ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-759

Provisional Propagation for Verifying Monotonicity of Bayesian Networks

Merel T. Rietbergen[1] and **Linda C. van der Gaag** and **Hans L. Bodlaender**

Abstract. Many real-world Bayesian networks are expected to exhibit commonly known properties of monotonicity. Since monotonicity violations may be introduced despite careful engineering efforts, these properties need be verified before using a network in practice. We will show that the problem of verifying monotonicity in general has a prohibitively high computational complexity. We will argue however, that the runtime complexity involved can be substantially reduced by using a tailored algorithm which we coined provisional propagation. By means of this algorithm in fact, verifying monotonicity may become feasible for a range of real-world networks.

1 INTRODUCTION

Many real-world Bayesian networks are expected to exhibit properties of monotonicity, in the sense that higher values for the observable variables should make higher values for the main variable of interest more likely. In a medical diagnostic application, for example, observing more severe symptoms and signs should result in a more severe disease becoming a more probable diagnosis. If such knowledge is quite common, then a model that does not exhibit the associated monotonicity properties, will not be easily accepted by its users, not even if it shows high performance otherwise. Yet, during the construction of a Bayesian network, violations of monotonicity may inadvertently be introduced despite careful engineering efforts.

Since monotonicity properties are commonly found in real-life application domains, many modelling techniques have been tailored to capture such properties. Recently, the concept of monotonicity was introduced also for Bayesian networks [5]. We say that a network is monotone if the probability distribution computed for the variable of interest given specific observations is stochastically dominated by any such distribution given higher-ordered observations.

In this paper we study the concept of monotonicity and show that the MONOTONICITY problem of verifying whether monotonicity holds for a given Bayesian network, is complete in general for the complexity class coNP$^{\text{PP}}$. In their study of monotonicity [5], Van der Gaag and colleagues give an anytime algorithm for verifying monotonicity which uses qualitative influences to decide whether a network is monotone in a (sub)set of observable variables, iteratively using upper and lower bounds on probabilities to resolve yet undecided statements of dominance. In this paper we solve the MONOTONICITY problem by direct exact computation of all necessary probabilities. For this purpose we employ a new propagation algorithm which uses the structure of a lattice of values for a network's observable variables to guide the search for violations. While this new algorithm substantially reduces the runtime for verifying monotonicity in practice, its runtime complexity is exponential in the number of observable variables in the network under study. The unfavourable computational complexity of the problem however, is likely to forestall the design of essentially more efficient methods.

The paper is organised as follows. In Section 2, we provide some preliminaries on Bayesian networks, and on the concept of monotonicity and the notion of assignment lattice; we also introduce our notational conventions. In Section 3, we prove the computational complexity of the problem of verifying monotonicity for Bayesian networks in general. In Section 4, we study the use of HUGIN propagation for verifying monotonicity, in terms of its runtime and storage requirements. In Section 5, we introduce provisional propagation and the concept of lattice cover, and we investigate their use for verifying monotonicity. In Section 6, we compare the runtime requirements of verifying monotonicity with HUGIN propagation and with provisional propagation respectively. The paper ends with our conclusions and directions for further research in Section 7.

2 PRELIMINARIES

We consider a Bayesian network $\mathcal{B}$ composed of an acyclic directed graph G and an associated set of (rational number) probabilities. We assume that the network has a single variable of interest C and a non-empty set $\mathbf{E}$ of observable variables; in addition, the network may include intermediate variables which cannot be observed in practice. We assume that all variables $X_i \in \mathbf{E} \cup \{C\}$ are binary, adopting values $true$ and $false$, which are denoted as x_i and $\bar{x}_i$ respectively; the intermediate variables may be arbitrarily valued. For each observable variable E_i, we assume that its values e_i, $\bar{e}_i$ are ordered as $\bar{e}_i \leq e$; the partial ordering which is thereby induced over the set $\Omega(\mathbf{E})$ of joint value assignments to $\mathbf{E}$ is indicated as $\preceq$.

The Bayesian network $\mathcal{B}$ represents a joint probability distribution $\Pr$ over its variables. We assume that this distribution is organised over a junction tree $\mathcal{J}$ of cliques and adjoining separators; each clique in $\mathcal{J}$ has associated a potential which in essence captures the clique's marginal distribution. Several algorithms are available for computing probabilities of interest from a network. The best-known among these is the HUGIN propagation algorithm, which is based on the idea of message passing between cliques in the junction tree; in the remainder of this paper, we assume that the reader is well acquainted with the HUGIN propagation algorithm [3].

The concept of monotonicity for Bayesian networks pertains to the posterior probability distributions over the variable of interest C given all possible joint value assignments to the set $\mathbf{E}$ of observable variables [5]. A network is said to be (positively) monotone in $\mathbf{E}$ if

$$\mathbf{e} \preceq \mathbf{e}' \quad \Rightarrow \quad \Pr(c \mid \mathbf{e}) \leq \Pr(c \mid \mathbf{e}')$$

[1] Department of Information and Computing Sciences, Utrecht University, The Netherlands; email: {M.T.Rietbergen, L.C.vanderGaag, H.L.Bodlaender}@uu.nl

for all assignments $\mathbf{e}, \mathbf{e}' \in \Omega(\mathbf{E})$, that is, it is monotone if entering a higher-ordered value assignment to the observable variables cannot make the higher-ordered value of the output variable less likely; this type of monotonicity is also known as monotonicity in distribution [5]. We refer to the inequality $\Pr(c \mid \mathbf{e}) \leq \Pr(c \mid \mathbf{e}')$ as the monotonicity inequality for $\mathbf{e}$ and $\mathbf{e}'$; if this inequality does not hold for $\mathbf{e}$ and $\mathbf{e}'$, then there is a violation of monotonicity for $\mathbf{e}$ and $\mathbf{e}'$.

