Proceedings of Machine Learning Research vol 72, 133-144, 2018 PGM 2018

Structure Learning for Bayesian Networks over Labeled DAGs

Antti Hyttinen ANTTI.HYTTINEN@HELSINKI.FI
Johan Pensar JOHAN.PENSAR@HELSINKI.FI
Juha Kontinen JUHA.KONTINEN@HELSINKI.FI
Jukka Corander JUKKA.CORANDER@HELSINKI.FI
HIIT, Dept. of Computer Science, Dept. of Mathematics and Statistics, University of Helsinki, Finland

Abstract

Graphical models based on labeled directed acyclic graphs (LDAGs) allow for representing context-specific independence relations in addition to regular conditional independencies. Modeling such constraints has been demonstrated to be important for expressiveness, interpretation and predictive ability. In this paper, we build theoretical results that make constraint-based and exact score-based structure discovery possible for this interesting model class. In detail, we present the first constraint-based learning method for LDAGs. The orientation rules use context-specific independencies for principled orientation of additional (causal) edges. We also present the first exact score-based learning method for LDAGs, that employs a branch and bound for the especially computational demanding task of local score calculation, after which exact DAG search can be used. Simulations verify the good performance of our methods in different data analysis tasks.

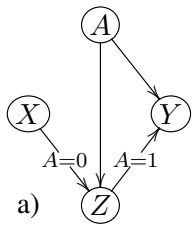
Keywords: directed graphical models; Bayesian networks; structure learning; context-specific independence; causal discovery.

1. Introduction

As evidenced by thousands of publications relating to their theory and applications, graphical models offer a versatile set of tools for data analysis, expert systems, etc. (Pearl, 1988). Many graphical models focus on statistical conditional independence, but context-specific independence (CSI) relations: $X \perp\!\!\!\perp Y|Z = 0$, meaning that $P(X|Y, Z = 0) = P(X|Z = 0)$ but (possibly) $P(X|Y, Z = 1) \neq P(X|Z = 1)$, forms another type of constraint worth investigating. For Bayesian networks, CSIs rise for example from local structure in the local conditional probability distributions (Koller and Friedman, 2009, chapter 5).

Labeled Directed Acyclic Graphs (LDAGs) of Pensar et al. (2015) are graphical models which support in addition to regular conditional and marginal independencies also context-specific independencies (Boutilier et al., 1996). This framework builds on Bayesian networks (Pearl, 1988) by adding labels to the arcs, specifying situations (contexts) in which the edge does not manifest itself in the child node distribution through statistical dependencies. There are several benefits arising from this added flexibility: 1) weak dependencies between variables that are only significant in some contexts are not overweighed by independencies in others and overlooked, 2) more theory (e.g. for edge orientation beyond DAG equivalence class) and separation criteria can be developed over the labels, 3) the increase in model complexity in terms of Bayesian scores due to adding edges can be smaller than for unlabeled networks.

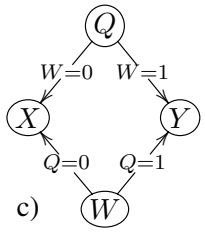
In this paper, we take on the challenging task of developing constraint-based and exact score-based structure learning for LDAGs. In addition to the super-exponential number of DAGs, we need to discover also labels for the edges, which means that the search space is an order of magnitude



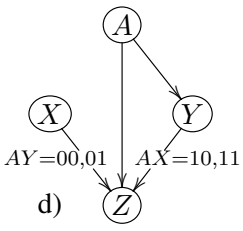
a)

b)

$P(Z\|A,X)$	$Z=0$	$Z=1$
$AX=00$	0.1	0.9
$AX=01$	0.1	0.9
$AX=10$	0.5	0.5
$AX=11$	0.6	0.4



c)



d)

Figure 1: Example binary LDAGs and a CPT for Z in a) respecting label $A = 0$ on $X \to Z$.

larger than for DAGs. Instead of the simple polynomial d-separation for DAGs, we need to deal with more complex separation criteria (Corander et al., 2016). The previous algorithm presented for learning LDAGs by Pensar et al. (2015) is a mixture of stochastic and greedy search procedures, without guarantees on the edge orientation or global optimality.

First, we develop previous theory on LDAGs (Section 2) further in Section 3, and derive a constraint-based learning algorithm in Section 4. In Section 5, we present an exact score-based method for learning LDAGs, which is further improved in Section 6. Related work is discussed in Section 7. Both of our methods are tested in Section 8. Conclusion is given in Section 9.

2. Bayesian Networks over Labeled DAGs

A labeled directed acyclic graph G (e.g. Fig. 1 a)) is a triple $(\mathcal{V}, \mathcal{E}, \mathcal{L})$, where $(\mathcal{V}, \mathcal{E})$ forms a DAG (Pensar et al., 2015). Importantly, the list $\mathcal{L}$ associates a label to each edge in $\mathcal{E}$. The CPTs of an LDAG can be parametrized similarly as for a DAG to form a graphical model (a Bayesian Network), the labels encode internal structure in these CPTs. The nodes correspond to categorical random variables with $n(X)$ values indexed as $0, \ldots, n(X) - 1$ (for node X).

