PREPROCESSING RULES FOR TRIANGULATION OF PROBABILISTIC NETWORKS*

HANS L. BODLAENDER

Institute of Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

ARIE M.C.A. KOSTER

Zuse Institute Berlin, Takustrasse 7, D-14195 Berlin-Dahlem, Germany

FRANK VAN DEN EIJKHOF

Department of Methodology and Statistics, Faculty of Social Sciences, Utrecht University,
P.O. Box 80.0140, 3508 TC Utrecht, The Netherlands

Currently, the most efficient algorithm for inference with a probabilistic network builds upon a triangulation of a network's graph. In this paper, we show that preprocessing can help in finding good triangulations for probabilistic networks, that is, triangulations with a maximum clique size as small as possible. We provide a set of rules for stepwise reducing a graph, without losing optimality. This reduction allows us to solve the triangulation problem on a smaller graph. From the smaller graph's triangulation, a triangulation of the original graph is obtained by reversing the reduction steps. Our experimental results show that the graphs of some well-known real-life probabilistic networks can be triangulated optimally just by preprocessing; for other networks, huge reductions in their graph’s size are obtained.

Key words: probabilistic networks, inference, triangulation, treewidth, preprocessing.

1. INTRODUCTION

To compute the inference with a probabilistic network, currently the most efficient algorithm is the junction-tree propagation algorithm that builds upon a triangulation of a network’s moralized graph (Lauritzen and Spiegelhalter 1998, Jensen, Lauritzen, and Olesen 1990). The running time of this algorithm depends on the specific triangulation used. In general, it is hard to find a triangulation for which this running time is minimal. Because there is a strong relationship between the running time of the algorithm and the maximum of the triangulation’s clique sizes, for real-life networks triangulations are sought for which this maximum is minimal. The minimum of the maximum clique size over all triangulations of a graph is a well-studied notion, both by researchers in the field of probabilistic networks (Amir 2001; Gogate and Dechter 2004; Kjærulff 1993) and by researchers in graph theory and graph algorithms (Bodlaender 1993, 1998). In the latter field of research, the notion of treewidth is used to denote this minimum minus one. Unfortunately, computing the treewidth of a given graph is an NP-complete problem (Arnborg, Corneil, and Proskurowski 1987).

When solving hard combinatorial problems, preprocessing is often profitable. The basic idea is to reduce the size of a problem under study, using relatively little computation time and without losing optimality. The smaller, and presumably easier, problem is subsequently solved. In this paper, we discuss preprocessing for triangulation of probabilistic networks. We provide a set of rules for stepwise reducing the problem of finding a triangulation for a network’s moralized graph with minimal maximum clique size to the same problem on...
a smaller graph. Various algorithms can then be used to solve the smaller problem. Given a triangulation of the smaller graph, a triangulation of the original graph is obtained by reversing the reduction steps. Our reduction rules are guaranteed not to destroy optimality with respect to maximum clique size. Experiments with preprocessing revealed that our rules can effectively reduce the problem size for various real-life probabilistic networks. In fact, the graphs of some well-known networks are triangulated optimally just by preprocessing.

In this paper, we do not address the actual construction of a triangulation for the graph remaining after preprocessing. Recent research results indicate, however, that for small graphs optimal triangulations can be feasibly computed. Building upon a variant of an algorithm by Arnborg, Corneil, and Proskurowski (Arnborg et al. 1987), Shoikhet and Geiger performed various experiments on randomly generated graphs (Shoikhet and Geiger 1997). Their results indicate that this algorithm (exponential in the treewidth) allows for computing optimal triangulations of graphs with up to 100 vertices. Gogate and Dechter (2004) obtained similar results with a branch and bound algorithm.

Most of the preprocessing rules introduced in this paper are based on the rules that characterize graphs of treewidth three (Arnborg and Proskurowski 1986). We extend these rules in the following ways: we introduce the almost simplicial vertex rule, which generalizes the series and triangle rule; we extend the cube rule; and we show that the rules are also safe for graphs which have treewidth larger than three when using an additional variable that maintains a lower bound on the treewidth.

The paper is organized as follows. In Section 2, we review some basic definitions. In Section 3, we present our preprocessing rules. The computational model in which these rules are employed, is discussed in Section 4. In Section 5, we report on our experiments with well-known real-life probabilistic networks. The paper ends with some conclusions and directions for further research in Section 6.

2. DEFINITIONS

Currently the most efficient algorithm for probabilistic inference operates on a junction tree that is derived from a triangulation of the moralization of the digraph of a probabilistic network. We review the basic definitions involved.

Let \( D = (V, A) \) be a directed acyclic graph with node set \( V \) and arc set \( A \). The indegree of a node \( v \) is the number of arcs directed to \( v \), whereas the outdegree is the number of arcs directed from \( v \). The moralization of \( D \) is the undirected graph \( M(D) \) obtained from \( D \) by adding edges between every pair of nonadjacent vertices that have a common successor (vertices \( v \) and \( w \) have a common successor if there is a vertex \( x \) with \( (v, x) \in A \) and \( (w, x) \in A \), and then dropping the directions of all edges.

Let \( G = (V, E) \) be an undirected graph with vertex set \( V \) and edge set \( E \). The degree of a vertex \( v \) is the number of edges incident to it. A set of vertices \( W \subseteq V \) is called a clique in \( G \) if there is an edge between every pair of disjoint vertices from \( W \); the cardinality of \( W \) is the clique’s size. A clique \( W \) is maximal if there is no \( v \in V - W \) such that \( W \cup \{v\} \) is also a clique. A clique in \( G \) is maximum if \( G \) contains no clique of larger size. For a set of vertices \( W \subseteq V \), the subgraph induced by \( W \) is the graph \( G[W] = (W, (W \times W) \cap E) \); for a single vertex \( v \), we write \( G - v \) to denote \( G[V - \{v\}] \). A cycle in \( G \) is an ordered set of vertices \( \{v_1, \ldots, v_k\} \) with \( v_i v_{i+1} \in E \), \( i = 1, \ldots, k-1 \) and \( v_k v_1 \in E \). A cycle \( C \) is called simple if \( G[C] \) only contains edges on the cycle. The graph \( G \) is triangulated if it does not contain an induced subgraph that is a simple cycle of length at least four. A triangulation of \( G \) is a triangulated graph \( H(G) \) that contains \( G \) as a subgraph. The treewidth of the triangulation \( H(G) \) of \( G \) is the maximum clique size in \( H(G) \) minus 1. The treewidth of \( G \), denoted \( \tau(G) \), is the minimum treewidth over all triangulations of \( G \).
A graph $H$ is a minor of a graph $G$ if $H$ can be obtained from $G$ by zero or more vertex deletions, edge deletions, and edge contractions (edge contraction is the operation that replaces two adjacent vertices $v$ and $w$ by a single vertex that is connected to all neighbors of $v$ and $w$). It is easy to verify (see, e.g., Bodlaender 1998), that the treewidth of a minor of $G$ is never larger than the treewidth of $G$ itself.

