Alcom-FT Technical Report Series
ALCOMFT-TR-03-121

Pre-Processing Rules for Triangulation of Probabilistic Networks

Hans L. Bodlaender

Arie M. C. A. Koster

Frank van den Eijkhof

institute of information and computing sciences,
utrecht university

technical report UU-CS-2003-001

www.cs.uu.nl

Pre-Processing Rules for Triangulation of Probabilistic Networks*

Hans L. Bodlaender† Arie M.C.A. Koster‡ Frank van den Eijkhof§

Abstract

The currently most efficient algorithm for inference with a probabilistic network builds upon a triangulation of a network's graph. In this paper, we show that pre-processing can help in finding good triangulations for probabilistic networks, that is, triangulations with a minimal maximum clique size. 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.

1 Introduction

The currently most efficient algorithm for inference with a probabilistic network is the *junction-tree propagation* algorithm that builds upon a *triangulation* of a network's moralised graph [10, 8]. 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. As 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

*The first author was partially supported by EC contract IST-1999-14186: Project ALCOM-FT (Algorithms and Complexity – Future Technologies). The work of the third author was done while he was working at the Institute of Information and Computing Sciences, Utrecht University, with partial support by the Netherlands Computer Science Research Foundation with financial support from the Netherlands Organisation for Scientific Research. This work was partially carried out in the project *Treewidth and Combinatorial Optimization* with financial support from the Netherlands Organisation for Scientific Research.

†Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. hansb@cs.uu.nl

‡Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustrasse 7, D-14195 Berlin-Dahlem, Germany. koster@zib.de

§Department of Methodology and Statistics, Faculty of Social Sciences, Utrecht University, P.O. Box 80.0140, 3508 TC Utrecht, the Netherlands. f.vandeneijkhof@fss.uu.nl

notion, both by researchers in the field of probabilistic networks and by researchers in graph theory and graph algorithms. 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 [1].

When solving hard combinatorial problems, *pre-processing* 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 pre-processing for triangulation of probabilistic networks. We provide a set of rules for stepwise reducing the problem of finding a triangulation for a network's moralised 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 pre-processing 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 pre-processing.

In this paper, we do not address the second phase in the pre-processing approach outlined above, that is, we do not address actually constructing triangulations with a minimal or close to minimal maximum clique size. 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 [1], Shoikhet and Geiger performed various experiments on randomly generated graphs [14]. Their results indicate that this algorithm allows for computing optimal triangulations of graphs with up to 100 vertices.

The paper is organised as follows. In Section 2, we review some basic definitions. In Section 3, we present our pre-processing rules. The computational model in which these rules are employed, is discussed in Section 4. In Section 4, 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

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

Let $D = (V, A)$ be a directed acyclic graph. The *moralisation* of D is the undirected graph $M(D)$ obtained from D by adding edges between every pair of non-adjacent 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. 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*. 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 $G[V - \{v\}]$. 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 neighbours of v and w). It is well known (see for example [4]), 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 neighbours 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. It is well known (see for example [6]), that a graph is triangulated if and only if it allows a perfect elimination scheme.

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 E. 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 neighbours 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 neighbours of a vertex [6].

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 for example [6]), 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. 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. As all rules that are applied are safe, we have that $\max(\tau(G), low) = \max(\tau(G'), low')$. (This can be shown with induction to the number of reduction rules that are applied.) Using $low \leq \tau(G)$, the result follows. □

The algorithmic technique used in this paper for pre-processing 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 *low*: as discussed above, *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 *low* to *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 *low*. In the sequel, we assume that the original moralised graph G has at least one edge and that *low* is initialised 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 *low* to the value computed by the heuristic. As 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 neighbours 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 2 *Let G be an undirected graph and let v be a simplicial vertex in G with degree $d \geq 0$. Then,*

- $\tau(G) = \max(d, \tau(G - v))$*;*
- *there is a linear ordering $(v; f)$ of G of minimum treewidth, where f is a linear ordering of $G - v$ of treewidth at most* $\max(d, \tau(G - v))$.

Proof. Since 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 neighbours 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. $\square$

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 *low* to $\max(\textit{low}, d)$.

