Comparing Loop Cutsets and Clique Trees in Probabilistic Inference

*Linda C. van der Gaag & Hans L. Bodlaender**
Department of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
e-mail: {linda,hansb}@cs.ruu.nl

Abstract

More and more knowledge-based systems are being developed that employ the framework of *Bayesian belief networks* for reasoning with uncertainty. Such systems generally use for probabilistic inference either the algorithm of J. Pearl or the algorithm of S.L. Lauritzen and D.J. Spiegelhalter. These algorithms build on different graphical structures for their underlying computational architecture. By comparing these structures we examine the complexity properties of the two algorithms and show that Lauritzen and Spiegelhalter's algorithm has at most the same computational complexity as Pearl's algorithm.

1 Introduction

For reasoning with uncertainty in knowledge-based systems, the framework of *Bayesian belief networks* is rapidly gaining in popularity [Pea88]. The framework provides a powerful formalism for representing a joint probability distribution on a set of statistical variables and offers algorithms for probabilistic inference. Since its introduction in the late 1980s, the belief-network framework has demonstrated its potential in various complex problem domains, ranging from medical diagnosis and prognostic assessment to probabilistic information retrieval and computer vision.

The formalism of Bayesian belief networks basically is a graphical formalism: a belief network comprises an *acyclic directed graph* representing statistical variables as vertices and probabilistic interdependences among variables as arcs. The topological properties of the digraph of a belief network determine to a large extent the computational complexity of probabilistic inference with the network. In general, the sparser the digraph, the less

*The research of this author was partially supported by ESPRIT Long Term Research Project 20244 (project ALCOM IT: *Algorithms and Complexity in Information Technology*).

complex inference is. For belief networks without any topological restrictions, probabilistic inference is known to be NP-hard [Coo90].

Several different algorithms have been designed for probabilistic inference with a Bayesian belief network, each having a worst-case computational complexity that is exponential in the number of vertices in a network's digraph. The best known are the algorithm of J. Pearl [Pea88] and the algorithm of S.L. Lauritzen and D.J. Spiegelhalter [LS88]. These algorithms build on different computational architectures that are constructed from the digraph of a belief network. Pearl's algorithm builds on a digraph more or less directly, after having 'cut' all its loops by appropriate vertices. The algorithm of Lauritzen and Spiegelhalter builds on a tree of cliques that is constructed from the digraph of a network after triangulation. The different graphical structures underlying these architectures give rise to different complexity properties for the two algorithms: while Pearl's algorithm has a computational complexity that is exponential in the number of vertices that are selected for cutting all loops in a digraph, the complexity of Lauritzen and Spiegelhalter's algorithm relates exponentially to the numbers of vertices in the separate cliques in the clique tree constructed for the graph.

In this paper, we examine the algorithm of Pearl and the algorithm of Lauritzen and Spiegelhalter as to their complexity properties. To this end, we build on an earlier result relating the two algorithms [SAS94]. By comparing the graphical structures underlying the algorithms' computational architectures, we show that Lauritzen and Spiegelhalter's algorithm has at most the same complexity as Pearl's algorithm. More specifically, we show that for belief networks for which Pearl's algorithm has a linear complexity, Lauritzen and Spiegelhalter's algorithm can also take linear time. We also show that the reverse property does not hold, that is, there are belief networks for which the algorithm of Lauritzen and Spiegelhalter has a linear computational complexity and Pearl's algorithm exhibits exponential behaviour at best. We conclude that in general Lauritzen and Spiegelhalter's algorithm will outperform Pearl's algorithm.

The paper is organised as follows. In Section 2, we briefly review the formalism of Bayesian belief networks. We present the two algorithms for probabilistic inference with a belief network outlined above in Section 3. In Section 4, we compare the graphical structures underlying the computational architectures employed by the algorithms. The paper is rounded off with our conclusions in Section 5.