A linear ordering of an undirected graph $G = (V, E)$ is a bijection $V \leftrightarrow \{1, \ldots, |V|\}$. For $v \in V$ and a linear ordering $f$ of $G - v$, we denote by $(v; f)$ the linear ordering $f'$ of $G$ that is obtained by adding $v$ at the beginning of $f$, that is, $f'(v) = 1$ and, for all $w \neq v$, $f'(w) = f(w) + 1$. For a linear order $f$ and vertices $v$, $w$, we use for $(v; (w; f))$ the shorthand notation $(v; w; f)$. A linear ordering $f$ is a perfect elimination scheme for $G$ if, for each $v \in V$, its higher ordered neighbors form a clique, that is, if every pair of distinct vertices in the set $\{w \in V \mid \{v, w\} \in E \text{ and } f(v) < f(w)\}$ is adjacent. A graph is triangulated if and only if it allows a perfect elimination scheme (see, e.g., Golumbic 1980).

For a graph $G = (V, E)$ and a linear ordering $f$ of $G$, there are one or more triangulations of $G$ that have $f$ for its perfect elimination scheme. A trivial example of such a triangulation is the complete graph with vertex set $V$. Of these triangulations of $G$ that have $f$ as perfect elimination scheme, there is a unique one that is minimal in the sense that it does not contain another such triangulation as proper subgraph. This triangulation, which we term the fill-in graph given $f$, can be constructed by, for $i = 1, \ldots, |V|$, turning the set of higher numbered neighbors of $f^{-1}(i)$ into a clique. The maximum clique size minus 1 of this fill-in is called the treewidth of $f$. The treewidth of a linear ordering of a triangulated graph equals the maximum number of higher numbered neighbors of a vertex (Golumbic 1980).

To conclude, a junction tree of an undirected graph $G = (V, E)$ is a tree $T = (I, F)$, where every node $i \in I$ has associated a vertex set $V_i$, such that the following two properties hold: the set $\{V_i \mid i \in I\}$ equals the set of maximal cliques in $G$ and, for each vertex $v$, the set $T_v = \{i \mid v \in V_i\}$ constitutes a connected subtree of $T$. It is well known (see, e.g., Golumbic 1980), that a graph is triangulated if and only if it has a junction tree.

### 3. SAFE REDUCTION RULES

#### 3.1. Framework

In this paper, we consider reduction rules that work on a pair, consisting of a graph, and an integer variable, called low. This variable will be used as a lower bound for the treewidth of the original undirected graph. Some rules appear to be only safe if the variable low is large enough. Formally, our reduction rules are binary relations between two pairs, each consisting of an undirected graph and an integer. We use the notation: $(G, low) \rightarrow_R (G', low')$. We call a rule $\rightarrow_R$ safe, if for all graphs $G$, $G'$, integers $low$, $low'$, we have

$$(G, low) \rightarrow_R (G', low') \Rightarrow \max(\tau(G), low) = \max(\tau(G'), low').$$

A set of rules $\mathcal{R}$ is safe, if each rule $\rightarrow_R \in \mathcal{R}$ is safe. The following straightforward lemma shows why we want to use safe rules.

**Lemma 1.** Let $\mathcal{R}$ be a safe set of reduction rules. Suppose $low \leq \tau(G)$, and suppose $(G', low')$ can be obtained from $(G, low)$ by zero or more successive applications of rules in $\mathcal{R}$. Then $\tau(G) = \max(\tau(G'), low')$.  

Proof. Because all rules that are applied are safe, we have that \( \max(\tau(G), \text{low}) = \max(\tau(G'), \text{low}') \). (This can be shown with induction to the number of reduction rules that are applied.) Using \( \text{low} \leq \tau(G) \), the result follows. 

The algorithmic technique used in this paper for preprocessing graphs for triangulation builds upon a set of reduction rules. These rules allow for stepwise reducing a graph to another graph with fewer vertices. The steps applied during the reduction can be reversed, thereby enabling us to compute a triangulation of the original graph from a triangulation of the smaller graph.

In this section, we discuss the various rules; a discussion of the computational method in which these rules are employed, is deferred to Section 4.

During a graph’s reduction, we maintain a stack of eliminated vertices. We maintain the (reduced) graph, and the integer variable \( \text{low} \) as discussed above, \( \text{low} \) gives a lower bound for the treewidth of the original graph at any step of the algorithm.

Application of a reduction rule serves to modify the current graph \( G \) to \( G' \) and to possibly update \( \text{low} \) to \( \text{low}' \). By applying safe rules, we have as an invariant that the treewidth of the original graph equals the maximum of the treewidth of the reduced graph and the value \( \text{low} \).

In the sequel, we assume that the original moralized graph \( G \) has at least one edge and that \( \text{low} \) is initialized at a number, at least 1 and at most \( \tau(G) \). (For instance, one can start with a heuristic that computes a lower bound for \( \tau(G) \), and sets \( \text{low} \) to the value computed by the heuristic. Because \( G \) has at least one edge, \( \tau(G) \geq 1 \).)

3.2. A Collection of Safe Reduction Rules

In this subsection, we give several safe reduction rules. Our first reduction rule applies to simplicial vertices. A vertex \( v \) is simplicial in an undirected graph \( G \) if the neighbors of \( v \) form a clique in \( G \). The following proposition shows that when \( v \) is a simplicial vertex in a graph \( G \), computing the treewidth of \( G \) is equivalent to computing the treewidth of \( G - v \).

**Proposition 1.** Let \( G \) be an undirected graph and let \( v \) be a simplicial vertex in \( G \) with degree \( d \geq 0 \). Then,

- \( \tau(G) = \max(d, \tau(G - v)) \);
- For every linear ordering \( f \) of \( G - v \) with treewidth at most \( \max(d, \tau(G - v)) \), \((v; f)\) is a linear ordering of \( G \) with minimum treewidth.

Proof. Because \( G \) contains a clique of size \( d + 1 \), we have that \( \tau(G) \geq d \). We further observe that \( \tau(G) \geq \tau(G - v) \), because \( G - v \) is a minor of \( G \). We therefore have that \( \tau(G) \geq \max(d, \tau(G - v)) \). Now, let \( f \) be a linear ordering of \( G - v \) of treewidth \( k \leq \max(d, \tau(G - v)) \). Let \( H \) be the fill-in of \( G - v \) given \( f \). Adding vertex \( v \) and its (formerly) incident edges to \( H \) yields a graph \( H' \) that is still triangulated: as every pair of neighbors of \( v \) is adjacent, \( v \) cannot belong to a simple (chordless) cycle of length at least four. The maximum clique size of \( H' \), therefore, equals the maximum of \( d + 1 \) and \( k + 1 \). Hence, \( \tau(G) \leq \max(d, \tau(G - v)) \), from which we conclude the first property stated in the proposition. To prove the second property, we observe that the linear ordering \((v; f)\) is a perfect elimination scheme for \( H' \), as removal of \( v \) upon computing the fill-in of \( H' \) does not create any additional edges. 