The second property stated in Proposition 2 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 3 *The* simplicial vertex rule *is safe.*

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

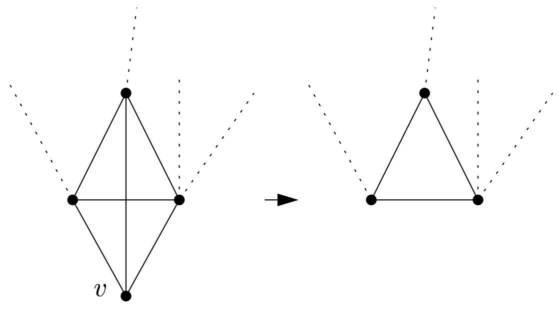

Figure 1: The *simplicial vertex rule*.

Because the digraph D of a probabilistic network is moralised before it is triangulated, it is likely to give rise to many simplicial vertices. We consider a vertex v with outdegree zero in D. Since all neighbours of v have an arc pointing into v, moralisation will connect every two neighbours 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. As every directed acyclic graph has at least one vertex of outdegree zero, at least one reduction will be performed. As the reduced graph need not be the moralisation 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 moralisation of D. We consider a vertex v with indegree zero and a single arc pointing into a vertex w. In the moralisation of D, w and its (former) predecessors constitute a clique. As all neighbours 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 [3].

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. As we assumed that we started with *low* ≥ 1, there is no need to update *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 neighbour w of v such that all other neighbours of v form a clique in G. Figure 2 illustrates the basic idea of an almost simplicial vertex. Note that we allow other neighbours of v to be adjacent to w. Simplicial vertices therefore are also almost simplicial.

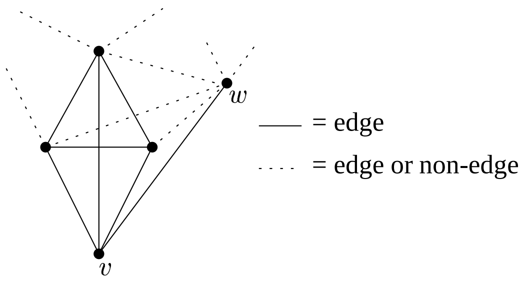

Figure 2: An almost simplicial vertex.

Proposition 4 *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 neighbours of v into a clique and then removing v. Then,*

- $\tau(G') \leq \tau(G)$ *and* $\tau(G) \leq \max(d, \tau(G'))$*;*
- *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 neighbour of v such that the other neighbours of v form a clique. As 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 2, we find that $\tau(H') = \max(k, d)$. The two properties stated in the proposition follow. □

Our second reduction rule, illustrated in Figure 3, now is:

> **Reduction Rule 2: Almost simplicial vertex rule**
> Let v be an almost simplicial vertex of degree $d \geq 0$.
> If *low* $\geq d$, then
> add an edge between every pair of non-adjacent neighbours of v;
> remove v.

Lemma 5 *The* almost simplicial vertex rule *is safe.*

Proof. Suppose $(G, \mathrm{low}) \rightarrow_R (G', \mathrm{low}')$ by the *almost simplicial vertex rule*. Then, $\tau(G') \leq \tau(G)$, $\tau(G) \leq \max(d, \tau(G'))$, and $d \leq$ *low* $=$ *low*$'$. We conclude that $\max(\tau(G),$*low*$) = \max(\tau(G'),$ *low*$')$. □

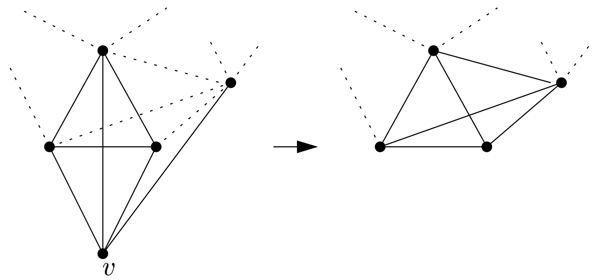

Figure 3: The *almost simplicial vertex rule*.

Examples can be constructed, unfortunately, that show that the rule is not safe for *low* $< d$. Let G be the graph in the left hand side of Figure 4, and suppose we have $low = 2$. G has treewidth three (as 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. It is also possible to construct more complicated similar examples to which also no other reductions can be applied.

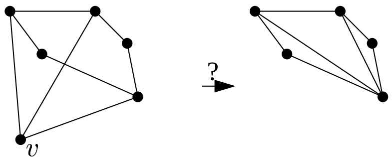

Figure 4: The *almost simplicial vertex rule* is not safe for *low* $< d$.