For each edge $X \to Y \in \mathcal{E}$ there is a label $L \in \mathcal{L}$, which is a set of assignments[1] to $pa(Y) \setminus X$. Each assignment in the label encodes a local CSI: if $l \in L$, then $Y \perp\!\!\!\perp X | pa(Y) \setminus X = l$. This means that when $pa(Y) \setminus X = l$, the particular value of X does not affect Y: this is enforced in the CPT of Y by setting identical probabilities to all rows which have $pa(Y) \setminus X = l$. For example, the label $A = 0$ on $X \to Z$ in Fig. 1 a) implies that $X \perp\!\!\!\perp Z | A = 0$. The local CPT in Fig. 1 b) respects this label and the local independence: it has two first rows identical.

In addition to local CSIs, CSI-separation (Boutilier et al., 1996) gives a sufficient global criterion for verifying non-local CSIs. It is defined for LDAGs as follows.

Consider the context[2] $S = s$ and a label L on edge $X \to Y$. Let $A = (pa(Y) \setminus \{X\}) \cap S$ i.e. the nodes that appear both on the label and in the set S. Let $B = (pa(Y) \setminus \{X\}) \setminus S$, i.e. the nodes that appear on the label L but not in S. Let a be an assignment of A consistent with $S = s$. A label L is satisfied by a context $S = s$ iff $(a, b) \in L$ for all possible assignments b. Now, an LDAG can be turned into a *context $S = s$ specific DAG* by removing edges whose labels are satisfied by $S = s$. For example, the context $A = 0$ specific DAG of Fig. 1 d) is identical to the underlying DAG except for edge $X \to Y$ being absent due to its satisfied label.

The CSI-separation criterion gives a sufficient condition for an independence to be implied by an LDAG structure: If nodes X, Y are d-separated given C, S in the context $S = s$ specific DAG of G, then $X \perp\!\!\!\perp Y | C, S = s$ is implied by G. Note that regular d-separation is a special case when

1. That only $pa(Y) \setminus X$ appear in the label for $X \to Y$ encodes a causally sensible restriction: only $pa(Y)$ take part in the local process of generating a value for Y, thus the other nodes should not dictate the role of X in this process.
2. Context is formally a particular assignment to random variables corresponding to nodes of the LDAG, e.g. $A = 0$, $S = s$ or $\mathcal{V} = v$. The assignments given by some context $\mathcal{V} = v$ to set $S \subset \mathcal{V}$ is marked by $v[S]$.

$S = \emptyset$, and thus obviously valid for LDAGs. For example, the labeling in Figure 1 a) implies by this criterion that $X \perp\!\!\!\perp Y | A = 1$, as $Z \to Y$ does not appear in the context $A = 1$ specific DAG.

Finally, throughout the paper, we restrict our attention to regular maximal LDAGs. *Maximality* requires that all labels that follow from other labels are recorded in the edges. *Regularity* means that edges absent in every context are not included in the graph. See Pensar et al. (2015) for details.

3. A Necessary Separation Criterion for Labeled DAGs

CSI-separation gives only a sufficient condition for independence, for example the LDAG in Figure 1 a) actually implies $X \perp\!\!\!\perp Y$ (Corander et al., 2016), although CSI-separation does not indicate this. The axiomatic approach in Corander et al. (2016) can infer that the independence is implied (conjectured to be complete) in this case, but their approach does not give any indication on how to proceed with structure discovery. The following gives a novel necessary criterion for an independence to be implied by an LDAG structure: d-separation at least in some context is needed.

Theorem 1 *If there is a d-connecting path btw. X, Y given S, C in any context $\mathcal{V} = v$ specific DAG of an LDAG G, then independence $X \perp\!\!\!\perp Y | S = v[S], C$ is not implied by G.*

Proof Appendix A specifies parameters such that $X \not\perp\!\!\!\perp Y | S = v[S], C$ in the gen. distribution. ■

Nodes d-connected in the underlying DAG but not d-connected in any context $\mathcal{V} = v$ specific DAG are still an issue: the variables may be implied independent or dependent. For example, in Figure 1 a) $X \perp\!\!\!\perp Y$ is implied, and the d-connecting path disappears in all contexts. Figure 1 c) shows an example where X, Y may be dependent, but the d-connections disappear in all contexts.

Motivated by Theorem 1, we continue by defining LDAG-colliders, which correspond to (unshielded) colliders in Bayesian networks. We also define LDAG-non-colliders.

Definition 2 (LDAG-(non-)collider) *A triple (X, Y, Z) is an LDAG-collider in an LDAG iff there is a context $\mathcal{V} = v$, s.t. the context $\mathcal{V} = v$ specific DAG has $X \to Y \leftarrow Z$, and X, Z non-adjacent. A triple (X, Y, Z) is an LDAG-non-collider in an LDAG iff there is a context $\mathcal{V} = v$, s.t. the context $\mathcal{V} = v$ specific DAG has $X \leftarrow Y \leftarrow Z$, $X \to Y \to Z$ or $X \leftarrow Y \to Z$, and X, Z non-adjacent.*

For example, the triple (X, Z, Y) in Figure 1 d) is not an LDAG-collider: for contexts where $A = 0$, the edge $X \to Z$ is missing and for contexts where $A = 1$ the edge $Y \to Z$ is missing. In Figure 1 a) the triple (X, Z, A) is an LDAG-collider: it appears as a collider in contexts where $A = 1$.