Our first reduction rule, illustrated in Figure 1, now is:

**Reduction Rule 1: Simplicial vertex rule**

Let \( v \) be a simplicial vertex of degree \( d \geq 0 \).

Remove \( v \).

Set \( \text{low} \) to max \((\text{low}, d)\).
The second property stated in Proposition 1 provides for the rule’s reversal when computing a triangulation of the original graph from a triangulation of the reduced one. The first property can be used to show that the rule is safe.

Lemma 2. The simplicial vertex rule is safe.

Proof. Suppose a reduction \((G, \text{low}) \rightarrow_r (G - v, \text{low}')\) is done by the simplicial vertex rule, with \(v\) a simplicial vertex of degree \(d\). Now \(\text{max}(\tau(G), \text{low}) = \text{max}(d, \tau(G - v), \text{low}) = \text{max}(\tau(G - v), \text{low}')\).

Because the digraph \(D\) of a probabilistic network is moralized before it is triangulated, it is likely to give rise to many simplicial vertices. We consider a vertex \(v\) with outdegree zero in \(D\). Because all neighbors of \(v\) have an arc pointing into \(v\), moralization will connect every two neighbors that are not yet adjacent, thereby effectively turning \(v\) into a simplicial vertex. The simplicial vertex rule will thus remove at least all vertices that have outdegree zero in the network’s original digraph. Because every directed acyclic graph has at least one vertex of outdegree zero, at least one reduction will be performed. Because the reduced graph need not be the moralization of a directed acyclic graph, it is possible that no further reductions can be applied.

The digraph \(D\) of a probabilistic network may also include vertices with indegree zero and outdegree one. These vertices will always be simplicial in the moralization of \(D\). We consider a vertex \(v\) with indegree zero and a single arc pointing into a vertex \(w\). In the moralization of \(D\), \(w\) and its (former) predecessors constitute a clique. Because all neighbors of \(v\) belong to this clique, \(v\) is simplicial.

A special case of the simplicial vertex rule now applies to vertices of degree 1; it is termed the twig rule, after Arnborg and Proskurowski (1986).

**Reduction Rule 1a: Twig rule**

Let \(v\) be a vertex of degree 1. Remove \(v\).

The twig rule is based upon the observation that vertices of degree one are always simplicial. Another special case is the islet rule that serves to remove vertices of degree zero. Because we assumed that we started with \(\text{low} \geq 1\), there is no need to update \(\text{low}\) in the twig or islet rule.

**Reduction Rule 1b: Islet rule**

Let \(v\) be a vertex of degree 0. Remove \(v\).
Our second reduction rule applies to almost simplicial vertices. A vertex \( v \) is *almost simplicial* in an undirected graph \( G \) if there is a neighbor \( w \) of \( v \) such that all other neighbors of \( v \) form a clique in \( G \). Figure 2 illustrates the basic idea of an almost simplicial vertex. Note that we allow other neighbors of \( v \) to be adjacent to \( w \). Simplicial vertices therefore are also almost simplicial.

**Proposition 2.** Let \( G \) be an undirected graph and let \( v \) be an almost simplicial vertex in \( G \) with degree \( d \geq 0 \). Let \( G' \) be the graph that is obtained from \( G \) by turning the neighbors of \( v \) into a clique and then removing \( v \). Then,

\[
\tau(G') \leq \tau(G) \quad \text{and} \quad \tau(G) \leq \max(d, \tau(G'));
\]

- the linear ordering \( (v; f) \) of \( G \), with \( f \) a linear ordering of \( G' \) of treewidth at most \( \max(d, \tau(G')) \), has treewidth at most \( \max(d, \tau(G')) \).

**Proof.** Let \( w \) be a neighbor of \( v \) such that the other neighbors of \( v \) form a clique. Because we can obtain \( G' \) from \( G \) by contracting the edge \( \{v, w\} \), \( G' \) is a minor of \( G \). We therefore have that \( \tau(G') \leq \tau(G) \). Now, let \( f \) be a linear ordering of \( G' \) of treewidth \( k \leq \max(d, \tau(G')) \). Let \( H \) be the fill-in of \( G' \) given \( f \). If we add \( v \) and its (formerly) adjacent edges to \( H \), then \( v \) is simplicial in the resulting graph \( H' \). Using Proposition 1, we find that \( \tau(H') = \max(k, d) \).

The two properties stated in the proposition follow. \( \blacksquare \)

Examples can be constructed, unfortunately, that show that the rule is not safe for \( \text{low} < d \).

Let \( G \) be the graph in the left-hand side of Figure 4, and suppose we have \( \text{low} = 2 \). \( G \) has treewidth three (because it contains a clique with four vertices as a minor), while if we would
apply the *almost simplicial vertex rule* to it, we would obtain the graph in the right-hand side of Figure 4, whose treewidth is two. While in the example it is possible to apply other rules first (i.e., the series rule), it is also possible to construct more complicated examples with the same property to which no other reductions can be applied.

In practice, the almost simplicial vertex rule can be expected to be applied regularly in moralizations of probabilistic networks as Figure 5 illustrates. If in the digraph $D$ of a probabilistic network there is an arc $(v, w)$, with the outdegree of $v$ exactly one, and the indegree of $w$ exactly one, then $v$ will be almost simplicial in the moralization $M(D)$: note that, because $v$ has one child in $D$ which has only $v$ as parent, no moralization edges will be added with $v$ as endpoint. Thus, the neighbors of $v$ in $M(D)$ are its parents in $D$ and $w$; due to the moralization, the parents of $v$ form a clique in $M(D)$.

A special case of the *almost simplicial vertex rule* applies to vertices of degree two. A vertex of degree two is, by definition, almost simplicial and we can, therefore, replace it by an edge between its neighbors, provided that the original graph has treewidth at least two. The resulting rule, illustrated in Figure 6, is called the *series rule*, after Arnborg and Proskurowski (1986).

### Reduction Rule 2a: Series rule
Let $v$ be a vertex of degree 2.
If $\text{low} \geq 2$, then
add an edge between the neighbors of $v$, if they are not already adjacent;
remove $v$.

Another special case is the *triangle rule*, shown in Figure 7.