For studying properties of monotonicity in Bayesian networks, the concept of assignment lattice was introduced [6]. An assignment lattice $\mathcal{L}$ for a Bayesian network $\mathcal{B}$ includes all joint value assignments to the set of observable variables $\mathbf{E}$ of $\mathcal{B}$: for each value assignment $\mathbf{e}$ to $\mathbf{E}$, an element $\mathcal{L}(\mathbf{e})$ is included in the lattice. The lattice ordering encodes the partial ordering $\preceq$ on the joint value assignments to $\mathbf{E}$, that is, $\mathcal{L}(\mathbf{e})$ precedes $\mathcal{L}(\mathbf{e}')$ in the lattice if and only if $\mathbf{e} \preceq \mathbf{e}'$. If $\mathbf{e}$ and $\mathbf{e}'$ assign a different value to exactly one variable, then $\mathcal{L}(\mathbf{e})$ and $\mathcal{L}(\mathbf{e}')$ are said to be linked in $\mathcal{L}$. The bottom of the lattice encodes the lowest-ordered assignment to $\mathbf{E}$, that is, the assignment in which all observable variables have the value *false*; the top of the lattice encodes the assignment in which all variables have the value *true*. The lattice is partitioned into levels such that level λ contains the elements of those joint value assignments which assign the value *true* to exactly λ variables; level λ thus consists of $\binom{|\mathbf{E}|}{\lambda}$ elements. The width of the lattice is the maximum number of elements on a level. In the sequel, we will drop the distinction between lattice elements and joint value assignments, and simply write $\mathbf{e}$ instead of $\mathcal{L}(\mathbf{e})$.

In view of the assignment lattice, studying monotonicity of a Bayesian network amounts to investigating all pairs of conditional probabilities $\Pr(c \mid \mathbf{e})$ and $\Pr(c \mid \mathbf{e}')$ such that $\mathbf{e}$ precedes $\mathbf{e}'$ in the lattice: if for each such pair we have that the monotonicity inequality holds, then the network is monotone in $\mathbf{E}$. We note that as a result of the property of transitivity, only the monotonicity inequalities for pairs of joint value assignments that are linked in the lattice need be investigated to decide upon monotonicity [6].

3 THE COMPUTATIONAL COMPLEXITY

To establish the computational complexity of the problem of verifying monotonicity for Bayesian networks, we begin by formulating it as a decision problem.

Definition 1 *Let $\mathcal{B}$ be a Bayesian network as before, and let $\Pr$ be the joint probability distribution defined by $\mathcal{B}$. Let $\mathbf{E}$ be the set of observable variables of $\mathcal{B}$ and let C be its variable of interest. The* MONOTONICITY *problem is the problem of deciding whether for all joint value assignments $\mathbf{e}, \mathbf{e}' \in \Omega(\mathbf{E})$ to $\mathbf{E}$ with $\mathbf{e} \preceq \mathbf{e}'$, it holds that* $\Pr(c \mid \mathbf{e}) \leq \Pr(c \mid \mathbf{e}')$.

We will show that the MONOTONICITY problem in general is complete for the complexity class coNP$^{\text{PP}}$. To this end, we will study the complement of the MONOTONICITY problem, and show its intractability by a reduction from a suitable variant of the MAP problem, which we now first introduce. Let $\mathbf{o}$ be a joint value assignment to the set $\mathbf{O}$ of observable variables of a network $\mathcal{B}$, and let $p \in [0, 1]$ be a rational number. Let $\mathbf{M}$ be the set of variables of interest of $\mathcal{B}$, with $\mathbf{O} \cap \mathbf{M} = \varnothing$. The MAP problem now is the problem of deciding whether there exists a joint value assignment $\mathbf{m}$ to $\mathbf{M}$ such that $\Pr(\mathbf{mo}) > p$. This problem was shown to be complete for the complexity class NP$^{\text{PP}}$ [4]; the problem is further known to remain NP-complete for polytrees. Hardness for the class NP$^{\text{PP}}$ was shown by the construction of a Bayesian network in which the MAP variables $\mathbf{M}$ did not have any incoming arcs and had associated uniform probability distributions. In the proof therefore, only joint value assignments $\mathbf{m}$ to $\mathbf{M}$ were considered for which $\Pr(\mathbf{m})$ was the same rational number. The proof further used a single observable variable O, with the evidence o. Based upon these considerations, the problem of deciding whether there exists a joint value assignment $\mathbf{m}$ to $\mathbf{M}$ such that $\Pr(\mathbf{o} \mid \mathbf{m}) > p$, that is, the COND-MAP problem, is also readily seen to be NP$^{\text{PP}}$-complete; the COND-MAP problem moreover remains NP-complete for polytrees. We now use a reduction from this COND-MAP problem to show that the complement of the MONOTONICITY problem is NP$^{\text{PP}}$-hard.

We consider an instance of the COND-MAP problem composed of a Bayesian network $\mathcal{B}$ and a rational number $p \in [0, 1]$; in $\mathcal{B}$, we have a set $\mathbf{M} = \{V_1, \ldots, V_k\}$, $k \geq 1$, of binary MAP variables, and a single binary observable variable $O \notin \mathbf{M}$ with evidence o. From the network $\mathcal{B}$, we build a new Bayesian network $\mathcal{B}'$ as follows:

- we add k new binary variables denoted by $V_{k+1}, \ldots, V_{2k}$, to the graph G. The set $\{V_1, \ldots, V_{2k}\}$ is denoted $\mathbf{E}$, and so $\mathbf{M} \subset \mathbf{E}$. Each new variable V_i is modelled as a root of the new digraph G' of $\mathcal{B}'$, and is assigned a uniform probability distribution, that is, $\Pr(v_i) = \Pr(\overline{v}_i) = \frac{1}{2}$, $i = k+1, \ldots, 2k$.
- we further add $O(k^2)$ new variables to G, denoted by $W_{i,j}$, $i = 1, \ldots, 2k$, $j = 1, \ldots, i$. These new variables are connected to the digraph under construction through arcs $W_{i-1,j-1} \rightarrow W_{i,j}$, $W_{i-1,j} \rightarrow W_{i,j}$ and $V_i \rightarrow W_{i,j}$ for all i, j. Each variable $W_{i,j}$ is assigned a conditional probability table which conveys the following meaning: $W_{i,j} = \mathit{true}$ iff exactly j variables in $V_1, \ldots, V_i$ have the value *true*.
- two more variables $X_=$ and $X_<$ are added to the digraph:
  - the new variable $X_=$ has the variable $W_{2k,k}$ for its only parent and is assigned conditional probability distributions so as to copy its parent's value;
  - the variable $X_<$ has all variables $W_{2k,j}$ with $j < k$ for its parents[2] and is assigned a conditional probability table describing the OR of the values of its parents.