Pensar et al. (2015) have shown that LDAGs are Markov equivalent if and only if they have the same Markov equivalent (applied to DAGs) context-specific DAGs. This implies that Markov equivalent LDAGs have the same skeleton, the same set of LDAG-colliders and the same set of LDAG-non-colliders. Although Fig. 1 a) and d) do not have Markov equivalent (applied to DAGs) underlying DAGs, they are in fact Markov equivalent LDAGs: the triple (X, Z, Y) is neither an LDAG-collider nor an LDAG-non-collider in either LDAG.

4. Constraint-based learning with the LPC Algorithm

In this section, we present a new constraint-based causal discovery algorithm that is able to correctly discover LDAGs. Throughout, we assume that we have the true distribution available, i.e. we do not concern ourselves with finite sample data producing erroneous test results (we prefer to use the score-based approach introduced later in those cases for better accuracy). Algorithm 1 (left) shows

procedure LPC
Run LPC-SKELETON.
Detect LDAG-colliders.
Run orientation rule R0.
Detect LDAG-non-colliders.
Run orientation rules R1-R4
    until no edges can be oriented.
Add labels to edges.
Return the repr. of the Markov eq. class.

procedure LPC-SKELETON
Initialize G as full undirected graph.
Initialize all $\mathrm{sepcon}(\cdot,\cdot)$ as empty.
for set size k in $0\ldots$ **do**
    for node X, node $Y \in \mathrm{nb}_G(X)$ **do**
        for set $S \in \mathrm{nb}_G(X) \setminus Y$, $|S| = k$ **do**
            for assignment s for S **do**
                if $X \perp\!\!\!\perp Y | S = s$ **then**
                    Add $S = s$ to $\mathrm{sepcon}(X, Y)$.
            if $\forall s : X \perp\!\!\!\perp Y | S = s$ **then**
                Delete edge $X - Y$ from G.
Return G, $\mathrm{sepcon}(\cdot,\cdot)$.

Algorithm 1: Left: The LPC algorithm. Right: The LPC skeleton search.

the steps of the LPC algorithm, which build on the PC algorithm of Spirtes et al. (2000). Each step is explained in the subsections.

We cannot assume the regular faithfulness (Spirtes et al., 2000) here, since labels can imply additional independencies beyond those that can be discovered using d-separation (remember $X \perp\!\!\!\perp Y$ for Fig. 1 a)). We need to make the following faithfulness assumption, directly based on Theorem 1.

Assumption 1 (LDAG-faithfulness) *If there is a d-connecting path btw. X, Y given S, C in any context $\mathcal{V} = v$ specific DAG of an LDAG G, then $X \not\perp\!\!\!\perp Y | S = v[S], C$.*

In our simulations, where parameters were sampled randomly from Dirichlet distributions, we did not encounter violations of this condition.

4.1 LPC Skeleton Search

Algorithm 1 (right) shows the steps of the skeleton search. This modifies the skeleton search of PC (Spirtes et al., 2000) by testing independencies in contexts, i.e. $X \perp\!\!\!\perp Y | S = s$ instead of $X \perp\!\!\!\perp Y | S$. It saves all found separating contexts as $\mathrm{sepcon}(\cdot,\cdot)$. Since LDAG implies independencies corresponding to d-separations in its underlying DAG, any absent $X \to Y$ will be detected as we have that $\exists S : \forall s : X \perp\!\!\!\perp Y | S = s$ (S can be the parents of Y). For any present $X \to Y$, because of regularity and faithfulness, there is a context $\mathcal{V} = v$ s.t. for every S we have that $X \not\perp\!\!\!\perp Y | S = v[S]$ and thus $X \to Y$ will not be deleted. Thus this step will return the correct skeleton.

4.2 LDAG-(non-)collider Detection

The first orientation rule for regular PC is to orient $X - Y - Z$ as $X \to Y \leftarrow Z$ if Y is not in the separating set for X, Z. Unfortunately, this is not correct when considering LDAGs. E.g. it would incorrectly orient $X \to Z \leftarrow Y$ for the LDAG in Fig. 1 a) since $X \perp\!\!\!\perp Y$. Thus, the orientation rules of PC (Meek, 1995) must be revised in order for them to also be valid for LDAGs. The following theorems are needed for detecting LDAG-(non-)colliders in the LPC skeleton search result.

Theorem 3 (LDAG-(non-)collider detection) *A triple (X, Y, Z) is an LDAG-collider (LDAG-non-collider) iff there is a context $\mathcal{V} = v$ and a set S that satisfy the requirements in the LPC skeleton result: 1) $S = v[S] \in \mathrm{sepcon}(X, Z)$, 2) $\forall (S' = s') \in \mathrm{sepcon}(X, Y) : v[S'] \neq s'$, 3)*