2 Bayesian belief networks

A *(Bayesian) belief network* is a concise representation of a joint probability distribution on a set of statistical variables. In a belief network, the independences among the variables discerned and the numerical quantities involved in a distribution are represented separately [Pea88]. Independences among variables are represented by an *acyclic directed graph*. In this digraph, each vertex v_i models a statistical variable that can take one of a finite set of values. In our complexity statements, we will assume all variables to be binary; the results presented, however, are generalised straightforwardly. We take an arc $v_i \rightarrow v_j$ in

the digraph to represent a direct causal relationship between the variables v_i and v_j; the direction of the arc designates v_j as the effect of the cause v_i. More formally, the set of arcs of a belief network's digraph is assigned a probabilistic meaning: absence of an arc between two vertices expresses that the corresponding variables are (conditionally) independent.

Associated with the digraph of a belief network is a set of functions representing numerical quantities from the probability distribution that is being represented: with each vertex is associated a function which basically is a set of (conditional) probabilities describing the influence of the values of the vertex' (immediate) predecessors in the digraph on the probabilities of the values of the vertex itself. These functions with each other provide all information necessary for *uniquely* defining a joint probability distribution on the represented variables that respects the independences portrayed by the network's digraph.

3 Algorithms for probabilistic inference

A Bayesian belief network is generally used for *probabilistic inference*, that is, for making probabilistic statements concerning the variables represented in the network. Since a belief network uniquely defines a joint probability distribution, it provides for computing any probability of interest. Several different algorithms have been designed for this purpose. The best known among these are the algorithm of J. Pearl [Pea88] and the algorithm of S.L. Lauritzen and D.J. Spiegelhalter [LS88]. We review these algorithms along with their complexity properties.

3.1 Pearl's algorithm

Pearl's basic algorithm for probabilistic inference takes the digraph of a belief network for its computational architecture. The vertices in the graph are viewed as *autonomous objects* and the arcs as *bi-directional communication channels*. Each vertex has a local processor that is capable of performing simple, pre-defined computations and a local memory in which its associated probabilities are stored. Through the communication channels the vertices send each other *messages* providing information about the represented probability distribution. Each vertex is able to compute the probabilities of its values from the probabilities stored in its memory and the information it receives from its neighbours [Pea88]. When a vertex' true value becomes known, the messages this vertex sends to its neighbours are updated to reflect the evidence, forcing its neighbours to compute updated messages in turn. The impact of evidence thus spreads throughout the graph by message-passing between neighbours.

Message-passing between neighbouring vertices suffices for correct probabilistic inference with a belief network comprising a *singly connected digraph*, that is, a digraph whose underlying, undirected graph is acyclic. For such a network, the computational complexity of Pearl's algorithm can be shown to be $O(n \cdot d \cdot 2^d)$, where n is the number of vertices in the network's digraph and d is the digraph's (maximal) in-degree. Note that the complexity of the algorithm is exponential in the number of vertices if the digraph's in-degree

is $\Omega(n)$. If, however, the in-degree of a belief network's digraph is bounded by a constant, the algorithm takes linear time.

For belief networks comprising a *multiply connected digraph*, that is, a digraph having one or more cycles in its underlying graph, message-passing between neighbouring vertices no longer suffices for correct probabilistic inference, as vertices may for example indefinitely send updated messages to their neighbours. For these networks, message-passing is supplemented with a method called *loop-cutset conditioning* [Pea88, SC90]. The basic idea of this method is to *cut* all cyclic chains, or *loops*, in the digraph so as to let it behave as if it were singly connected.

Definition 3.1 *Let $G = (V, A)$ be an acyclic digraph. A set $L \subseteq V$ is a* loop cutset *for G if for each loop c in G there is a vertex $v_i \in L$ with an outgoing arc on c.*

In general, a multiply connected digraph allows several different loop cutsets.