  The role of these variables will be to determine if fewer than or exactly k variables in $\mathbf{E}$ are true; we note that $X_<$ takes the value *true* iff fewer than k variables are true, and $X_=$ is *true* iff exactly k of the $2k$ variables have adopted the value *true*.
- we finally add a new evidence variable C, having the variables $X_=$, $X_<$ and the original evidence variable O from G for its parents; it is associated with conditional probability distributions such that

$$\Pr(c \mid X_=, X_<, O) = \begin{cases} 0 & \text{if } X_< \text{ is } \mathit{true} \\ 0 & \text{if } X_< \text{ is } \mathit{false}, X_= \text{ is } \mathit{true}, O \text{ is } \mathit{false} \\ 1 & \text{if } X_< \text{ is } \mathit{false}, X_= \text{ is } \mathit{true}, O \text{ is } \mathit{true} \\ p & \text{if } X_< \text{ is } \mathit{false}, X_= \text{ is } \mathit{false} \end{cases}$$

The thus constructed digraph G' is illustrated in Figure 1.

We now show that there exist $\mathbf{e}, \mathbf{e}' \in \Omega(\mathbf{E})$ such that $\mathbf{e} \preceq \mathbf{e}'$ and $\Pr(c \mid \mathbf{e}) > \Pr(c \mid \mathbf{e}')$ if and only if there exists a joint value assignment $\mathbf{m}$ to $\mathbf{M}$ with $\Pr(o \mid \mathbf{m}) > p$:

($\Leftarrow$): Suppose that there exists an $\mathbf{m}$ in $\Omega(\mathbf{M})$ with $\Pr(o \mid \mathbf{m}) > p$, and suppose that $\mathbf{m}$ assigns the value *true* to exactly ℓ variables. We now construct an $\mathbf{e} \in \Omega(\mathbf{E})$ which matches the assignment $\mathbf{m}$

[2] We assume that the set of parents is encoded by a binary tree of variables to effectively reduce the size of the conditional probability table of $X_<$.

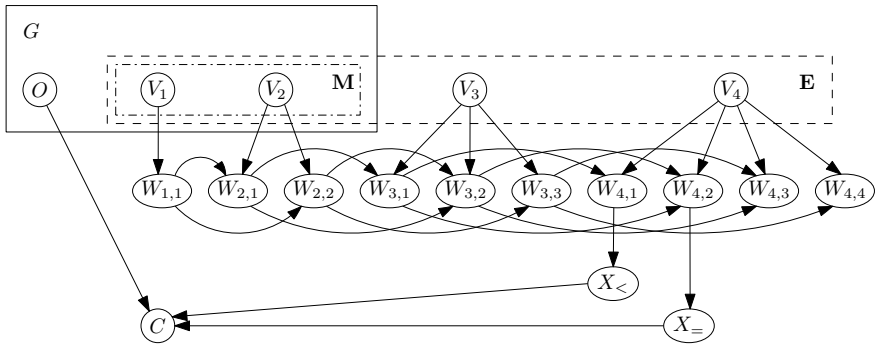

Figure 1. A sketch of the graphical structure used in the reduction of the complement of the MONOTONICITY problem from the COND-MAP problem.

on the variables $V_1, \ldots, V_k$; from among the remaining k variables $V_{k+1}, \ldots, V_{2k}$, exactly $k - \ell$ variables are assigned the value *true* and ℓ variables are set to *false*. The constructed joint value assignment $\mathbf{e}$ thus includes $\ell + (k - \ell) = k$ variables with the value *true*, that is, exactly half of the variables are *true*. By our construction, it follows that $X_=$ is true and $X_<$ is false. Since $\Pr(o \mid \mathbf{e}) = \Pr(o \mid \mathbf{m})$ and $\Pr(o \mid \mathbf{m}) > p$, we have that $\Pr(o \mid \mathbf{e}) > p$ and thus find that $\Pr(c \mid \mathbf{e}) > p$. For the joint value assignment $\mathbf{e}'$ with $\mathbf{e} \preceq \mathbf{e}'$ which assigns the value *true* to all variables in $\mathbf{E}$, we find that $X_=$ and $X_>$ are both false, and hence that $\Pr(c \mid \mathbf{e}') = p$, from which it follows that $\Pr(c \mid \mathbf{e}) > \Pr(c \mid \mathbf{e}')$ for these $\mathbf{e}, \mathbf{e}' \in \Omega(\mathbf{E})$.