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 moralisation $M(D)$: note that, as v has one child in D which has only v as parent, no moralisation edges will be added with v as endpoint. Thus, the neighbours of v in $M(D)$ are its parents in D and w; due to the moralisation, the parents of v form a clique in $M(D)$. See Figure 5.

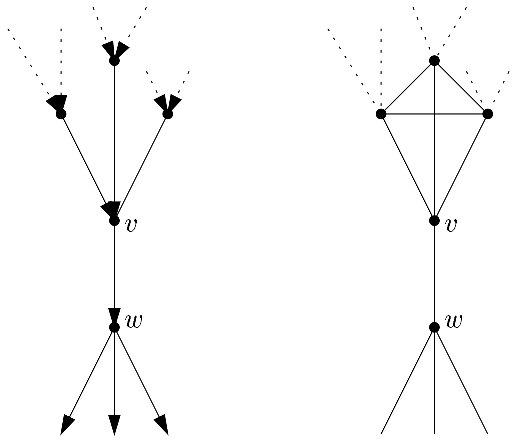

Figure 5: A case where an almost simplicial vertex is created when moralizing

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 neighbours, provided that the original graph has treewidth at least two. The resulting rule, illustrated in Figure 6, is called the *series rule*, after [3].

Reduction Rule 2a: Series rule
Let v be a vertex of degree 2.
If *low* ≥ 2, then
  add an edge between the neighbours 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 neighbours are adjacent.

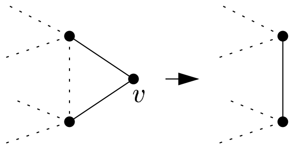

Figure 6: The *series rule*.

If *low* ≥ 3, then
add an edge between every pair of non-adjacent neighbours of v;
remove v.

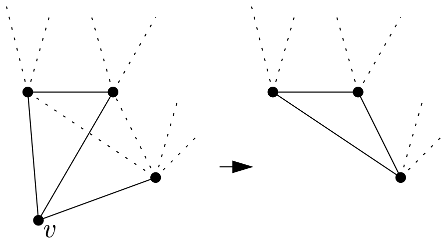

Figure 7: The *triangle rule*.

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

Using the fact that a non-empty 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 non-empty 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 non-empty 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.) So, if *low* ≥ 2 for a given non-empty 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*$, 3)$.

As for 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 [3]. 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 pre-processing.

Proposition 6 *Let G be an undirected graph and let v, w be two vertices of degree three having* *the same set of neighbours. Let G' be the graph that is obtained from G by turning the set of neighbours 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 neighbours of v and w. By contracting the edges $\{v, x\}$ and $\{w, y\}$ in G, we obtain G'. So, 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. $\square$

From Proposition 6, we obtain safeness of the *buddy rule*, which is illustrated in Figure 8.

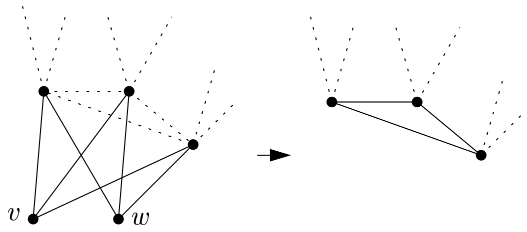

Figure 8: The *buddy rule*.

Reduction Rule 3: Buddy rule
Let v, w be vertices of degree 3 having the same
set of neighbours.
If *low* ≥ 3, then
   add an edge between every pair of non-adjacent neighbours of v;
   remove v;
   remove w.

Lemma 7 *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 neighbours turned into a clique. We have $\max(\tau(G), low) \leq \max(3, \tau(G'), low) = \max(\tau(G'), low) \leq \max(\tau(G), low)$. $\square$

The *cube rule*, which is presented schematically in Figure 10, is slightly more complicated. The subgraph shown on the left is replaced by the subgraph on the right; in addition, *low* is set to $\max(low, 3)$. Vertices v, w and x in the subgraph may be adjacent to other vertices in the rest of the graph; the four non-labeled 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 9, whose correctness we prove first.