Example 3.2 Consider the digraph G shown in Figure 1. The set of vertices $\{v_1, v_7\}$ is an example loop cutset for G. □

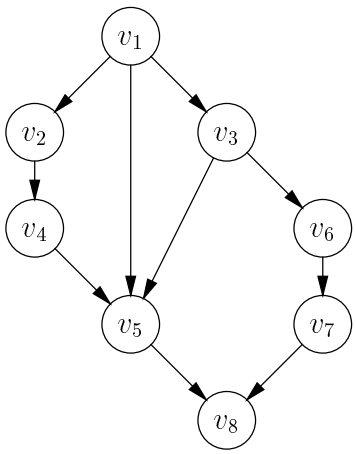

Figure 1: An example digraph G.

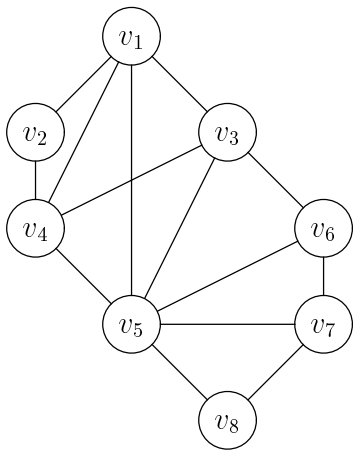

Figure 2: An example triangulated moral graph H for G.

For a belief network comprising a multiply connected digraph, the computational complexity of Pearl's algorithm equals $O(n \cdot d \cdot 2^{d+l})$ where n is the number of vertices in the network's digraph, d is the digraph's in-degree, and l is the number of vertices in the loop cutset that is selected for the digraph. Note that the algorithm's complexity is exponential in the size of the loop cutset used.

Since the computational complexity of Pearl's algorithm relates exponentially to loop-cutset size, the best loop cutset to use in practical applications is a loop cutset with a minimal number of vertices. The problem of finding for a multiply connected digraph a smallest loop cutset is known to be NP-hard [SC90]. An efficient heuristic algorithm exists, however, that finds a loop cutset for a multiply connected digraph with at most twice the minimal number of vertices needed to cut all loops in the graph [BG96].

3.2 Lauritzen and Spiegelhalter's algorithm

In *Lauritzen and Spiegelhalter's algorithm* for probabilistic inference, the digraph of a belief network is not exploited directly as a computational architecture as it is in Pearl's algorithm. Instead, the digraph is first transformed into an undirected *triangulated* graph, that is, an undirected graph in which no cycle of length four or more exists without a shortcut [Gol80].

Definition 3.3 *Let $G = (V, A)$ be an acyclic digraph. A* triangulated moral graph *for G is an undirected triangulated graph $H = (V, E)$ such that*

- *if $v_i \to v_j \in A$, then $v_i - v_j \in E$;*
- *if $v_i \to v_j, v_k \to v_j \in A$, then $v_i - v_k \in E$.*

In general, an acyclic digraph allows several different triangulated moral graphs.

Example 3.4 Consider once more the digraph G from Figure 1. The undirected graph H shown in Figure 2 is a triangulated moral graph for G. Note that for any vertex, its original set of predecessors is included in the same (maximal) clique of H as the vertex itself. □

From a belief network's triangulated moral graph, a *clique tree*, or *junction tree*, is constructed [Jen96]. A clique tree is a tree in which the vertices represent the (maximal) cliques of the graph it is constructed from; the intersections between the cliques give rise to the tree's edges.

Definition 3.5 *Let $G = (V, A)$ be an acyclic digraph. Let H be a triangulated moral graph for G; let $C = \{C_i \mid i = 1, \ldots, n\}$ be the set of cliques of H and, for each clique $C_i \in C$, let V_i be its vertex set. A* clique tree *for G is a tree $T = (C, E)$ in which the set of edges E satisfies the following property: for any two cliques $C_i, C_j \in C$ and each clique C_k on the (unique) path from C_i to C_j in T, we have that $V_i \cap V_j \subseteq V_k$.*

In general, a triangulated moral graph allows various different clique trees. For constructing a clique tree from a triangulated graph efficient algorithms are available [Pea88, Jen96]; details of these algorithms are not relevant for the present paper.