($\Rightarrow$): Suppose that there exist joint value assignments $\mathbf{e}, \mathbf{e}'$ in $\Omega(\mathbf{E})$ with $\mathbf{e} \preceq \mathbf{e}'$ and $\Pr(c \mid \mathbf{e}) > \Pr(c \mid \mathbf{e}')$. Since $\mathbf{e} \preceq \mathbf{e}'$, we have that the assignment $\mathbf{e}$ includes fewer variables with the value *true* than $\mathbf{e}'$. The assignment $\mathbf{e}$ cannot have fewer than k variables with the value *true*, since we would then have, by our construction, that $\Pr(x_< \mid \mathbf{e}) = 1$ and hence $\Pr(c \mid \mathbf{e}) = 0$; we then cannot have that $\Pr(c \mid \mathbf{e}) > \Pr(c \mid \mathbf{e}')$. The assignment $\mathbf{e}$ also cannot have more than k variables with the value *true*, since we would then have, by our construction, that $X_<$ and $X_=$ are both false and hence that $\Pr(c \mid \mathbf{e}) = p$. By the same argument we would then also have that $\Pr(c \mid \mathbf{e}') = p$, which contradicts $\Pr(c \mid \mathbf{e}) > \Pr(c \mid \mathbf{e}')$. We conclude that $\mathbf{e}$ must assign *true* to exactly k variables. Since the number of variables with the value *true* in $\mathbf{e}'$ must then be larger than k, it follows that $\Pr(c \mid \mathbf{e}') = p$. For the assignment $\mathbf{e}$ we have that $X_<$ is false and $X_=$ is true, and so $\Pr(c \mid \mathbf{e}) = \Pr(o \mid \mathbf{e})$. Now, let $\mathbf{m}$ be obtained by restricting $\mathbf{e}$ to the set $\mathbf{M}$, then $\Pr(o \mid \mathbf{e}) = \Pr(o \mid \mathbf{m})$, from which it follows that $\Pr(o \mid \mathbf{m}) > \Pr(c \mid \mathbf{e}') = p$.

Based upon the computational complexity of the COND-MAP problem and the construction described above, we have that the complement of the MONOTONICITY problem is NP$^{\text{PP}}$-hard. We now prove completeness for NP$^{\text{PP}}$ by providing a polynomial certificate of membership of NP$^{\text{PP}}$ for the complement of the MONOTONICITY problem. To this end, we take a certificate that consists of a rational number $q \in [0, 1]$ and two joint value assignments $\mathbf{e}$ and $\mathbf{e}'$, with $\mathbf{e} \preceq \mathbf{e}'$, to the set of observable variables of the network under study. Whether this certificate proves the existence of a violation of monotonicity can be decided in polynomial time by determining through an INFERENCE oracle whether $\Pr(c \mid \mathbf{e}) > q$ and $q \geq \Pr(c \mid \mathbf{e}')$. We note that the rational number q need not have a higher precision than these output probabilities and can thus be represented in polynomial size with respect to the network under study [1]. Since INFERENCE is in PP, it follows that the complement of the MONOTONICITY problem is NP$^{\text{PP}}$-complete, and hence that the MONOTONICITY problem itself is coNP$^{\text{PP}}$-complete.

4 HUGIN PROPAGATION

To verify monotonicity of a Bayesian network, we must ascertain that the inequality $\Pr(c \mid \mathbf{e}) \leq \Pr(c \mid \mathbf{e}')$ holds for all joint value assignments $\mathbf{e}, \mathbf{e}'$ with $\mathbf{e} \preceq \mathbf{e}'$ to the network's set of observable variables. More specifically, verifying monotonicity amounts to checking that this inequality holds for every pair of joint value assignments which are linked in the network's assignment lattice. In this section we consider use of the HUGIN propagation algorithm for establishing all required probabilities for this purpose, and study the runtime and storage requirements involved. Throughout this section, we will often use just the term propagation to refer to HUGIN propagation.

Verifying monotonicity with* HUGIN *propagation. With HUGIN propagation, we have that after entering and propagating some evidence $\mathbf{e}$ throughout a junction tree, we can readily compute the probability $\Pr(c \mid \mathbf{e})$ from the potential of a clique containing the variable of interest. To compute the conditional probability $\Pr(c \mid \mathbf{e}')$ for some evidence $\mathbf{e}'$ with $\mathbf{e}' \neq \mathbf{e}$, the original junction tree needs in essence be restored before this alternative evidence can be entered and propagated. As verifying monotonicity requires the probability $\Pr(c \mid \mathbf{e})$ for each combination of evidence $\mathbf{e}$, using HUGIN propagation requires the propagation of each combination separately. We would like to note that for each such value combination $\mathbf{e}$, a single COLLECTEVIDENCE pass initiated from a clique containing the variable C suffices for computing the desired probability.

To forestall having to recompute conditional probabilities upon verifying monotonicity, each calculated probability $\Pr(c \mid \mathbf{e})$ must be stored until it is no longer needed, that is, until all inequalities in which it occurs have been checked. For each joint value assignment $\mathbf{e} \in \Omega(\mathbf{E})$ therefore, the probability $\Pr(c \mid \mathbf{e})$ must remain stored until it has been compared with the probabilities for all joint value assignments which are linked to $\mathbf{e}$ in the assignment lattice $\mathcal{L}$. To limit the storage requirements involved, we apply a greedy approach: once a joint value assignment $\mathbf{e}$ has been propagated and its associated conditional probability $\Pr(c \mid \mathbf{e})$ has been computed and stored, the joint value assignments linked to $\mathbf{e}$ in $\mathcal{L}$ are propagated as soon as possible; each monotonicity inequality moreover, is verified as soon as both conditional probabilities involved are available. Figure 2 provides an outline of the resulting algorithm.

Our algorithm for verifying monotonicity using HUGIN propagation starts at the bottom of the lattice and proceeds towards the top. If the joint value assignment $\mathbf{e}$ taken from the list $\boldsymbol{\rho}$ in step 1. is on level λ of the lattice therefore, the assignments added to $\boldsymbol{\rho}$ in step 2. are on level $\lambda + 1$. As a result of the first-in, first-out strategy of $\boldsymbol{\rho}$, the assignments on level λ are not propagated until all assignments on

Verifying monotonicity with HUGIN propagation

Input: a Bayesian network $\mathcal{B}$ with a variable of interest C and observable variables $\mathbf{E}$; a junction tree $\mathcal{J}$ of $\mathcal{B}$ with a clique Cl containing C; the assignment lattice $\mathcal{L}$ for $\mathcal{B}$, with the bottom $\mathbf{e}^0$.

Output: a report stating whether or not $\mathcal{B}$ is monotone in $\mathbf{E}$.