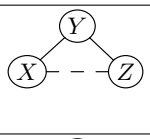
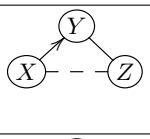
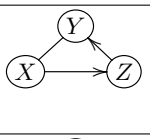
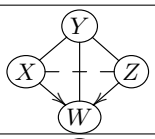
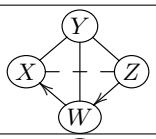
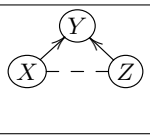
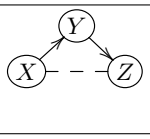
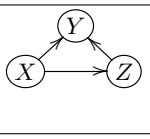
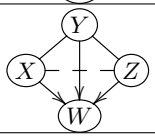
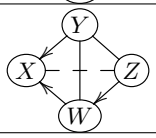
Rule	R0	R1	R2	R3	R4
(X, Y, Z)	LDAG-collider	LDAG-non-collider		LDAG-non-collider	LDAG-non-collider
Pattern					
Result					

Table 1: Orientation rules of LPC. In the patterns, all regularly drawn edges must be present, dashed edges may be present or absent. Undirected edges in the patterns may be also directed. Additional edges may be present to nodes not drawn.

$(S'' = s'') \in \mathrm{sepcon}(Z, Y) : v[S''] \neq s''$*, and either 4a)* $Y \notin S$ *for an LDAG-collider or 4b)* $Y \in S$ *for an LDAG-non-collider.*

The detected LDAG-colliders can be oriented when found, this is R0 in Table 1. The detected LDAG-non-colliders are used in the more intricate orientation rules of the next section.

4.3 Orientation Rules

Table 1 shows the orientation rules for LPC. They are modified from Meek (1995), but there are several differences. First, in most rules we require triple (X, Y, Z) to be detected as a collider or non-collider by the theorems; the corresponding requirement regarding (regular) colliders is built into PC implicitly. Second, the rules allow for X and Z to be adjacent, as long as the joining edge disappears in contexts such that the LDAG-collider-non-collider requirement is fulfilled. Third, PC requires rule R4 only in the presence of background knowledge of some orientations (Meek, 1995), in LPC we get extra orientations due to CSIs and the rule is very important and used regularly also without background knowledge. Meek (1995) showed that his orientation rules are complete also under background knowledge. Given that detected LDAG-colliders and LDAG-non-colliders can be seen as background knowledge, we conjecture that LPC algorithm is also orientation complete.

4.4 Labeling

We are left with the task of adding labels to edges, only some of which are oriented. This is achieved by recording the contexts in which X, Y were found independent to the label associated with the edge $X - Y$. Given any fully determined orientation, we can find the labels corresponding to local CSIs (see Section 2) for that particular member (LDAG) of the equivalence class.

4.5 Examples

Figure 2 shows two results of the LPC algorithm. Fig. 2 a) is the result when the data generating model has the LDAG structure given in Fig. 1 a). Note that this result includes the (LDAG) Markov equivalent structure Fig. 1 d). As described earlier regular PC would orient $X \rightarrow Z \leftarrow Y$ incorrectly to the true structure Fig. 1 a). When the generating LDAG structure is Fig. 2 b) the result

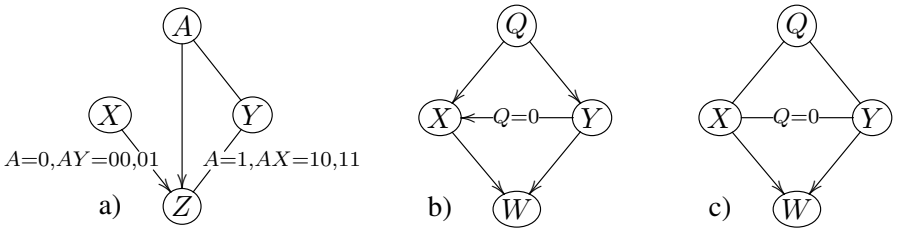


Figure 2: Example results of the LPC algorithm.

of the LPC algorithm is Fig. 2 c). Here PC would leave all edges unoriented, whereas LPC uses $X \perp\!\!\!\perp Y | Q = 0$ to correctly orient the edges from X and Y into W.

5. Exact Score-based Learning for Labeled DAGs

Although the LPC algorithm is valuable in that it outputs a characterization of an equivalence class, its results can be far from accurate when run on finite sample data. Instead, because the labels can produce local minima, we find the globally optimal LDAG by exact score-based learning[3]. Finding the LDAG maximizing the global BIC score can be cast as (similarly as in the derivation by Chickering et al. (1997)):

$$\max_G \sum_{X \in V} s(X, \mathrm{pa}_G(X)), \quad \text{where} \quad s(X, \mathrm{pa}_G(X)) = \max_L s(X, \mathrm{pa}_G(X), L)$$

Here $s(X, \mathrm{pa}_G(X), L)$ is the local BIC score of the CPT of X, with rows partitioned according to labels L. Naturally for a network without labels the score is exactly equal to DAG-based BIC of the DAG. This learning problem was also considered by Pensar et al. (2015), however, whereas they used a stochastic greedy approach, we present here an exact approach. The above formulation divides this computationally hard task into two tasks: DAG structure learning and the search for optimal label structure for a local CPT. We first concentrate on the latter task of calculating local scores. This is only currently possible for maximum parent set size 4, depending on the number of categories for the variables. At worst case we need to search through a vast number of CPTs: for a node with 4 binary parents, we have $27\ 202\ 841$ CPTs possible for a regular maximal LDAG.