Example 3.6 Consider once more the digraph G from Figure 1. Figure 3 shows a clique tree for G that is constructed from the triangulated moral graph H from Figure 2. □

Lauritzen and Spiegelhalter's algorithm now takes a clique tree constructed from a belief network's digraph for its computational architecture: the cliques in the tree are viewed as autonomous objects and the tree's edges are looked upon as bi-directional communication channels. Each clique has a local processor that is capable of performing simple, predefined probabilistic computations and a local memory in which the marginal distribution

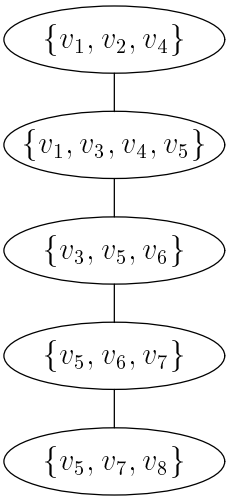

Figure 3: An example clique tree for the graph H.

on its vertices is stored. Through the communication channels the cliques send each other messages providing information about the represented joint probability distribution. Each clique is able to compute the (updated) marginal distribution on its vertices from the marginal distribution stored in its local memory and the information it receives from its neighbours.

The computational complexity of Lauritzen and Spiegelhalter's algorithm equals $O(n \cdot 2^c)$ where n is the number of vertices in a belief network's digraph and c is the number of vertices in the largest clique in the clique tree that is constructed from the digraph. Note that the algorithm's complexity is exponential in the size of the largest clique. If, however, in the clique tree used, the clique sizes are bounded by a constant, the algorithm takes linear time.

Since the computational complexity of Lauritzen and Spiegelhalter's algorithm relates exponentially to clique size, the best clique tree to use in practical applications is a tree comprising cliques with the smallest maximal number of vertices (or, more precisely, a clique tree inducing smallest state space). The problem of finding for a digraph such a clique tree is known to be NP-hard [Wen90]; we would like to note that the problem is closely related to the well-studied problem of determining treewidth of a graph [Bod93]. For finding a clique tree for an acyclic digraph, various efficient heuristic algorithms are available [RTL76, TY84, Kjæ91]. These algorithms, however, do not exhibit any optimality properties.

4 Comparing loop cutsets and clique trees

In the previous section, we have illustrated that the two best-known algorithms for probabilistic inference with a belief network build on different graphical structures: the algorithm of J. Pearl builds on a *loop cutset* for a network's digraph whereas the algorithm of S.L.

Lauritzen and D.J. Spiegelhalter builds on a *clique tree* for the graph. In this section, we compare the two algorithms as to their computational complexity by comparing loop cutsets and clique trees. We show that the algorithm of Lauritzen and Spiegelhalter has at most the same complexity as Pearl's algorithm.

4.1 Relating clique trees to loop cutsets

We begin our analysis of the two algorithms for probabilistic inference with a belief network by relating clique trees to loop cutsets. Given a loop cutset for a network's digraph, we show that a clique tree for this graph exists in which the cliques differ in size from the loop cutset by at most a linear term.

Proposition 4.1 *Let $G = (V, A)$ be an acyclic digraph and let L be a loop cutset for G. Then, there exists a clique tree $T = (C, E)$ for G such that for each clique $C_i \in C$ with vertex set V_i, we have that*

$$|V_i| \leq 1 + \textit{in-degree}(G) + |L|$$

where in-degree(G) denotes the (maximal) in-degree of G.

Proof. The property stated in the proposition is proved by demonstrating the construction of a clique tree having the desired property from an acyclic digraph and associated loop cutset; the presented construction is derived to a large extent from the clique-tree construction for global conditioning [SAS94].