Let $\boldsymbol{\rho}$ be a first-in, first-out list of joint value assignments to $\mathbf{E}$, initialised as $\boldsymbol{\rho} = (\mathbf{e}^0)$. While $\boldsymbol{\rho} \neq \varnothing$, repeat the following steps:

1. Take the first joint value assignment $\mathbf{e}$ from $\boldsymbol{\rho}$.
2. Add to the end of $\boldsymbol{\rho}$ all unvisited assignments linked to $\mathbf{e}$ in $\mathcal{L}$.
3. Enter $\mathbf{e}$ in $\mathcal{J}$, and perform COLLECTEVIDENCE from clique Cl.
4. Compute the conditional probability $\Pr(c \mid \mathbf{e})$ from Cl.
5. For every visited assignment $\mathbf{e}'$ linked with $\mathbf{e}$ in $\mathcal{L}$, check the monotonicity inequality for $\mathbf{e}$ and $\mathbf{e}'$. If such a check fails, report that $\mathcal{B}$ is not monotone in $\mathbf{E}$.
6. Mark $\mathbf{e}$ as visited, and reinitialise the junction tree $\mathcal{J}$.

If $\boldsymbol{\rho} = \varnothing$, report that $\mathcal{B}$ is monotone in $\mathbf{E}$.

Figure 2. An algorithm for verifying monotonicity which uses HUGIN propagation for computing the required conditional probabilities.

level $\lambda - 1$ have been visited. If the assignment $\mathbf{e}$ is on level λ therefore, the inequalities checked in step 5. involve assignments on level $\lambda - 1$ only. Once all inequalities involving the assignment $\mathbf{e}$ have been checked, its computed probability is no longer required. The algorithm continues to compute probabilities through HUGIN propagation until all assignments from $\Omega(\mathbf{E})$ have been propagated or a violation has been found. As the algorithm will always find a violation of monotonicity if one exists, it correctly reports whether or not the network $\mathcal{B}$ under study is monotone in its observable variables.

Runtime and storage requirements. Computing the probability $\Pr(c \mid \mathbf{e})$ for a joint value assignment $\mathbf{e} \in \Omega(\mathbf{E})$ requires entering $\mathbf{e}$ and performing a COLLECTEVIDENCE pass through the junction tree, followed by one potential-marginalization to compute $\Pr(\mathbf{e})$ and one potential-marginalization to compute $\Pr(c, \mathbf{e})$. In a junction tree $\mathcal{J}$ with n cliques, in d of which evidence is entered, a single iteration of our algorithm thus requires $d + 3 \cdot (n-1) + 2$ potential-operations. If no violation of monotonicity is found or the first violation found involves the top of the lattice, the algorithm performs an iteration for every assignment in $\Omega(\mathbf{E})$, which is exponential in $|\mathbf{E}|$. Since the number of links in $\mathcal{L}$ is exponential in $|\mathbf{E}|$ too, the algorithm is exponential in the number of observable variables in the network under study. We expect other methods that solve the MONOTONICITY problem for arbitrary networks to be exponential in $|\mathbf{E}|$ as well.

As discussed above, each probability computed for the joint value assignments on level $\lambda - 1$ in the lattice $\mathcal{L}$ needs to be stored until the probabilities for the joint value assignments on level λ have been computed and stored. We thus need never store probabilities for more than two levels in $\mathcal{L}$, which means that the storage requirements are loosely bounded by two times the width of $\mathcal{L}$.

5 PROVISIONAL PROPAGATION

Using HUGIN propagation for verifying monotonicity has the large number of propagations involved for its main drawback. For studying the effects of entering alternative evidence, a variant of HUGIN propagation has been proposed, called cautious propagation. The basic idea of this algorithm is that the original potentials of a junction tree are not updated upon propagation and thus remain available for further computation. To this end, the tree's separators are enhanced with memory space for storing update information. Moreover, evidence is entered through dummy cliques which are connected to the junction tree by dummy separators. As a result of the availability of the original potentials and the separate update information, cautious propagation allows a larger number of probabilities to be computed from a single propagation throughout the junction tree. Although cautious propagation requires little extra storage, the additional computational cost of a single propagation when compared to HUGIN may be considerable, depending on the size and structure of the junction tree [2]. For obtaining the probabilities required for verifying monotonicity however, cautious propagation computes and stores much more information than is necessary. Based upon these considerations, we propose a new algorithm, called provisional propagation, which builds upon ideas from cautious propagation to expose multiple probabilities of interest and which meets our computational needs by more closely resembling HUGIN propagation.

Provisional propagation. While evidence is entered directly into cliques for HUGIN propagation and through dummy cliques for cautious propagation, we distinguish for provisional propagation between two types of observable variables for which the mechanisms for evidence entering differ. For the set $\mathbf{E}^S$ of observable variables that occur in separators in the junction tree, evidence is entered directly into the cliques, as for HUGIN propagation. For the set $\mathbf{E}^P = \mathbf{E} \setminus \mathbf{E}^S$, evidence is entered in so-called pockets. A pocket pertains to a subset of variables from $\mathbf{E}^P$ which are jointly contained in a single clique, and is connected to this clique. Storing just its associated finding potential, it acts like a combined dummy clique and dummy separator as used for cautious propagation.

The difference between HUGIN propagation and provisional propagation in a junction tree now lies only in the passing of messages to and from pockets. A clique Cl with a pocket P requests and receives the pocket's finding potential f_P during the COLLECTEVIDENCE pass. Each potential sent from Cl to its adjacent separators is multiplied by f_P; the clique's potential is thus never updated to incorporate the evidence entered in P. During the DISTRIBUTEEVIDENCE pass, clique Cl sends its potential to its pocket P, which marginalises this potential over all variables in $Cl \setminus P$ and stores the result as $\phi(\mathbf{e}^-, P)$, where $\mathbf{e}^-$ is the evidence propagated for all variables in $\mathbf{E} \setminus P$. Provisional propagation of additional information, such as the value c for the variable of interest C, yields an additional potential $\phi(c, \mathbf{e}^-, P)$ in P. The potentials thus stored in P allow direct access to probabilities not only for the assignment propagated for the variables in P, but also for alternative evidence for P. A single provisional propagation thus makes multiple probabilities of interest available, while requiring at most three potential-operations per pocket more than HUGIN propagation. Technical details on provisional propagation will be provided in a forthcoming journal paper.