### Reduction Rule 2b: Triangle rule
Let $v$ be a vertex of degree 3 such that at least two of its neighbors are adjacent.
If $\text{low} \geq 3$, then
add an edge between every pair of nonadjacent neighbors of $v$;
remove $v$. 

Because the *series* and *triangle rules* are special cases of the *almost simplicial vertex rule*, both are safe.

Using the fact that a nonempty graph of treewidth at most $k$ has a vertex with degree at most $k$, the following well known observations easily follow. If the *twig* and *islet rules* cannot be applied to a nonempty undirected graph, then all its vertices have degree at least two, and hence the graph has treewidth at least two. We can then set $low$ to $\max(low, 2)$. Note that from this observation we have that the *islet* and *twig rules* suffice for reducing any graph of treewidth one to the empty graph. The *islet*, *twig*, and *series rules* suffice for reducing any graph of treewidth two to the empty graph. (A nonempty graph of treewidth at most two has a vertex of degree at most two, to which one of these rules can be applied; possibly setting $low$ to $\max(low, 2)$ when no *islet* or *twig* rule can be applied.) Thus, if $low \geq 2$ for a given nonempty graph and the *islet*, *twig*, and *series rules* cannot be applied, then we know that the graph has treewidth at least three. We can then set $low$ to $\max(low, 3)$. 

---

**Figure 5.** A case where an almost simplicial vertex is created when moralizing.

**Figure 6.** The *series rule*.

**Figure 7.** The *triangle rule*. 
Similar to treewidths one and two, there is a set of rules that suffice for reducing any graph of treewidth three to the empty graph. This set of rules was first identified by Arnborg and Proskurowski (1986). The islet, twig, series, and triangle rules are among the set of six. The two other rules are of interest to us, not just because they provide for computing optimal triangulations for graphs of treewidth three, but also because they give new reduction rules for the purpose of preprocessing.

**Proposition 3.** Let $G$ be an undirected graph and let $v$, $w$ be two vertices of degree three having the same set of neighbors. Let $G'$ be the graph that is obtained from $G$ by turning the set of neighbors of $v$ into a clique and then removing $v$ and $w$. Then,

- $\tau(G') \leq \tau(G)$ and $\tau(G) \leq \max(3, \tau(G'))$;
- the linear ordering $(v;w;f)$, with $f$ a linear ordering of $G'$ of treewidth at most $\max(3, \tau(G'))$, has treewidth at most $\max(3, \tau(G'))$.

**Proof.** Suppose that $x$, $y$ and $z$ are the neighbors of $v$ and $w$. By contracting the edges $\{v, x\}$ and $\{w, y\}$ in $G$, we obtain $G'$. Thus, $G'$ is a minor of $G$ and we find that $\tau(G') \leq \tau(G)$. Now, let $f$ be a linear ordering of $G'$ and let $H$ be the fill-in of $G'$ given $f$. If we subsequently add $v$ and $w$ with their (formerly) adjacent edges to $H$, then both are simplicial in the resulting graph. The treewidth of the ordering $(v;w;f)$ of $G$, therefore, equals the maximum of 3 and the treewidth of $f$. The properties stated in the proposition now follow. ■

From Proposition 3, we obtain safeness of the **buddy rule**, which is illustrated in Figure 8.

**Reduction Rule 3: Buddy rule**

Let $v$, $w$ be vertices of degree three having the same set of neighbors.

If $\text{low} \geq 3$, then

- add an edge between every pair of nonadjacent neighbors of $v$;
- remove $v$;
- remove $w$.

**Lemma 4.** The buddy rule is safe.

**Proof.** Suppose $G'$ is obtained from $G$ by applying the buddy rule, with vertices $v$ and $w$ removed and their neighbors turned into a clique. We have $\max(\tau(G), \text{low}) \leq \max(3, \tau(G'), \text{low}) = \max(\tau(G'), \text{low}) \leq \max(\tau(G), \text{low})$. ■

**Figure 8.** The buddy rule.
The *cube rule*, which is presented schematically in Figure 9, is slightly more complicated. The subgraph shown on the left is replaced by the subgraph on the right; in addition, $\text{low}$ is set to $\max(\text{low}, 3)$. Vertices $v$, $w$, and $x$ in the subgraph may be adjacent to other vertices in the rest of the graph; the four nonlabeled vertices occurring in the rule cannot have such ‘outside’ edges. There is a slightly more general form of the cube rule, which we call the *extended cube rule*, given in Figure 10, whose correctness we prove first.

**Proposition 4.** Let $G = (V, E)$ be an undirected graph; $a, b, c, d, v, w, x \in V$. Suppose $a, b, c$, have degree three in $G$, with edges $\{a, d\}, \{a, v\}, \{b, d\}, \{b, v\}, \{b, x\}, \{c, d\}, \{c, w\}, \{c, x\} \in E$. Let $G'$ be obtained by making $v, w, x,$ and $d$ mutually adjacent, and removing $a, b,$ and $c$ from $G$. Then,

- $\tau(G) = \max(3, \tau(G'))$;
- there is a linear ordering $(a; b; c; f)$ of $G$ of minimum treewidth, where $f$ is a linear ordering of $G'$ of treewidth at most $\max(3, \tau(G'))$.

**Proof.** Because the subgraph of $G$, induced by $a, b, c, d, v, w,$ and $x$ has treewidth three, $\tau(G) \geq 3$. Because $G'$ is a minor of $G$ (contract edges $\{a, v\}, \{b, x\},$ and $\{c, w\}$), $\tau(G) \geq \tau(G')$. If $f$ is a linear ordering of $G'$ with fill-in $H$, then note that the additional edges in the fill-in of the linear order $(a; b; c; f)$ all belong to $H$: the additional fill-in edges, created by $a, b,$ and $c$ all belong to $G'$ and hence also to $H$. Because the numbers of higher numbered neighbors of $a, b,$ and $c$ in the new fill-in all equal three, the treewidth of $(a; b; c; f)$ equals the maximum of 3 and the treewidth of $f$. The lemma now follows.

**Reduction Rule 4: Extended cube rule**

Let $a, b, c, d, v, w, x$ be as in Figure 10. If $\text{low} \geq 3$, then
- add an edge between every pair of nonadjacent vertices in $\{v, w, x, d\}$;
- remove $c$;
- remove $b$;
- remove $a$.

**Lemma 5.** The extended cube rule is safe.

**Proof.** If $G'$ is obtained from $G$ by applying the extended cube rule, then, by Proposition 4, $\max(\tau(G), \text{low}) = \max(\tau(G'), 3, \text{low}) = \max(\tau(G'), \text{low})$. 