We consider the digraph G and its loop cutset L. From G, we construct the digraph $G' = (V, A')$ by deleting the outgoing arcs of all vertices from L. From L being a loop cutset for G, we have that the digraph G' is singly connected. From G', we now construct the undirected graph $H' = (V, E')$ by first replacing all directed arcs from A' by undirected edges and by subsequently adding an edge between any two, yet unconnected, vertices v_i, v_k for which $v_i \rightarrow v_j, v_k \rightarrow v_j \in A'$ for some vertex v_j. From G' being singly connected, we have that the graph H' is triangulated. It further is readily verified that H' is a triangulated moral graph for G'. We now construct from H' the graph $H = (V, E)$ by adding to each clique all vertices from the set L. From H' being triangulated and from the construction of H, it follows that H is triangulated as well. In fact, H is a triangulated moral graph for G. From the graph H, we construct a clique tree for G.

From the construction of the graph H', we have that its maximal clique size equals $1 + \textit{in-degree}(G')$. From the construction of the graph H from H', we conclude that H's maximal clique size equals $1+\textit{in-degree}(G')+|L|$. From the observation that $\textit{in-degree}(G') \leq \textit{in-degree}(G)$, we find the property stated in the proposition. □

Note that for an acyclic digraph G the term *in-degree*(G) in the expression stated in the previous proposition may be linear in the number of vertices in G.

Example 4.2 Consider once more the construction of a clique tree for an acyclic digraph as outlined in the proof of the previous proposition. We illustrate (part of) the construction

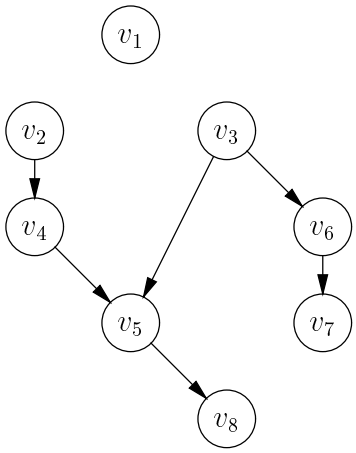

Figure 4: The graph G' for the example digraph G.

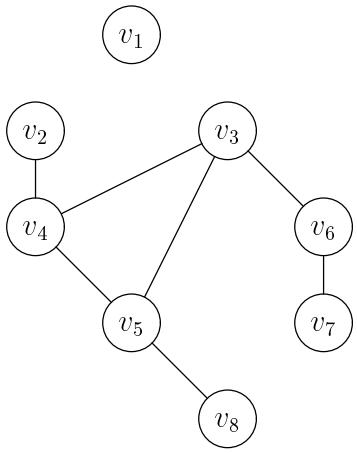

Figure 5: The triangulated moral graph H' for the graph G'.

for the digraph G from Figure 1 and its loop cutset $L = \{v_1, v_7\}$. From G, the digraph G' shown in Figure 4 is constructed. From G', we construct the undirected graph H' shown in Figure 5. After adding to each clique from H' all vertices from the set L, the clique tree T shown in Figure 6 may be yielded. □

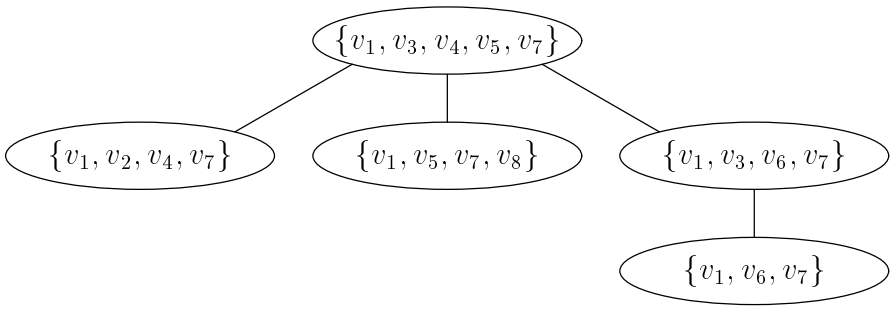

Figure 6: The clique tree T resulting from the graph H'.