Verifying monotonicity with provisional propagation. After provisionally entering and propagating the evidence $\mathbf{e}$ to the set $\mathbf{E}$ of observable variables, and subsequent propagation of the value c for the variable of interest, each pocket P in the junction tree contains the potentials $\phi(c, \mathbf{e}^-, P)$ and $\phi(\mathbf{e}^-, P)$. Division of these potentials readily yields the conditional probability $\Pr(c \mid \mathbf{e}^-, \mathbf{e}_P)$ for each assignment $\mathbf{e}_P$ to the variables in P. To compute the conditional probabilities necessary for verifying monotonicity, it is thus not necessary to separately propagate all joint value assignments to the observable variables. To decide which subset of assignments is best propagated to minimise the total number of propagations involved, we build upon the concept of lattice cover.

Given a set $\mathbf{E} = \mathbf{E}^S \cup \mathbf{E}^P$ of observable variables, a lattice cover $\boldsymbol{\omega}$ of the assignment lattice $\mathcal{L}$ of $\mathbf{E}$ is a set of joint value assignments such that each joint value assignment $\mathbf{e}$ to $\mathbf{E}$ is either itself contained in $\boldsymbol{\omega}$ or linked in $\mathcal{L}$ to an $\mathbf{e}' \in \boldsymbol{\omega}$ which differs from $\mathbf{e}$ in the value assigned to exactly one variable of $\mathbf{E}^P$; the proportion $\boldsymbol{\omega}/|\Omega(\mathbf{E})|$ is referred to as the cover's degree. To construct such a cover $\boldsymbol{\omega}$, we first obtain a perfect cover $\boldsymbol{\omega}^\star$ of the lattice $\mathcal{L}^\star$ for a subset $\mathbf{E}^\star$ of $\mathbf{E}^P$, that is, a cover such that for all $\mathbf{e}^\star \in \Omega(\mathbf{E}^\star)$, either $\mathbf{e}^\star \in \boldsymbol{\omega}^\star$ or $\mathbf{e}^\star$ is linked in $\mathcal{L}^\star$ to exactly one $\mathbf{e}' \in \boldsymbol{\omega}^\star$; this perfect cover is readily obtained using the property that all its assignments must differ in the value assigned to at least three variables. We derive from the constructed perfect cover $\boldsymbol{\omega}^\star$ a lattice cover $\boldsymbol{\omega}$ of $\mathcal{L}$ by concatenating each assignment $\mathbf{e}^\star \in \boldsymbol{\omega}^\star$ with each assignment $\mathbf{e}^-$ to the variables in $\mathbf{E} \setminus \mathbf{E}^\star$; note that the degree of $\boldsymbol{\omega}$ is equal to the degree of $\boldsymbol{\omega}^\star$. It is readily seen that as a perfect cover exists only for $|\mathbf{E}^\star| = 2^k - 1$ with $k \in \mathbb{N}_{>0}$, the perfect cover $\boldsymbol{\omega}^\star$ and its derived cover $\boldsymbol{\omega}$ both have degree $\frac{1}{2^k}$. To minimize the degree of the cover to be used, we take $\mathbf{E}^\star$ to be the largest possible subset of $2^k - 1$ variables from $\mathbf{E}^P$, and use the phrase extended cover to refer to the cover $\boldsymbol{\omega}$ derived from a perfect cover of its lattice $\mathcal{L}^\star$.

To verify monotonicity of a Bayesian network using provisional propagation, we provisionally propagate only the assignments of an extended cover $\boldsymbol{\omega}$ of its lattice. The probability given a joint value assignment $\mathbf{e}$ is obtained either from propagation of $\mathbf{e}$ itself or from $\mathbf{e}$ being linked to some propagated $\mathbf{e}' \in \boldsymbol{\omega}$. For each assignment $\mathbf{e} \in \Omega(\mathbf{E})$, the computed probability $\Pr(c \mid \mathbf{e})$ needs to be stored until all inequalities in which it occurs have been checked, that is, until probabilities have been obtained for all joint value assignments linked to $\mathbf{e}$ in the assignment lattice $\mathcal{L}$. To limit the storage requirements involved, the assignments of $\boldsymbol{\omega}$ are propagated in order of increasing level number in the lattice, similar to the order used for HUGIN propagation; moreover, each monotonicity inequality is checked as soon as both probabilities involved are available. The resulting algorithm is outlined in Figure 3.

Our algorithm for verifying monotonicity using provisional propagation starts with the bottom assignment $\mathbf{e}^0$ of the lattice $\mathcal{L}$. In each iteration of the algorithm, if the joint value assignment $\mathbf{e}$ taken from $\boldsymbol{\rho}$ in step 1. is on level λ, then the joint value assignments included in $\boldsymbol{\mu}$ in step 3. are on levels $\lambda - 1$ to $\lambda + 1$. Since the joint value assignments of $\boldsymbol{\omega}$ are provisionally propagated in order of increasing level number, an assignment on level λ is not propagated until all assignments on level $\lambda - 2$ have been visited. The inequalities to be checked in step 4(c) can thus involve assignments on the levels $\lambda - 2$ to $\lambda + 1$ only. Once all inequalities involving an assignment have been checked, the probability for this assignment need no longer be stored. The algorithm continues to obtain probabilities through provisional propagation until all assignments in the extended cover $\boldsymbol{\omega}$ have been propagated or a violation has been found. The algorithm correctly reports whether or not the network $\mathcal{B}$ under study is monotone in its observable variables, since it will always find a violation of monotonicity if one exists.