![Figure 9. The cube rule.](image-url)
The ‘standard’ cube rule is a variant of the extended cube rule. The cube rule is one of the rules in a set that is sufficient to reduce graphs of treewidth three to the empty graph, and it has an easier and faster implementation than the extended cube rule.

**Reduction Rule 4: Cube rule**

Let \(a, b, c, d, v, w, x\) be as in Figure 9. If \(\text{low} \geq 3\), then
- add an edge between every pair of nonadjacent vertices in \(\{v, w, x\}\);
- remove \(d\);
- remove \(c\);
- remove \(b\);
- remove \(a\).

**Lemma 6.** The cube rule is safe.

Proof. The cube rule can be obtained by first applying the extended cube rule, and then applying the simplicial vertex rule: note that vertex \(d\) becomes simplicial after the extended cube rule is applied to the left-hand side graph of Figure 9.

The difference between cube rule and extended cube rule is that in the cube rule, \(d\) has degree three and is removed, while in the extended cube rule, the degree of \(d\) may be larger than three.

The subgraph in the left-hand side of the cube rule is not very likely to occur in the moralization of a probabilistic network’s digraph, although it is not impossible. The main reason of our interest in the rule is that it is one of the six rules that suffice for reducing graphs of treewidth three to the empty graph. Thus, if \(\text{low} \geq 3\) for a given nonempty graph and the islet, twig, series, triangle, buddy, and cube rules cannot be applied, then we know by a result of Arnborg and Proskurowski (1986), that the graph has treewidth at least four; hence in this case, \(\text{low}\) can be set to \(\max(\text{low}, 4)\). Matoušek and Thomas showed that special cases of the rules with additional degree restrictions on some of the vertices can be used and still are sufficient for recognizing graphs of treewidth three (Matoušek and Thomas 1991); this can lead to a linear time algorithm for recognizing and triangulating graphs of treewidth three.

To conclude this section, Figure 11 depicts a fragment of the well-known ALARM network, along with its moralization. The figure further shows how successive application of our reduction rules serves to reduce the moralization to a single vertex. In fact, the moralized
4. COMPUTATIONAL METHOD

The various reduction rules described in the previous section are employed within a computational method that implements preprocessing of probabilistic networks for triangulation. We argued that application of our rules may reduce a network’s moralized graph to the empty graph. The computational method complements this reduction by its reversal, thereby providing for the construction of a triangulation of minimum treewidth. For networks that cannot be triangulated optimally just by preprocessing, our reduction rules are combined with an algorithm that serves to find an optimal or close to optimal triangulation of a network’s reduced moralized graph.

The computational method takes for its input the directed acyclic graph \(D\) of a probabilistic network; it outputs a triangulation of the moralization of \(D\). The method uses a stack \(S\) to hold the eliminated vertices in the order in which they were removed during the graph’s reduction. Moreover, the value \(low\) maintains a lower bound for the treewidth of the original moralized graph; it is initialized at 1, or possibly any larger value that gives a lower bound for the treewidth of \(M(D)\). Because a node \(v\) and its parents in \(D\) are turned into a clique by the moralization, \(low\) can be initialized at the maximum indegree of a node in \(D\). The method now amounts to the following sequence of steps:

1. Moralize \(D\) and initialize \(G\) by \(M(D)\).
2. If a reduction rule can be applied to \(G\), execute it and modify \(G\) accordingly. Push each vertex thus removed onto the stack \(S\); if prescribed by the rule, update the lower bound \(low\). In case of a reduction by the cube rule, push the vertex marked \(d\) last onto \(S\). Repeat this step until the reduction rules are no longer applicable.
3. If \(G\) is not an empty graph, and \(low < 4\), then increase \(low\) by 1 and continue the reduction at step 2.
4. Let \(G\) be the graph that results after execution of the previous steps. Using an exact or heuristic algorithm, triangulate \(G\).
5. Let $H$ be the triangulation that results from step 4. Construct a perfect elimination scheme $f$ for $H$.
6. Until $S$ is empty, pop the top element $v$ from $S$ and replace $f$ by $(v; f)$.
7. Let $f'$ be the linear ordering resulting from the previous step. Construct the fill-in of $M(D)$ given $f'$.

The steps 1 through 3 of our computational method describe the reduction of the graph of a probabilistic network. In step 4, the graph that results after reduction is triangulated. For this purpose, various different algorithms can be used. If the algorithm employed is exact, that is, if it yields a triangulation of minimum treewidth, then our method yields an optimal triangulation for the original moralized graph. An example of such an exact algorithm can be found in the work of Shoikhet and Geiger (1997), where an implementation is given of a variant of an algorithm of Arnborg et al. (1987), that appears practical for small size networks. For many real-life networks, the combination of our reduction rules with an exact algorithm results in an optimal triangulation in reasonable time. If after reduction a graph of considerable size remains for which an optimal triangulation cannot be feasibly computed, a heuristic triangulation algorithm can be used. The treewidth yielded for the original moralized graph then is not guaranteed to be optimal. As we will argue in the next section, however, these heuristic algorithms tend to result in better triangulations for the graphs that result from preprocessing than for the original graphs. If, after executing the steps 1 through 3, the reduced graph is empty, we can construct a triangulation of minimum treewidth for the moralized graph just by reversing the various reduction steps, and further triangulation is not necessary. This situation occurs, for example, if the original graph is already triangulated or has treewidth at most 3. The ALARM network gives another example: its moralized graph has treewidth four and is reduced to the empty graph.

In step 2 of our computational method, each of the reduction rules is investigated to establish whether or not it can be applied to the current (reduced) graph $G$. As soon as an applicable rule is found, it is executed. When analysing the computational complexity of our method, it is readily seen that investigating applicability of the various reduction rules is the main bottleneck, as all other steps (except for the triangulation in step 4) take linear time (Golumbic 1980).

Investigating applicability of the islet, twig, and series rules takes a total amount of computation time that is linear in the number of vertices. To this end, we maintain for each vertex an integer that indicates its degree; we further maintain lists of the vertices of degree zero, one, and two. The buddy, triangle, and cube rules can also be implemented to take overall linear time, for example, using techniques from (Arnborg et al. 1993), see also (Matoušek and Thomas 1991). More straightforward implementations, however, will also be fast enough for moderately sized networks.

