004.

M. Bartlett and J. Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

E. Y.-J. Chen, Y. Shen, A. Choi, and A. Darwiche. Learning Bayesian networks with ancestral constraints. In *Advances in Neural Information Processing Systems*, pages 2325–2333, 2016.

T.-T. Chen and S.-S. Leu. Fall risk assessment of cantilever bridge projects using Bayesian network. *Safety science*, 70:161–171, 2014.

D. M. Chickering, C. Meek, and D. Heckerman. Large-sample learning of Bayesian networks is NP-hard. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 124–133, 2003.

S. Dasgupta. Learning polytrees. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 134–141, 1999.

L. M. de Campos and J. G. Castellano. Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning*, 45:233–254, 2007.

M. J. Flores, A. E. Nicholson, A. Brunskill, K. B. Korb, and S. Mascaro. Incorporating expert knowledge when learning bayesian network structure: a medical case study. *Artificial intelligence in medicine*, 53(3):181–204, 2011.

7. https://www.westgrid.ca
8. www.computecanada.ca

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.

R. Giordano, D. D'Agostino, C. Apollonio, N. Lamaddalena, and M. Vurro. Bayesian belief network to support conflict analysis for groundwater protection: the case of the Apulia region. *Journal of environmental management*, 115:136–146, 2013.

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

R. J. Kennett, K. B. Korb, and A. E. Nicholson. Seabreeze prediction using Bayesian networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 148–153. Springer, 2001.

K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. CRC press, 2010.

C. Lee and P. van Beek. Metaheuristics for score-and-search Bayesian network structure learning. In *Proceedings of the 30th Canadian Conference on Artificial Intelligence*, pages 129–141, 2017. Available as: LNCS 10233.

S.-S. Leu and C.-M. Chang. Bayesian-network-based safety risk assessment for steel construction projects. *Accident Analysis & Prevention*, 54:122–133, 2013.

T.-Y. Ma, J. Y. Chow, and J. Xu. Causal structure learning for travel mode choice using structural restrictions and model averaging algorithm. *Transportmetrica A: Transport Science*, 13(4):299–325, 2017.

D. Oyen, B. Anderson, and C. M. Anderson-Cook. Bayesian networks with prior knowledge for malware phylogenetics. In *AAAI Workshop: Artificial Intelligence for Cyber Security*, 2016.

J. Peters and P. Bühlmann. Structural intervention distance for evaluating causal graphs. *Neural computation*, 27(3):771–799, 2015.

C. A. Pollino, O. Woodberry, A. Nicholson, K. Korb, and B. T. Hart. Parameterisation and evaluation of a Bayesian network for use in an ecological risk assessment. *Environmental Modelling & Software*, 22(8):1140–1152, 2007.

Ş. Ö. Şahin, F. Ülengin, and B. Ülengin. A Bayesian causal map for inflation analysis: The case of Turkey. *European Journal of Operational Research*, 175(2):1268–1284, 2006.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

M. B. Sesen, A. E. Nicholson, R. Banares-Alcantara, T. Kadir, and M. Brady. Bayesian networks for clinical decision support in lung cancer care. *PloS one*, 8(12):e82349, 2013.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 548–549, 2005.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.