Runtime and storage requirements. In each iteration of our algorithm, a single joint value assignment $\mathbf{e} \in \boldsymbol{\omega}$ is provisionally entered and propagated in the junction tree; the value c for the variable of interest C is subsequently propagated through a single DISTRIBUTEEVIDENCE pass. For all $\mathbf{e}'$ on the list $\boldsymbol{\mu}$ for $\mathbf{e}$, the conditional probabilities $\Pr(c \mid \mathbf{e}')$ are obtained through potential-divisions in the pockets of $\mathcal{J}$. In a junction tree $\mathcal{J}$ with n cliques, in d_2 of which evidence is entered directly, and d_1 pockets, a single iteration of the algorithm thus requires $d_2 + 9 \cdot (n - 1) + 6 \cdot d_1 + 1$ potential-operations. If no violation of monotonicity is found or if the violation found involves an assignment obtained through propagation of the last assignment on the sorted list $\boldsymbol{\rho}$ for $\boldsymbol{\omega}$, an iteration is performed for every joint value assignment in the lattice cover $\boldsymbol{\omega}$.

Verifying monotonicity with provisional propagation

Input: a Bayesian network $\mathcal{B}$ with C and $\mathbf{E}$ as before; a junction tree $\mathcal{J}$ of $\mathcal{B}$ with a clique Cl in $\mathcal{J}$ containing C; a subset $\mathbf{E}^\star \subseteq \mathbf{E}$ of $2^k - 1$ observable variables in pockets of $\mathcal{J}$; the assignment lattice $\mathcal{L}$ of $\mathbf{E}$, with the bottom $\mathbf{e}^0$; an extended cover $\boldsymbol{\omega}$ derived from the perfect cover $\boldsymbol{\omega}^\star$ for $\mathbf{E}^\star$.

Output: a report stating whether or not $\mathcal{B}$ is monotone in $\mathbf{E}$.

Let $\boldsymbol{\rho}$ be a list of all the joint value assignments of $\boldsymbol{\omega}$ ordered by increasing level number in $\mathcal{L}$. While $\boldsymbol{\rho} \neq \varnothing$, repeat the following steps:

1. Take the first joint value assignment $\mathbf{e}$ from $\boldsymbol{\rho}$, enter $\mathbf{e}$ in $\mathcal{J}$ as evidence for $\mathbf{E}$, and perform provisional propagation on $\mathcal{J}$.
2. Propagate c in $\mathcal{J}$ through provisional DISTRIBUTEEVIDENCE from clique Cl.
3. Let $\boldsymbol{\mu}$ be a list, ordered by increasing level number in $\mathcal{L}$, of unvisited joint value assignments which differ from $\mathbf{e}$ in the value assigned to at most one variable in $\mathbf{E}^\star$.
4. While $\boldsymbol{\mu} \neq \varnothing$, repeat the following steps:
   (a) Take the first joint value assignment $\mathbf{e}'$ from $\boldsymbol{\mu}$.
   (b) Obtain the conditional probability $\Pr(c \mid \mathbf{e}')$ from $\mathcal{J}$, and mark $\mathbf{e}'$ as visited.
   (c) Check the monotonicity inequalities for $\mathbf{e}'$ and the visited assignments linked with $\mathbf{e}'$ in $\mathcal{L}$, in order of increasing level number. If such a check fails, report that $\mathcal{B}$ is not monotone in $\mathbf{E}$.
5. Reinitialise the junction tree $\mathcal{J}$.

If $\boldsymbol{\rho} = \varnothing$, report that $\mathcal{B}$ is monotone in $\mathbf{E}$.

Figure 3. An algorithm for verifying monotonicity which uses provisional propagation for computing the required conditional probabilities.

As discussed above and as illustrated in Figure 4, the conditional probabilities obtained for the joint value assignments on level $\lambda - 2$ must be stored, at the longest, until the iterations for the assignments of $\boldsymbol{\omega}$ on level λ have been completed. Moreover, the highest level on which these iterations may store probabilities for assignments is level $\lambda + 1$. We thus need never store probabilities for more than four levels in the lattice $\mathcal{L}$, which means that the storage requirements are loosely bounded by four times the width of $\mathcal{L}$.

6 THE ALGORITHMS COMPARED

Let $\mathcal{B}$ be a Bayesian network with C and $\mathbf{E} = \mathbf{E}^S \cup \mathbf{E}^P$ as before, and let $\mathcal{J}$ be a junction tree representation of $\mathcal{B}$ with n cliques. For HUGIN propagation, evidence is entered directly into d of these n cliques, while evidence is provisionally entered into d_1 pockets and directly into d_2 cliques. Monotonicity of $\mathcal{B}$ in $\mathbf{E}$ can be verified with HUGIN propagation or with provisional propagation using a cover $\boldsymbol{\omega}$ of the lattice $\mathcal{L}$ of $\mathcal{B}$. With HUGIN propagation each iteration takes $d + 3 \cdot (n - 1) + 2$ potential-operations, and up to $|\Omega(\mathbf{E})|$ iterations may be required. With provisional propagation each iteration takes $d_2 + 9 \cdot (n-1) + 6 \cdot d_1 + 1$ potential-operations, and up to $|\boldsymbol{\omega}|$ iterations may be required. Using provisional propagation is faster than using HUGIN propagation only if we have a lattice cover $\boldsymbol{\omega}$ such that

$$\frac{|\boldsymbol{\omega}|}{|\Omega(\mathbf{E})|} < \frac{d + 3 \cdot (n - 1) + 2}{d_2 + 9 \cdot (n - 1) + 6 \cdot d_1 + 1} = \Delta,$$

that is, if we have a cover $\boldsymbol{\omega}$ with a degree smaller than Δ. The factor Δ thus constitutes an upper bound on a cover's degree be-