For the simplicial vertex and almost simplicial vertex rule, efficient implementation is less straightforward. To investigate whether or not a vertex is simplicial, we must verify that each pair of its neighbors are adjacent. For this purpose, we have to use a data structure that allows for quickly checking adjacency, such as a two-dimensional array. For a vertex of degree $d$, investigating simpliciality then takes $O(d^2)$ time. In a graph with $n$ vertices, we may have to check for simplicial vertices $O(n)$ times. Each such check costs $O(\sum_v d(v)^2) = O(ne)$ time, where $d(v)$ is the degree of vertex $v$ and $e$ denotes the number of edges in the graph. The total time spent on investigating applicability of the simplicial vertex rule is therefore $O(n^2e)$. Because the treewidth of the moralized graph of a real-life probabilistic network is typically bounded, we can refrain from checking simpliciality for vertices of large degree, giving a running time of $O(n^2)$ in practice. For the almost simplicial vertex rule, similar observations apply. The extended cube rule can easily be implemented by first listing all
pairs of vertices of degree three that have two neighbors in common, and then checking for
each such pair all vertices of degree three if these three form a case where the extended
cube rule can be applied. With an adjacency list data structure one can check for each triple
of vertices in constant time whether they have neighbors as in Figure 10 and thus the rule
can be checked for in $O(n^3)$ time. In practice, this simple implementation is in general fast
enough.

With some additional efforts, the check for the existence of an extended cube rule can
be done in $O(n^2)$ time: For every pair of vertices $(a, b)$ of degree three with two common
neighbors, we store the possible triples which can play the role of $(w, x, d)$, cf. Figure 10. In
addition, for each vertex of degree three, its set of neighbors is stored as well. With radix sort
(Cormen et al. 2001, Section 8.3) the set of triples is sorted in $O(n^2)$ time. If the extended
cube rule can be applied, there are two successive entries with the same triple, one the neighbor
set of a vertex, and one stored for a pair of vertices $(a, b)$.

5. EXPERIMENTAL RESULTS

The computational method outlined in the previous section implements our method of
preprocessing probabilistic networks for triangulation. We conducted some experiments with
the method to study the effect of preprocessing. The results of these experiments are reported
in this section.

The experiments were conducted on 24 real-life probabilistic networks in the fields of
medicine, agriculture, water purification, and maritime use. The sizes of the digraphs of these
networks and of their moralizations, expressed in terms of the number of vertices and the
number of arcs and edges, respectively, are given in Table 1.

The effects of employing various different sets of reduction rules on the twenty-four
networks under study are summarized in Table 2. The various sets employed are denoted:

\[
\begin{align*}
\text{simplicial} & = \{\text{simplicial vertex}\} \\
\tau \leq 1 & = \{\text{islet, twig}\} \\
\tau \leq 2 & = (\tau \leq 1) \cup \{\text{series}\} \\
\tau \leq 3 & = (\tau \leq 2) \cup \{\text{triangle, buddy, cube}\} \\
\text{all} & = \text{simplicial} \cup (\tau \leq 3) \cup \{\text{almost simplicial vertex, extended cube}\}
\end{align*}
\]

With each of these sets of rules, the moralizations of the networks’ graphs were reduced
until the rules were no longer applicable. The table reports the sizes of the resulting reduced
digraphs. The computation times reported in the last column of the table are measured for the
case that all rules are applied. All computations have been carried out on a Linux-operated
PC with a 1700 MHz Intel Pentium 4 processor. C++ was used as programming language.

Table 2 reveals, for example, that the set of rules $\tau \leq 3$ suffices for reducing the moralized
graphs of four of the networks to the empty graph; with the additional simplicial vertex rule
and almost simplicial vertex rule, the moralized graphs of four other networks are also
reduced to the empty graph. These eight networks are, therefore, triangulated optimally just
by preprocessing.

Application of the simplicial vertex rule only reduces the number of vertices by 51% on
average, whereas overall the average percentage is 77%. Even for the worst performing in-
stances still a reduction of 30% is achieved (e.g., OOW-TRAD and WATER). With the exception
of PIGNET2 the computation time is marginal. Also for PIGNET2, the time is still justifiable
taking into account that more than 2000 vertices are removed.

Table 3 shows the differences of effectiveness of the various rules. This table was built
by checking all listed rules from left to right until one is applicable after every reduction
Table 1. Moralization of Probabilistic Networks

| Instance       | \(|V|\) | \(|A|\) | \(|E|\) |
|---------------|--------|--------|--------|
| ALARM         | 37     | 46     | 65     |
| BARLEY        | 48     | 84     | 126    |
| BOBLO         | 221    | 254    | 328    |
| DIABETES      | 413    | 602    | 819    |
| LINK          | 724    | 1125   | 1738   |
| MILDEW        | 35     | 46     | 80     |
| MUNIN1        | 189    | 282    | 366    |
| MUNIN2        | 1003   | 1244   | 1662   |
| MUNIN3        | 1044   | 1315   | 1745   |
| MUNIN4        | 1041   | 1397   | 1843   |
| MUNIN-KGO     | 1066   | 1278   | 1730   |
| OESOCA+       | 67     | 123    | 208    |
| OESOCA        | 39     | 55     | 67     |
| OESOCA42      | 42     | 59     | 72     |
| OOW-BAS       | 27     | 36     | 54     |
| OOW-SOLO      | 40     | 58     | 87     |
| OOW-TRAD      | 33     | 47     | 72     |
| PATHFINDER    | 109    | 192    | 211    |
| PIGNET2       | 3032   | 5400   | 7264   |
| PIGS          | 441    | 592    | 806    |
| SHIP-SHIP     | 50     | 75     | 114    |
| VSD           | 38     | 52     | 62     |
| WATER         | 32     | 66     | 123    |
| WILSON        | 21     | 23     | 27     |

of the graph, thus, for example, the extended cube rule is only checked when no other rule can be applied. We see that the simplicial vertex rule and almost simplicial vertex rule and their special cases are effective, but that the buddy, cube, and extended cube rule are never applied. The use of the latter rules is that checking these can help to increase the value of low to four, thus possibly enabling an almost simplical rule for a vertex of degree four. The effectiveness of the simplicial vertex rule in comparison with the almost simplicial vertex rule differs from network to network. In many cases the simplicial vertex rule (and its specializations) is responsible for the majority of the removals. However, for some instances (e.g., DIABETES, SHIP-SHIP) the almost simplicial vertex rule, and in particular the triangle rule, is very important.