We re-examine the complexity properties of the two algorithms for probabilistic inference in view of the property stated in Proposition 4.1. For an acyclic digraph with an in-degree that is bounded by a constant, we have that a clique tree can be found in which the largest clique is of the same order of magnitude as a given loop cutset for the graph. For a belief network comprising a digraph with bounded in-degree, Pearl's algorithm takes $O(n \cdot 2^l)$ time, where n once more denotes the number of vertices in the network's digraph and l is the number of vertices in the loop cutset used. From the property mentioned above we have that for this network a clique tree exists using which Lauritzen and Spiegelhalter's algorithm takes $O(n \cdot 2^l)$ time as well. For a belief network with bounded in-degree, therefore, Lauritzen and Spiegelhalter's algorithm has at most the same computational complexity as Pearl's algorithm. More specifically, if the latter algorithm takes linear time for a belief network, then the former can show linear behaviour for the network as well.

We would like to note that, for a belief network comprising an acyclic digraph with an in-degree of $\Omega(n)$, where n is the number of vertices in the graph, both Pearl's algorithm and the algorithm of Lauritzen and Spiegelhalter will show exponential behaviour, regardless of the loop cutset and clique tree used, respectively. In practical applications, however, very rarely a belief network with a digraph of in-degree $\Omega(n)$ is found.

The conclusion with regard to the algorithms' computational complexity mentioned above, may seem to contradict earlier results reported in the literature [SC91, Nea90]. These results pertain to the digraph G shown in Figure 7. Upon constructing a triangulated moral graph for G, a clique tree may be yielded that comprises a clique with as many as $\Omega(\sqrt{n})$ vertices, dependent on the heuristic algorithm used for this purpose [SC91, Nea90]. Building on this particular clique tree for its computational architecture, the algorithm of Lauritzen and Spiegelhalter takes exponential time. Now, an example loop cutset for G is the set $L = \{v_1\}$, having size one. Since, in addition, the in-degree of the digraph equals two, Pearl's algorithm will behave linearly for G. The apparent difference in complexity of the two algorithms can be attributed to the clique tree used with Lauritzen and Spiegelhalter's algorithm being an unfortunate one: the heuristic algorithm used for constructing the clique tree has yielded a tree that is far from optimal. From Proposition 4.1 we have that for the digraph G a clique tree exists that comprises cliques with at most four vertices.

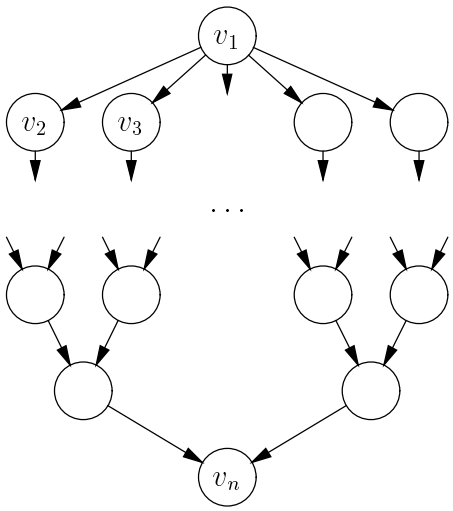

Figure 7: An acyclic digraph with large parallel loops.

4.2 Relating Loop Cutsets to Clique Trees

In the previous section, we have argued that, if for a belief network Pearl's algorithm for probabilistic inference has a linear computational complexity, then Lauritzen and Spiegelhalter's algorithm can show linear behaviour for this network as well. The reverse property does not hold. There exist belief networks for which Lauritzen and Spiegelhalter's algorithm has a linear computational complexity and Pearl's algorithm takes exponential time

at best. That is, there exist belief networks comprising a digraph for which a clique tree can be constructed in which the maximal clique size is bounded by a constant, yet for which no loop cutset of constant size exists. We give an example of such a digraph, taken from [SC91].