5.1 Local Score Calculation

Our procedure for calculating local scores is given in Algorithm 2. Instead of a naive search over labels, we search over partitions P of rows in a local CPT similarly as Pensar et al. (2017). To cut down calculation times for unfavorable partitions, we use a branch and bound search (applied also to similar tasks by Suzuki (1996); Tian (2000); Malone and Yuan (2014); van Beek and Hoffmann (2015); Suzuki and Kawahara (2017)). We start from the case where all parts are single rows, and combine rows further in the branches. On lines 9 and 10 in BB, we first search the branches where p_{f+1} is combined to some preceding part and then when p_{f+1} is not combined to a preceding part.

Upper bound for a branch The BIC score we use is the sum of likelihood L and a penalization term for the number of parameters. Since we are combining more and more rows (and forming a

3. This produces a single representative of the equivalence class, similarly as score-based learning for Bayesian networks. The properties of this optimal equivalence class can be examined in several ways, e.g. with LPC of Sect. 4.

procedure LSCORE(Y, S)
Set initial bound B (Sect. 5.1).
Initialize partition P as singleton rows.
BB($Y, S, P, 1, B$).

procedure BB(Y, S, P, f, B)
Calculate likelihood L according to P.
if $(\mathrm{L} - |P|(n(Y)-1)\log N/2 > B)$
  $\wedge \quad IsLDAG(Y, S, P)$ **then**
  $B := \mathrm{L} - |P|(n(Y)-1)\log N/2$.
if $\mathrm{L} - f \cdot (n(Y)-1)\log N/2 < B$ **then**
  Return.
for i=1…f **do**
  BB($Y, S, [\cdots, p_i \cup p_{f+1}, \cdots], f, B$).
BB($Y, S, P, f+1, B$).

procedure ISLDAG(Y, S, P)
for part $p \in P$ **do**
  Set $K = \{p[1]\}$.
  for $p' \in K$ **do**
    for $X \in S$ **do**
      if $\forall x : (p'[S \backslash X], x) \in p$ **then**
        add them to K.
  if $K \neq p$ **then** return FALSE.
Return TRUE.

Algorithm 2: Local score calculation for node Y and parent set S.

simpler model) when going deeper in the search branch, we know that the current likelihood term is equal or higher than for any partition further down in the branch. The number of parameters used can go as low as $n(Y) - 1$ deeper in the branch. We keep an index f on how many elements of the partition are fixed, i.e. they are not going to be combined further (although parts can be merged to them). The idea is to have a lower bound on the number of parameters. Initially we set $f = 1$, as the first element can be considered to be fixed. Since f parts are kept fixed we get a tighter bound: $\mathrm{L} - f \cdot (n(Y) - 1)\log N/2$. If this is less than the current best score B, the branch can be closed.

Initial lower bound Since different rows in the local CPT can be combined, the search for the optimal labels for node Y from parents S will (essentially) search through the best solutions for all subsets of S as well. To prevent this we will perform local score calculation from smaller parent sets to larger ones. Then we can give the initial lower bound B for node Y and set S as the best score from previously calculated local scores for proper subsets of S.

LDAG consistency check Not all partitions are consequences of adding labels to the edges. For example, for two binary parents, partition $\{\{00, 11\}, \{01\}, \{10\}\}$ cannot be explained by adding labels: configurations 00 and 11 can only belong to the same part if either 01 or 10 belongs to it as well. IsLDAG in Algorithm 2 performs an LDAG consistency check by checking each part of a partition individually. We start with one row, and consider any label on $X \to Y$ that would set rows equal and always add *all* other rows if they are present in the part. If in the end some rows in a part are left outside the set K, we know the partition is not consistent with an LDAG. We check for LDAG consistency only when we find a solution whose score is higher than the current best score.

5.2 Finding the DAG structure

Given a set of LDAG local scores, we can simply run Bayesian network structure learning. We use Gobnilp (Bartlett and Cussens, 2017): it does not use DAG-based symmetries that may be invalid for LDAGs and it benefits from a limit on the number of parents, which is essential also for the LDAG-based local score calculation.

6. Enforcing Strong Regularity

Remember that regularity implies that there are no edges absent in every context. After standard score pruning: deleting local score $s(X, S)$ if $\exists S' \subset S : s(X, S') \geq s(X, S)$ (Campos and Ji, 2011), the optimal LDAG structures will be regular[4]. However, score-based learning can still produce structures that are almost irregular: they have edges with a lot of labels such that the edges are only present in few, very low probability contexts. This overfitting phenomenon, also observed by Pensar et al. (2015), is visible in our simulations (Sect. 8). We suggest three remedies for this issue.

Strong Score Pruning Since standard score pruning takes irregular models out of consideration, we can take out almost irregular models by performing *strong score pruning*. We delete local score $s(Y, S)$ if $\exists S' \subset S : s(Y, S') + \gamma \geq s(Y, S)$. That is, we delete also local scores that are only slightly better than the score for some subset. With increasing parent set size, the required score improvement should get higher. Thus, we use $\gamma = t \cdot (n_S - n_{S'}) \log N/2$ where $n_S, n_{S'}$ refer to the number of parameters for a DAG (without labels) for Y with parent sets S and S'. Note that after strong score pruning, the found LDAG structures are optimal only over the pruned scores.