We further studied the effect of preprocessing on the treewidths yielded by various heuristic triangulation algorithms. Table 4 summarizes the results obtained with two well-known heuristics for triangulation: the Greedy Fill-in heuristic, and the Minimum Degree Fill-In heuristic. In the Greedy Fill-in heuristic a linear ordering of the vertices is constructed by repeatedly selecting a vertex that causes the least fill-in in the triangulation (e.g., all simplicial vertices are ordered first). In the Minimum Degree Fill-in heuristic, repeatedly a vertex of
| Instance     | \( |V| \) | \( |E| \) | \( |V| \) | \( |E| \) | \( \tau \leq 1 \) | \( |V| \) | \( |E| \) | \( \tau \leq 2 \) | \( |V| \) | \( |E| \) | \( \tau \leq 3 \) | \( |V| \) | \( |E| \) | \( \text{low} \) | CPU Time (s) |
|-------------|------|------|------|------|-------------|------|------|-------------|------|------|-------------|------|------|-------------|------|------|-------------|--------|
| ALARM       | 37   | 65   | 11   | 19   | 4           | 31   | 59   | 13          | 28   | 5    | 10          | 0    | 0    | 4           | 0.00   |
| BARLEY      | 48   | 126  | 35   | 92   | 4           | 48   | 126  | 39          | 112  | 31   | 91          | 26   | 78   | 4           | 0.00   |
| BOBLO       | 221  | 328  | 71   | 132  | 2           | 117  | 224  | 70          | 131  | 0    | 0           | 0    | 0    | 3           | 0.09   |
| DIABETES    | 413  | 819  | 335  | 666  | 2           | 413  | 819  | 332         | 662  | 212  | 492         | 116  | 276  | 4           | 0.67   |
| LINK        | 724  | 1738 | 494  | 1349 | 3           | 641  | 1665 | 528         | 1439 | 472  | 1327        | 308  | 1158 | 4           | 1.58   |
| MILDEW      | 35   | 80   | 20   | 40   | 3           | 34   | 79   | 32          | 75   | 12   | 27          | 0    | 0    | 4           | 0.00   |
| MUNIN1      | 189  | 366  | 108  | 241  | 3           | 161  | 338  | 104         | 243  | 66   | 188         | 66   | 188  | 4           | 0.08   |
| MUNIN2      | 1003 | 1662 | 449  | 826  | 2           | 819  | 1478 | 367         | 736  | 175  | 471         | 165  | 451  | 4           | 2.34   |
| MUNIN3      | 1044 | 1745 | 419  | 790  | 3           | 852  | 1553 | 344         | 717  | 142  | 429         | 96   | 313  | 4           | 2.48   |
| MUNIN4      | 1041 | 1843 | 436  | 920  | 3           | 863  | 1665 | 379         | 869  | 237  | 686         | 215  | 642  | 4           | 2.47   |
| MUNIN-KGO   | 1066 | 1730 | 298  | 549  | 5           | 882  | 1546 | 207         | 470  | 104  | 298         | 0    | 0    | 5           | 2.46   |
| OESOCA+     | 67   | 208  | 30   | 141  | 9           | 48   | 189  | 34          | 162  | 30   | 150         | 14   | 75   | 9           | 0.01   |
| OESOCA      | 39   | 67   | 5    | 7    | 3           | 24   | 52   | 12          | 29   | 0    | 0           | 0    | 0    | 3           | 0.00   |
| OESOCA42    | 42   | 72   | 6    | 10   | 3           | 25   | 55   | 13          | 32   | 0    | 0           | 0    | 0    | 3           | 0.00   |
| OOW-BAS     | 27   | 54   | 19   | 37   | 3           | 27   | 54   | 20          | 42   | 8    | 18          | 0    | 0    | 4           | 0.00   |
| OOW-SOLO    | 40   | 87   | 31   | 68   | 3           | 39   | 86   | 33          | 76   | 29   | 66          | 27   | 63   | 4           | 0.01   |
| OOW-TRAD    | 33   | 72   | 27   | 59   | 3           | 33   | 72   | 27          | 63   | 23   | 54          | 23   | 54   | 4           | 0.01   |
| PATHFINDER  | 109  | 211  | 14   | 49   | 5           | 68   | 170  | 37          | 112  | 17   | 63          | 12   | 43   | 5           | 0.03   |
| PIGNET2     | 3032 | 7264 | 1643 | 4556 | 3           | 3032 | 7264 | 1552        | 4464 | 1051 | 3835        | 1002 | 3730 | 4           | 27.20  |
| PIGS        | 441  | 806  | 163  | 305  | 2           | 441  | 806  | 126         | 265  | 60   | 164         | 48   | 137  | 4           | 0.46   |
| SHIP-SHIP   | 50   | 114  | 39   | 92   | 3           | 50   | 114  | 41          | 98   | 30   | 77          | 24   | 65   | 4           | 0.02   |
| VSD         | 38   | 62   | 12   | 21   | 4           | 23   | 47   | 12          | 28   | 6    | 14          | 0    | 0    | 4           | 0.00   |
| WATER       | 32   | 123  | 24   | 101  | 5           | 30   | 121  | 29          | 119  | 26   | 110         | 22   | 96   | 5           | 0.00   |
| WILSON      | 21   | 27   | 6    | 8    | 2           | 11   | 17   | 4           | 6    | 0    | 0           | 0    | 0    | 3           | 0.00   |
TABLE 3. Contribution of the Various Rules

| Instance   | $|V|$  | $|E|$ | IS | TW | SI | SE | TR | AS | BU | CU | EC | Total |
|------------|------|------|-----|----|----|----|----|----|----|----|----|-----|-------|
| ALARM      | 37   | 70   | 1   | 11 | 21 | 1  | 3  | 0  | 0  | 0  | 0  | 0   | 37    |
| BARLEY     | 48   | 139  | 0   | 1  | 13 | 3  | 2  | 3  | 0  | 0  | 0  | 0   | 22    |
| BOBLO      | 221  | 373  | 1   | 105| 80 | 11 | 24 | 0  | 0  | 0  | 0  | 0   | 221   |
| DIABETES   | 413  | 1085 | 0   | 2  | 124| 3  | 143| 25 | 0  | 0  | 0  | 0   | 297   |
| LINK       | 724  | 2257 | 10  | 73 | 147| 12 | 10 | 164| 0  | 0  | 0  | 0   | 416   |
| MILDEW     | 35   | 99   | 1   | 2  | 18 | 1  | 8  | 5  | 0  | 0  | 0  | 0   | 35    |
| MUNIN1     | 189  | 431  | 0   | 40 | 42 | 5  | 34 | 0  | 0  | 0  | 0  | 0   | 123   |
| MUNIN2     | 1003 | 2065 | 0   | 272| 300| 74 | 182| 10 | 0  | 0  | 0  | 0   | 838   |
| MUNIN3     | 1044 | 2178 | 0   | 298| 368| 76 | 180| 26 | 0  | 0  | 0  | 0   | 948   |
| MUNIN4     | 1041 | 2183 | 0   | 290| 324| 60 | 136| 16 | 0  | 0  | 0  | 0   | 826   |
| MUNIN-KGO  | 1066 | 2042 | 1   | 365| 476| 96 | 78 | 50 | 0  | 0  | 0  | 0   | 1066  |
| OESOCA+    | 67   | 247  | 0   | 19 | 19 | 1  | 14 | 0  | 0  | 0  | 0  | 0   | 53    |
| OESOCA     | 39   | 68   | 1   | 16 | 21 | 1  | 0  | 0  | 0  | 0  | 0  | 0   | 39    |
| OESOCA42   | 42   | 73   | 1   | 18 | 22 | 1  | 0  | 0  | 0  | 0  | 0  | 0   | 42    |
| OOW-BAS    | 27   | 69   | 1   | 2  | 13 | 1  | 8  | 2  | 0  | 0  | 0  | 0   | 27    |
| OOW-SOLO   | 40   | 95   | 0   | 2  | 7  | 1  | 1  | 2  | 0  | 0  | 0  | 0   | 13    |
| OOW-TRAD   | 33   | 77   | 0   | 1  | 5  | 2  | 2  | 0  | 0  | 0  | 0  | 0   | 10    |
| PATHFINDER | 109  | 213  | 0   | 47 | 48 | 0  | 2  | 0  | 0  | 0  | 0  | 0   | 97    |
| PIGNET2    | 3032 | 8311 | 0   | 71 | 1341| 89 | 481| 48 | 0  | 0  | 0  | 0   | 2030  |
| PIGS       | 441  | 948  | 0   | 57 | 236 | 34 | 55 | 11 | 0  | 0  | 0  | 0   | 393   |
| SHIP-SHIP  | 50   | 132  | 0   | 2  | 11 | 0  | 11 | 2  | 0  | 0  | 0  | 0   | 26    |
| VSD        | 38   | 69   | 1   | 16 | 15 | 3  | 3  | 0  | 0  | 0  | 0  | 0   | 38    |
| WATER      | 32   | 127  | 0   | 2  | 6  | 0  | 2  | 0  | 0  | 0  | 0  | 0   | 10    |
| WILSON     | 21   | 29   | 1   | 12 | 6  | 2  | 0  | 0  | 0  | 0  | 0  | 0   | 21    |