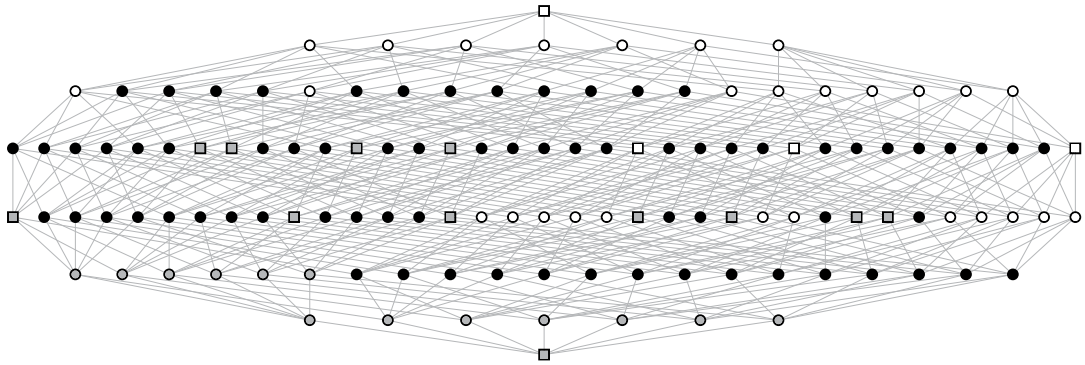

Figure 4. Representation of the assignment lattice for 7 variables and its perfect cover (squares), showing the state of the assignments at the start of the 13th iteration of the algorithm: unvisited (white), visited and currently storing a probability (black), and visited but no longer storing a probability (gray).

low which provisional propagation is guaranteed to execute fewer potential-operations than HUGIN propagation. Since the factor Δ depends on the constants n, d, d_1 and d_2, it may vary considerably among real-world Bayesian networks.

Despite the dependence on the constants involved, we can make some general comparative statements about the runtime characteristics of the two algorithms. We make the following assumptions about the relationship between the constants d, d_1 and d_2 pertaining to evidence entering in the junction tree $\mathcal{J}$ at hand:

- $1 \leq d_1 \leq d$, that is, $\mathcal{J}$ has no more than d pockets for provisional evidence entering and at least one pocket to ensure that $\mathbf{E}^P \neq \varnothing$;
- $0 \leq d_2 \leq d$, that is, provisional evidence entering is done directly into at most d cliques in $\mathcal{J}$;
- $d_1 + d_2 \geq d$, that is, every clique in $\mathcal{J}$ in which evidence is entered for HUGIN propagation also receives evidence for provisional propagation, either through a pocket or directly or both.

With fixed n, we now find that the factor Δ takes its maximum value for $d = n$, $d_1 = 1$, $d_2 = n - 1$, and its minimum value for $d = d_1 = d_2 = n$. Figure 5 illustrates these maximum and minimum values for junction trees with 2 to 20 cliques. For a cover with a degree in between the two bounds, it depends on the precise values of d, d_1 and d_2 which of the two algorithms has the better runtime. If the cover's degree is smaller than the smallest possible value of Δ, using provisional propagation for verifying monotonicity is certainly faster than using HUGIN propagation; using HUGIN is faster if the degree is larger than the largest possible value of Δ. If at least 3 variables are contained in the pockets of the junction tree, however, we will have a cover of degree at most $\frac{1}{4}$, which is already slightly below the smallest value of Δ. From the analysis above, we conclude that when for verifying monotonicity probabilities must be obtained for all combinations of evidence and all monotonicity inequalities must be checked, using provisional propagation is expected to have better runtime performance than using HUGIN propagation, given a cover of sufficiently small degree. If a network under study is not monotone in its observable variables however, both algorithms will find a violation and terminate. As the algorithms check the monotonicity inequalities in different orders, termination may be caused by different violations. Consequently, either algorithm may terminate first.

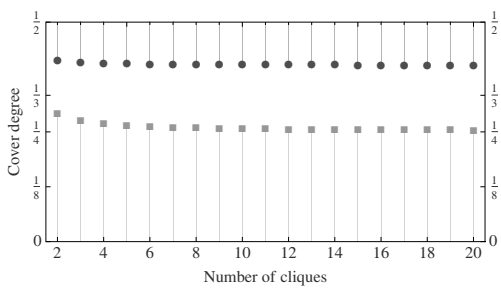

Figure 5. Bounds on the cover degree above which using HUGIN propagation is faster (circles) and below which using provisional propagation is faster (squares), depending on the number of cliques in the junction tree.

7 CONCLUSIONS

We studied the computational complexity of the MONOTONICITY problem of verifying whether a given Bayesian network is monotone in its observable variables, and studied two propagation algorithms for solving this problem exactly by direct computation of all required probabilities. With both HUGIN propagation and provisional propagation, the structure of the assignment lattice for a network is used to guide the search for violations of monotonicity. For provisional propagation moreover, the structure of the lattice and its cover are exploited to reduce the runtime involved, thereby increasing the feasibility of its use in practice. We hope to report in the near future the results of verifying monotonicity with both propagation algorithms for real-world networks. We will further study the application of provisional propagation for verifying other types of monotonicity and for solving related computational problems for Bayesian networks.

REFERENCES

[1] H.L. Bodlaender, F. van den Eijkhof, L.C. van der Gaag, 'On the complexity of the MPA problem in probabilistic networks', in *Proceedings of the 15th European Conference on Artificial Intelligence*, ed., F. van Harmelen, 675–679. IOS Press, (2002).

[2] F.V. Jensen, 'Cautious propagation in Bayesian networks', in *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, eds., P. Besnard, S. Hanks, 323–328. Morgan Kaufmann, (1995).

[3] F.V. Jensen, S.L. Lauritzen, K.G. Olesen, 'Bayesian updating in causal probabilistic networks by local computations', *Computational Statistics Quarterly*, **4**, 269–282, (1990).

[4] J.D. Park, 'MAP complexity results and approximation methods', in *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, eds., A. Darwiche, N. Friedman, 388–396. Morgan Kaufmann, (2002).

[5] L.C. van der Gaag, H.L. Bodlaender, A. Feelders, 'Monotonicity in Bayesian networks', in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, eds., M. Chickering, J. Halpern, 569–576. AUAI Press, (2004).

[6] L.C. van der Gaag, S. Renooij, P.L. Geenen, 'Lattices for studying monotonicity in Bayesian networks', in *Proceeding of the Third European Workshop on Probabilistic Graphical Models*, eds., M. Studený, J. Vomlel, 99–106. Prague, (2006).