Mixed BIC Penalty Instead of LDAG-based BIC score, we also test $s'(X, S, P) = \mathrm{L} - a \cdot |P|(n(X) - 1) \cdot \log N/2 - b \cdot n_S \cdot \log N/2$ where n_S is the number of parameters in the BN local CPT without labels. We retain the equal score for a DAG without labels to DAG-based BIC by setting $a + b = 1$.

LDAG over Optimal DAG Skeleton As a final option, we first find the optimal DAG structure with the DAG-based BIC score, and then find the optimal LDAG structure over the skeleton. Thus, only orientation is done with the LDAG-based BIC score.

7. Related Work

Over the last decades, there has been a lot of research pertaining to lift the often unnecessarily strict conditional independence based restrictions implied by standard DAG based models (Boutilier et al., 1996; Geiger and Heckerman, 1996; Chickering et al., 1997; Friedman and Goldszmidt, 1998; Poole and Zhang, 2003; DesJardins et al., 2008; Smith and Anderson, 2008; Barclay et al., 2013; Pensar et al., 2015, 2016, 2017). The benefit from allowing more flexible model structures is the possibility of more accurately modelling the underlying data generating process. In terms of structure learning, the added flexibility has been shown to improve the predictive properties of the inferred models (Friedman and Goldszmidt, 1998; DesJardins et al., 2008; Pensar et al., 2015, 2016), however, it also puts higher demands on the learning algorithms to retain scalability and avoid overfitting. For Bayesian networks in particular, structured CPTs in which collections of local conditional distributions are assumed identical have become increasingly popular (Koller and Friedman, 2009, Chapter 5). One reason for this is that standard scores can still be evaluated in closed form (Chickering et al., 1997; Friedman and Goldszmidt, 1998). The perhaps most well-known type of structured CPTs are those explained by CSI (Boutilier et al., 1996). CSIs have also been used to improve the efficiency of inference algorithms (Boutilier et al., 1996; Poole and Zhang, 2003).

4. If the optimal LDAG is irregular, for a node X that has parents $pa(X)$, different values for some $Y \in pa(X)$ do not alter $p(X|pa(X))$. This implies that $pa(X)$ would be pruned as $pa(X)/Y$ will get the same (or better) BIC score.

algo	models	av. degree	label prob.	edges found	corr. oriented	oriented %	reversed	reversed %
LPC	300	2.99	0 %	4481	3498	78 %	0	0 %
PC	300	2.99	0 %	4481	3498	78 %	0	0 %
cPC	300	2.99	0 %	4481	3498	78 %	0	0 %
LPC	300	2.9	25 %	4343	3387	78 %	0	0 %
PC	300	2.9	25 %	4343	3340	77 %	27	1 %
cPC	300	2.9	25 %	4343	3345	77 %	0	0 %
LPC	300	2.18	50 %	3276	2319	71 %	0	0 %
PC	300	2.18	50 %	3276	2243	68 %	103	4 %
cPC	300	2.18	50 %	3276	2285	70 %	0	0 %

Table 2: LPC, PC and cPC orientation results. 10-node binary LDAGs.

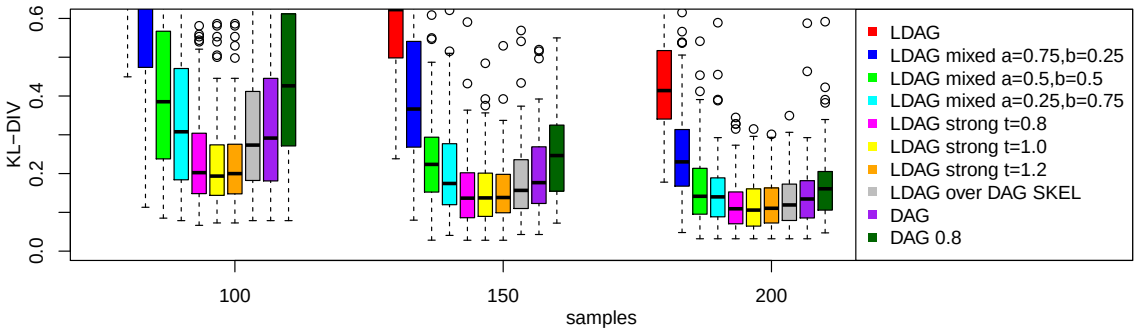


Figure 3: KL-divergence to the true distribution. 10-node binary LDAGs.

8. Simulations

We implemented the LDAG theory and the LPC algorithm in R. The computationally demanding task of local score calculation was implemented in C++. For the first two tests we used simulated LDAGs with 10 nodes. Edges were drawn at random such that the mean degree of a node in underlying DAG was 3. The labels were also drawn randomly with probability 0.5 if not stated otherwise. Parameters for each row of the CPTs were drawn from Dirichlet (ESS=1), redrawing if probabilities were not in $[0.05, 0.95]$. The true networks were maximal and regular (after a regularization procedure, this slightly lowered the mean degree).

Table 2 shows the result comparing the orientation produced by the LPC algorithm to PC and cPC as implemented in PCALG-package (Kalisch et al., 2012), run on the true distribution produced by the generating model. When no labels are present in the generating model, all algorithms behave similarly. When the probability of labels increases, PC starts producing incorrect orientations. Interestingly, cPC (Ramsey et al., 2006) does well at noticing these problems, it does not orient the edges where PC is at fault. However, compared to LPC, cPC orients fewer edges. LPC orients more edges due to found CSIs and all of its orientations are correct.