Proposition 8 *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 $\{a, d\}$, $\{a, v\}$, $\{a, w\}$, $\{b, d\}$, $\{b, v\}$, $\{b, x\}$, $\{c, d\}$, $\{c, w\}$, $\{c, x\} \in E$. 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. As the subgraph of G, induced by a, b, c, d, v, w, and x has treewidth three, $\tau(G) \geq 3$. As 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. As the numbers of higher numbered neighbours 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. □

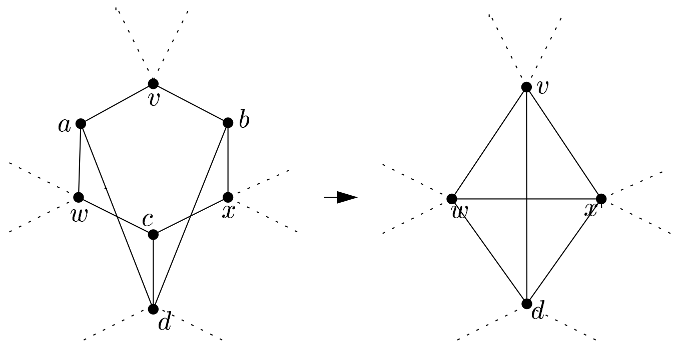

Figure 9: The *extended cube rule*.

Reduction Rule 4: Extended cube rule
Let a, b, c, d, v, w, x be as in Figure 9. If *low* ≥ 3, then
add an edge between every pair of non-adjacent vertices in $\{v, w, x, d\}$;
remove c;
remove b;
remove a.

Lemma 9 *The extended cube rule is safe.*

Proof. If G' is obtained from G by applying the extended cube rule, then, by Proposition 8, $\max(\tau(G), low) = \max(\tau(G), 3, low) = \max(\tau(G'), low)$. $\square$

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 10. If *low* ≥ 3, then
add an edge between every pair of non-adjacent vertices in $\{v, w, x\}$;
remove d;
remove c;
remove b;
remove a.

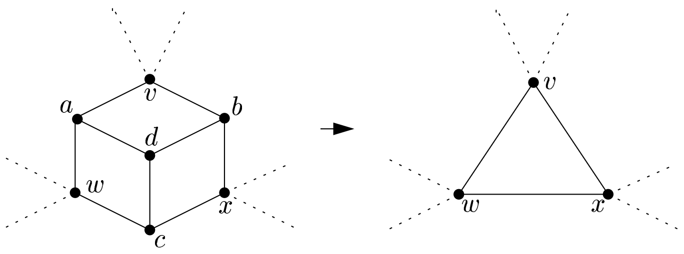

Figure 10: The *cube rule*.

Lemma 10 *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 10. $\square$

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 moralisation 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. So, if *low* ≥ 3 for a given non-empty graph and the *islet*, *twig*, *series*, *triangle*, *buddy* and *cube rules* cannot be applied, then we know by a result of Arnborg and Proskurowski [3], that the graph has treewidth at least four; hence in this case, *low* can be set to

$\max(low, 4)$. 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 [11]; 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 moralisation. The figure further shows how successive application of our reduction rules serves to reduce the moralisation to a single vertex. In fact, the moralised graph of the entire ALARM network is thus reduced to the empty graph. Our reduction rules provide for constructing an optimal triangulation of this network.

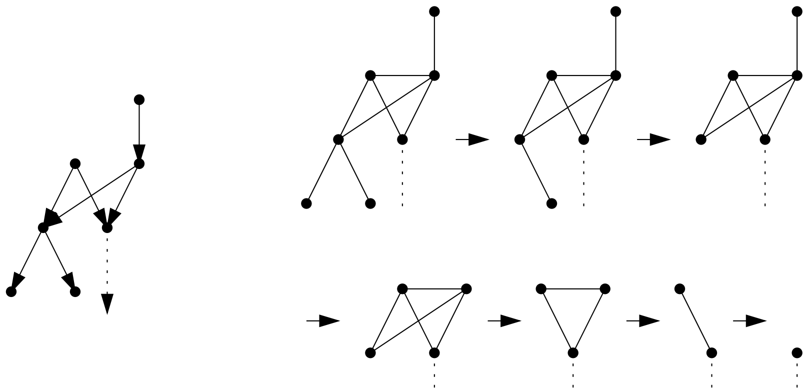

Figure 11: A fragment of the ALARM network and the reduction of its moralisation.