IS = Islet, TW = Twig, SI = Simplicial, SE = Series, TR = Triangle, AS = Almost simplicial, BU = Buddy, CU = Cube, EC = Extended cube

minimum degree is selected and removed from the graph (e.g., for the unpreprocessed graph, vertices that are removed by the islet, twig, and series rule are ordered first). We see that sometimes, but not always, the reduced graphs give better bounds for the treewidth obtained with these heuristics. In addition, a lower bound low for the treewidth is obtained which allows for an estimation of the quality of the heuristics. More precisely, for DIABETES we can conclude that the treewidth is four by combining the low with the Minimum Degree Fill-In heuristic. Therefore, a possible approach is to run the heuristics both for the original and for the reduced graph, and take the best value. We would like to note that, using integer linear programming techniques on the most reduced graph, we found the exact treewidth of the PATHFINDER network to be 6.

6. CONCLUSIONS AND FURTHER RESEARCH

When solving hard combinatorial problems, preprocessing is often profitable. Based upon this general observation, we designed a computational method that provides for preprocessing
of probabilistic networks for triangulation. Our method exploits a set of rules for stepwise reducing the problem of finding a triangulation of minimum treewidth for a network’s moralized graph to the same problem on a smaller graph. The smaller graph is triangulated, using an exact or heuristic algorithm, depending on the graph’s size. From the triangulation of the smaller graph, a triangulation of the original graph is obtained by reversing the reduction steps. The reduction rules are guaranteed not to destroy optimality with respect to maximum clique size.

Experiments with our preprocessing method revealed that the graphs of some well-known real-life probabilistic networks can be triangulated optimally just by preprocessing. In contrast to the Minimum Degree Fill-In heuristic, the rules guarantee that the optimal triangulation is found. The experiments further showed that heuristic triangulation algorithms regularly yield better results for graphs that are preprocessed than for the original graphs. Moreover, if an exact algorithm for triangulating the remaining graph is used, such as the branch and bound in Gogate and Dechter (2004), each reduction in the size of the graph can be expected to cause a significant reduction in the running time. From these observations, we conclude that preprocessing probabilistic networks for triangulation is profitable.
The preprocessing rules given in this paper can also applied successfully for finding tree decompositions of networks, arising in applications from fields, different from probabilistic networks. For instance, in Koster, Bodlaender, and van Hoe sel (2001), experiments on determining the treewidth of networks are reported, including a successful application of the reduction rules to instances arising from a frequency assignment application. See Treewidthlib (2004) for these and other results.

It is possible to also apply other rules for preprocessing purposes. For example, Sanders designed a set of rules for reducing any graph of treewidth at most four to the empty graph (Sanders 1996). Although this set is comprised of a large number of complex rules and many of these rules do not have the property that a linear ordering with minimum treewidth of the graph can be directly obtained from a linear ordering with minimum treewidth of the reduced graph (see also Lagergren 1994), it may give rise to new reduction rules that can be employed for preprocessing.

So far, we considered the use of rules for reducing the graph of a probabilistic network. Recently we have investigated the use of separators for preprocessing (Bodlaender and Koster 2004), building upon earlier work by Olesen and Madsen (2002). For example, if a network’s moralized graph has a separator of size two, then the graph can be partitioned into smaller graphs that can be triangulated separately without losing optimality.

Because there is a strong relationship between the running time of the junction-tree propagation algorithm and the treewidth of the triangulation used, most triangulation algorithms currently in use aim at finding a triangulation of minimum treewidth. However, if the variables in a probabilistic network have state spaces of diverging sizes, such a triangulation may not be optimal. A triangulation with minimal state space over all cliques then is likely to perform better. Some of our reduction rules are safe also with respect to minimum overall state space. Other rules, however, are safe only under additional constraints on their application. It is interesting to investigate preprocessing for finding triangulations with minimum overall state space. Recently, we have studied a weighted variant of treewidth (Eijkhof and Bodlaender 2002); in this variant, vertices have a weight equal to the number of values the corresponding variable can attain in the probabilistic network. In Eijkhof and Bodlaender (2002), we generalize the rules given in this paper to the weighted variant, and show most of these rules can be obtained as a special case of one general rule, called the Contraction Reduction Rule.

In our experiments, we have observed that applying rules in a different order never affected the size of the finally resulting reduced network. We conjecture that the set of rules, given in this paper is actually confluent, that is, changing the order in which the rules are applied does not affect the final outcome, up to isomorphism of graphs. We were unable to prove or disprove this conjecture, so we leave it as an open problem.

ACKNOWLEDGMENTS

We thank Linda van der Gaag for many very useful discussions, ideas, and comments on this paper. We are grateful to the members of the Decision Support Systems group of the Institute of Information and Computing Sciences, Utrecht University, and in particular to Silja Renooij for several usefull comments on this paper. We thank Kristian Kristensen, Anders L. Madsen, Kristian G. Olesen, Claus Skaaning Jensen, and Linda van der Gaag for providing instances of probabilistic networks.
REFERENCES