Then we examined the accuracy of LDAGs as predictive graphical models, when only few samples are available. We sampled data from the BNs over LDAGs. We learned LDAG or DAG structure by different *exact*[5] methods and then estimated the parameters (with regularization). We evaluated the quality of the probabilistic models by comparing the KL-divergence between the distribution of the (optimal) graphical model and the true distribution. Figure 3 shows the result. First, the optimal

5. Due the stochastic and greedy nature of the non-exact method by Pensar et al. (2015), the results of that algorithm would depend very highly on the sensitive tuning parameters and the time it is allowed to run.

LDAG identified under a mere LDAG-based BIC penalty ('LDAG') severely overfits the data, the KL-divergence is high even with 200 samples. The suggested mixed penalty term ('mixed') corrects this overfitting to some extent. Strong score pruning ('strong') does even better especially for 100 samples. Here the parameter t has no strong effect. Optimal DAGs ('DAG') are not as good with 100 data points, even if we introduce a coefficient in front of the BIC penalty term ('DAG 0.8'). Although the label accuracy of LDAGs gets better with increasing sample size, DAG-based prediction catches up in accuracy when using this measure.

Finally, we demonstrate the scalability of exact LDAG learning on 1000 samples from the well studied 37-node Alarm network (Beinlich et al., 1989). This real-life data set is especially interesting since the available CPTs include many local CSIs that correspond to labels e.g. $\mathrm{HREKG} \perp\!\!\!\perp \mathrm{CRRCAUTER} \mid \mathrm{HR} = \mathrm{LOW}$ (indexes 10, 11, 35 respectively). Note that the variables have up to 4 categories, resulting in additional computational cost over binary LDAGs. We used 100s timeout for calculating individual local scores, maximum parent set size 4 and employed strong score pruning with $t = 1.0$. The task of calculating the around 2.5 million local scores took 7 hours on a modern desktop computer[6]. Fortunately a majority of the scores were pruned out by the applied strong score pruning. The subsequent run of Gobnilp took only less than 10 seconds. Figure 4 shows the result. Some edges are missed, e.g. the learned network is unable to find node 13 as parent of node 34, but most adjacencies are found correctly for this sample size.

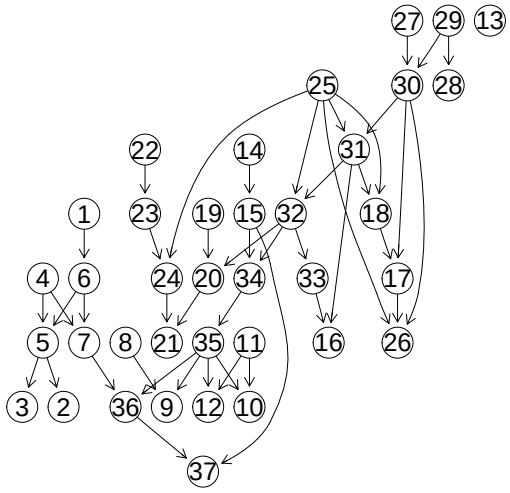


Figure 4: The learning result for the Alarm data set.

9. Conclusion

We presented new theory on labeled DAGs and developed both constraint-based and exact score-based structure learning methods. Simulations showed improved edge orientation and accuracy of probabilistic models. An attractive target for future research is to apply these findings to the discovery of causal structures with latent variables, which poses further steep challenges. The boosted ability of context-specific independencies to orient edges suggests they may be valuable for finding additional orientation and improving the accuracy of learning and inference also for such models.

Appendix A. Proofs

Proof of Theorem 1 Suppose there is a shortest d-connecting path btw. X, Y given S, C in a context $\mathcal{V} = v$ specific DAG of an LDAG G. We give here a parametrization for binary variables such that in the generated distribution $X \not\perp\!\!\!\perp Y | S = v[S], C$. The generalization to n-ary variables is straightforward. We can assume v is all zeros and also drop conditioning on C, since $C = v[C]$ can be added to $S = v[S]$. The path consist of sources, sinks and intermediate nodes such that only the sinks are in S. Consider for example the graph $X \leftarrow Q \rightarrow W \leftarrow R \rightarrow Y, W \rightarrow B$, that d-connects

6. We are not yet employing advanced techniques such as AD-trees in the calculation, but only simple caching of counts. Some speedups may be possible here.