4 Computational method

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

The computational method takes for its input the directed acyclic graph D of a probabilistic network; it outputs a triangulation of the moralisation 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 moralised graph; it is initialised at 1, or possibly any larger value that gives a lower bound for the treewidth of $M(D)$. E.g., *low* can be initialized at the maximum indegree of a node in D, as for every node v, $M(D)$ will contain a clique contain v and its parents in D, and the treewidth of a graph is at least its maximum clique size minus one. The method now amounts to the following sequence of steps:

1. The moralisation $M(D)$ of D is computed and G is initialised at $M(D)$.
2. If a reduction rule can be applied to G, it is executed, modifying G accordingly. Each vertex thus removed is pushed onto the stack S; if prescribed by the rule, the lower bound *low* is updated. In case of a reduction by the *cube rule*, the vertex marked d must be pushed last onto S. This step is repeated until the reduction rules are no longer applicable.
3. If no reduction rule can be applied, G is not an empty graph, and *low* < 4, then *low* is increased by 1. The reduction is continued at step 2.
4. Let G be the graph that results after execution of the previous steps. Using an exact or heuristic algorithm, G is triangulated.
5. Let H be the triangulation that results from step 4. For H, a perfect elimination scheme f is constructed.
6. Until S is empty, the top element v is popped from S and f is replaced by $(v; f)$.
7. Let f' be the linear ordering resulting from the previous step. The fill-in of $M(G)$ given f' is constructed.

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 minimal treewidth, then our method yields an optimal triangulation for the original moralised graph. An example of such an exact algorithm can be found in the work of Shoikhet and Geiger [14], where an implementation is given of a variant of an algorithm of Arnborg, Corneil, and Proskurowski [1], 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 moralised 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 pre-processing than for the original graphs. If, after executing the steps 1 through 3, the reduced graph is empty, we can construct a triangulation of minimal treewidth for the moralised 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 moralised 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 [6].

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 [2], see also [11]. 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 neighbours 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 $O(n^2 e)$. As the treewidth of the moralised 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 neighbours 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. This gives an $O(n^3)$ check to see if the extended cube rule can be applied; in practice, this simple implementation is in general fast enough. With some additional efforts, the check can be done in $O(n^2)$ time.

5 Experimental results

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

The experiments were conducted on twenty-four 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 moralisations, 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 summarised in Table 2. The various sets employed are denoted:

simplicial	=	$\{$*simplicial vertex*$\}$
$\tau \leq 1$	=	$\{$*islet, twig*$\}$
$\tau \leq 2$	=	$(\tau \leq 1) \cup \{$*series*$\}$
$\tau \leq 3$	=	$(\tau \leq 2) \cup \{$*triangle, buddy, cube*$\}$
all	=	simplicial $\cup$ $(\tau \leq 3)$ $\cup$ $\{$ *almost simplicial vertex, extended cube* $\}$

With each of these sets of rules, the moralisations of the networks' graphs were reduced until

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

Table 1: Moralisation of probabilistic networks

the rules were no longer applicable. The table reports the sizes of the resulting reduced graphs. 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 moralised graphs of four of the networks to the empty graph; with the additional *simplicial vertex rule* and *almost simplicial vertex rule*, the moralised graphs of four other networks are also reduced to the empty graph. These eight networks are therefore triangulated optimally just by pre-processing.

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 instances 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 of the graph, so, e.g., 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 specialisations) is responsable 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 pre-processing on the treewidths yielded by various heuristic triangulation algorithms. Table 4 summarises 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 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 precies, 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.

instance	original		simplicial			$\tau \leq 1$		$\tau \leq 2$		$\tau \leq 3$		all			CPU
	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert E\rvert$	low	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert E\rvert$	low	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	665	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 2: Preprocessing for treewidth

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

Table 3: Contribution of the various rules

instance	original		preprocessed			Greedy Fill-In		Min Degree Fill-In	
	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	low	before	after	before	after
ALARM	37	65	0	0	4	4	-	4	-
BARLEY	48	126	26	78	4	9	7	7	7
BOBLO	221	328	0	0	3	4	-	3	-
DIABETES	413	819	116	276	4	10	7	4	5
LINK	724	1738	308	1158	4	21	21	19	19
MILDEW	35	80	0	0	4	5	-	4	-
MUNIN1	189	366	66	188	4	12	12	11	11
MUNIN2	1003	1662	165	451	4	8	8	7	7
MUNIN3	1044	1745	96	313	4	8	7	7	7
MUNIN4	1041	1843	215	642	4	9	9	8	8
MUNIN-KGO	1066	1730	0	0	5	6	-	5	-
OESOCA+	67	208	14	75	9	11	11	11	11
OESOCA	39	67	0	0	3	4	-	3	-
OESOCA42	42	72	0	0	3	3	-	3	-
OOW-BAS	27	54	0	0	4	5	-	4	-
OOW-SOLO	40	87	27	63	4	6	7	6	7
OOW-TRAD	33	72	23	54	4	7	6	6	6
PATHFINDER	109	211	12	43	5	7	7	7	7
PIGNET2	3032	7264	1002	3730	4	144	148	160	150
PIGS	441	806	48	137	4	12	11	10	10
SHIP-SHIP	50	114	24	65	4	9	8	8	8
VSD	38	62	0	0	4	5	-	4	-
WATER	32	123	22	96	5	12	10	11	11
WILSON	21	27	0	0	3	4	-	3	-