Example 4.3 Consider the digraph G shown in Figure 8. For this digraph, a clique tree exists in which all cliques include three vertices. The smallest loop cutset for the digraph, however, comprises $\frac{n-1}{3}$ vertices. □

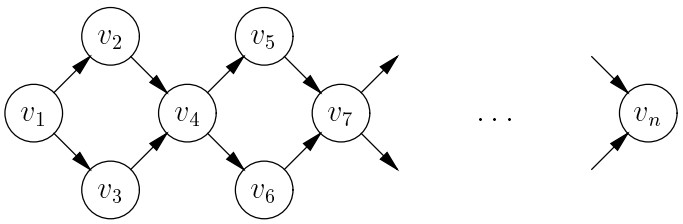

Figure 8: An acyclic digraph with small loops in series.

5 Conclusions

We have examined the computational complexity of the two best-known algorithms for probabilistic inference with a Bayesian belief network. By comparing the graphical structures underlying the computational architectures employed by these algorithms, we have shown that Lauritzen and Spiegelhalter's algorithm has at most the same complexity as Pearl's algorithm. More specifically, if Pearl's algorithm has a linear complexity for a belief network, then Lauritzen and Spiegelhalter's algorithm can behave linearly for this network as well. If, however, Lauritzen and Spiegelhalter's algorithm has a linear complexity for a network, then Pearl's algorithm may show exponential behaviour at best. In general, therefore, Lauritzen and Spiegelhalter's algorithm will outperform Pearl's algorithm.

For probabilistic inference with a belief network using Lauritzen and Spiegelhalter's algorithm, a clique tree is constructed from the network's digraph. Unfortunately, no polynomial-time algorithms are known for constructing a clique tree with smallest clique size. For practical applications, therefore, generally a heuristic algorithm is used. Several such heuristic algorithms are available, neither of which exhibit any optimality properties. Our main proposition, relating clique trees to loop cutsets, provides for yet another heuristic algorithm for constructing a clique tree for a belief network's digraph. Although this algorithm also does not exhibit any optimality properties, it guarantees for a digraph with bounded in-degree that, if the digraph has a small loop cutset, then a clique tree with small clique size is found.

References

[BG96] A. Becker and D. Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, vol. 83, 1996, pp. 167 – 188.

[Bod93] H.L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, vol. 11, 1993, pp. 1 – 23.

[Coo90] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, vol. 42, 1990, pp. 393 – 405.

[Gol80] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[Jen96] F.V. Jensen. *An Introduction to Bayesian Networks*, UCL Press, London, 1996.

[Kjæ91] U. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, vol. 2, 1991, pp. 7 – 17.

[LS88] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, vol. 50, 1988, pp. 157 – 224.

[Nea90] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems. Theory and Algorithms*, John Wiley & Sons, New York, 1990.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference*, Morgan Kaufmann, Palo Alto, 1988.

[RTL76] D.J. Rose, R.E. Tarjan, and G.S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, vol. 5, 1976, pp. 266 – 283.

[SAS94] R.D. Shachter, S.K. Andersen, and P. Szolovits. Global conditioning for probabilistic inference in belief networks. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 1994, pp. 514 – 522.

[SC90] H.J. Suermondt and G.F. Cooper. Probabilistic inference in multiply connected belief networks using loop cutsets. *International Journal of Approximate Reasoning*, vol. 4, 1990, pp. 283 – 306.

[SC91] H.J. Suermondt and G.F. Cooper. A combination of exact algorithms for inference on Bayesian belief networks. *International Journal of Approximate Reasoning*, vol. 5, 1991, pp. 521 – 542.

[TY84] R.E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, vol. 13, 1984, pp. 566 – 579.

[Wen90] W.X. Wen. Optimal decomposition of belief networks. *Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence*, Cambridge, Massachusetts, 1990, pp. 245 – 256.