X and Y given $B = v[B]$. Fill in the directed paths from sources to sinks as follows. The chosen letters directly refer to the example. 1) Set nodes not appearing on the path to have values according to v with probability 1. This will isolate the path and make all edges in the path active. 2) Put all sources on the path to have a uniform distribution. In the example $P(Q = 1) = P(Q = 0) = 0.5$ and $P(R = 1) = P(R = 0) = 0.5$. 3) For directed edges $Q \to X$ put $X = Q$. (In the example such parametrization is given also for $W \to B$ and $R \to Y$.) 4) For structures $Q \to W \leftarrow R$, set CPT of W s.t. $P(W = 0|Q = 0, R = 0) = 0$ and the remaining probabilities > 0 (exact values may depend on whether possible labels $Q = 1$ on $W \leftarrow R$ and $R = 1$ on $Q \to W$ exist). The parametrization is such that along paths nodes get the same value and therefore form a dependence, in the presence of a collider, such as in the example, we have $P(W = 0) > 0$, $P(X = 0, Y = 0|B = 0) = P(Q = 0, R = 0|W = 0) = P(W = 0|Q = 0, R = 0) \cdot 0.25/P(W = 0) = 0$ and $P(X = 0|B = 0) = P(Q = 0, R = 1|W = 0) = P(W = 0|Q = 0, R = 1) \cdot 0.25/P(W = 0) > 0$, and similarly $P(Y = 0|B = 0) > 0$, hence $X \not\perp\!\!\!\perp Y|B$. ■

Proof of Theorem 3 Suppose we have context $V = v$ and a set S satisfying the conditions of Theorem 3. In particular $S = v[S] \in \mathrm{sepcon}(X, Z)$ implies that we found $X \perp\!\!\!\perp Z|S = v[S]$. Suppose $X - Y - Z$ is oriented as $X \leftarrow Y \to Z$ in the true LDAG. This would mean by Theorem 1 that $X \perp\!\!\!\perp Z|S = s$ is not implied (due to $X \leftarrow Y \to Z$), unless either $X \leftarrow Y$ or $Y \to Z$ is absent in the context $V = v$ specific DAG of the true LDAG, for every context $V = v$ (s.t. $v[S] = s$). So suppose $X \leftarrow Y$ is deleted in context $V = v$. This means the label is satisfied under context $pa(X)\backslash Y = v[pa(X)\backslash Y]$ and hence we would have local CSI $X \perp\!\!\!\perp Y|pa(X)\backslash Y = v[pa(X)\backslash Y]$. This would be detected by labeled skeleton search as $pa(X) \setminus Y = v[pa(X) \setminus Y] \in \mathrm{sepcon}(X, Y)$. This cannot happen if all contexts in $sepcon(X, Y)$ disagree with $V = v$. Other orientations forming a non-collider can be similarly ruled out. The proof for LDAG-non-colliders is analogous. ■

Acknowledgments

This research was supported by Academy of Finland through grants 295673, 308712 and 251170.

References

L. Barclay, J. Hutton, and J. Smith. Refining a Bayesian network using a chain event graph. *International Journal of Approximate Reasoning*, 54(9):1300–1309, 2013.

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

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *AIME 89*, pages 247–256. Springer Berlin Heidelberg, 1989.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. UAI*, pages 115–123, 1996.

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

D. M. Chickering, D. Heckerman, and C. Meek. A bayesian approach to learning Bayesian networks with local structure. In *Proc. UAI*, pages 80–89, 1997.

J. Corander, A. Hyttinen, J. Kontinen, J. Pensar, and J. Väänänen. A logical approach to context-specific independence. In *Proc. Logic, Language, Information, and Computation: 23rd Interna-*

tional Workshop, WoLLIC 2016, pages 165–182, 2016.

M. DesJardins, P. Rathod, and L. Getoor. Learning structured Bayesian networks: combining abstraction hierarchies and tree-structured conditional probability tables. *Computational Intelligence*, 24(1):1–22, 2008.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Learning in Graphical Models*, volume 89. Springer, 1998.

D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82(1):45–74, 1996.

M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

B. M. Malone and C. Yuan. A depth-first branch and bound algorithm for learning optimal Bayesian networks. In *GKR 2013 Revised Selected Papers*, volume 8323 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 2014.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proc. UAI*, pages 403–410, 1995.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Francisco, CA, 1988.

J. Pensar, H. J. Nyman, T. Koski, and J. Corander. Labeled directed acyclic graphs: a generalization of context-specific independence in directed graphical models. *Data Min. Knowl. Discov.*, 29(2): 503–533, 2015.

J. Pensar, H. Nyman, J. Lintusaari, and J. Corander. The role of local partial independence in learning of Bayesian networks. *Int. Journal of Approximate Reasoning*, 69:91–105, 2016.

J. Pensar, J. Kohonen, and J. Corander. Exact bayesian learning of partition directed acyclic graphs. In *Proceedings of the International Meeting on High-Dimensional Data-Driven Science (HD3-2017)*, Journal of Physics, 2017.

D. Poole and N. L. Zhang. Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research*, 18:263–313, 2003.

J. Ramsey, J. Zhang, and P. Spirtes. Adjacency-faithfulness and conservative causal inference. In *Proc. UAI*, pages 401–408. AUAI Press, 2006.

J. Smith and P. Anderson. Conditional independence and chain event graphs. *Artificial Intelligence*, 172(1):42–68, 2008.

P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, prediction, and search*. MIT press, 2000.

J. Suzuki. Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the b & b technique. In *ICML*, 1996.

J. Suzuki and J. Kawahara. Branch and Bound for regular Bayesian network structure learning. In *Proc. UAI*. AUAI Press, 2017.

J. Tian. A branch-and-bound algorithm for MDL learning Bayesian networks. In *Proc. UAI*, pages 580–588. Morgan Kaufmann, 2000.

P. van Beek and H. Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Proc. CP*, volume 9255 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2015.