Table 4: Performance of heuristics without/with preprocessing

6 Conclusions and further research

When solving hard combinatorial problems, pre-processing is often profitable. Based upon this general observation, we designed a computational method that provides for pre-processing 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 moralised 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 pre-processing method revealed that the graphs of some well-known real-life probabilistic networks can be triangulated optimally just by pre-processing. The experiments further showed that heuristic triangulation algorithms tend to yield better results for graphs that are pre-processed than for the original graphs. Moreover, the further reduced a graph, the less computation time is spent by the triangulation algorithms. From these observations, we conclude that pre-processing probabilistic networks for triangulation is profitable.

The preprocessing rules given in this paper can also applied succesfully for finding tree decompositions of networks, arising in applications from fields, different from probabilistic networks. For instance, in [7], experiments on determining the treewidth of networks are reported, including a succesfull application of the reduction rules to instances arising from a frequency assignment application.

It is possible to also apply other rules for pre-processing purposes. For example, Sanders designed a set of rules for reducing any graph of treewidth at most four to the empty graph [13]. 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 [9]), it may give rise to new reduction rules that can be employed for pre-processing.

So far, we considered the use of rules for reducing the graph of a probabilistic network. The use of separators constitutes another approach to pre-processing that we are currently considering, building upon earlier work by Olesen and Madsen [12]. For example, if a network's moralised graph has a separator of size two, then the graph can be partitioned into smaller graphs that can be triangulated separately without losing optimality.

As 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 minimal 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 investigate pre-processing for finding triangulations with minimum overall state space. Recently, we have studied a weighted variant of treewidth [5]; in this variant, vertices have a weight equal to the number of values the corresponding variable can attain in the probabilistic network. In [5], 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*, i.e., 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 leave it as an open problem.

Acknowledgements

We thank Linda van der Gaag for many very usefull discussions, ideas, and comments on this paper. We are gratefull 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

[1] S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, vol. 8, pp. 277–284, 1987.

[2] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the ACM*, vol. 40, pp. 1134–1164, 1993.

[3] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM Journal on Algebraic and Discrete Methods*, vol. 7, pp. 305–314, 1986.

[4] H.L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, vol. 209, pp. 1–45, 1998.

[5] F. van den Eijkhof, H.L. Bodlaender, and A.M.C.A. Koster. Safe reduction rules for weighted treewidth. Technical Report UU-CS-2002-051, Institute of Information and Computing Science, Utrecht University, the Netherlands, 2002.

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

[7] A.M.C.A. Koster, H.L. Bodlaender, and C.P.M. van Hoesel. Treewidth: Computational Experiments. Technical Report UU-CS-2001-049, Institute of Information and Computing Science, Utrecht University, the Netherlands, 2001.

[8] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, vol. 4, pp. 269–282, 1990.

[9] J. Lagergren. The nonexistence of reduction rules giving an embedding into a k-tree. *Discrete and Applied Mathematics*, 54:219–223, 1994.

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

[11] J. Matoušek and R. Thomas. Algorithms finding tree-decompositions of graphs. *Journal of Algorithms*, 12:1–22, 1991.

[12] K.G. Olesen and A.L. Madsen. Maximal prime subgraph decomposition of Bayesian networks. Technical report, Department of Computer Science, Aalborg University, Aalborg, Denmark, 1999.

[13] D.P. Sanders. On linear recognition of tree-width at most four. *SIAM Journal on Discrete Mathematics*, vol. 9, pp. 101–117, 1996.

[14] K. Shoikhet and D. Geiger. A practical algorithm for finding optimal triangulations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 97)*, pp. 185–190. Morgan Kaufmann, 1997.