Complexity Results for Local Monotonicity in Probabilistic Networks

Johan Kwisthout

Hans L. Bodlaender

Gerard Tel

Department of Information and Computing Sciences, Utrecht University

Technical Report UU-CS-2007-050

www.cs.uu.nl

ISSN: 0924-3275

Complexity Results for Local Monotonicity in Probabilistic Networks

Johan Kwisthout, Hans L. Bodlaender, and Gerard Tel
Department of Information and Computer Sciences, University of Utrecht,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands.
email: `johank@cs.uu.nl`, `hansb@cs.uu.nl`, `gerard@cs.uu.nl`

March 13, 2008

Abstract

Often, monotonicity is a desirable property of probabilistic networks. For example, when medical knowledge in a particular domain dictates that more severe symptoms increase the likeliness of a more serious disease, these properties should be reflected in the network. Unfortunately, the problem to determine for a given probabilistic network whether it is monotone is known to be a highly intractable problem. Often, approximation algorithms are employed that work on a local scale. These algorithms determine the monotonicity of the arcs, rather than the network as a whole. However, whether an arc is monotone may depend on the ordering of the values of the variables that it uses. Sometimes, the choice of such an ordering is rather arbitrary. In these cases, it is desirable to order the values of these variables such that all arcs (or as many arcs as possible) are monotone. In this paper we discuss the concept of local monotonicity and its computational complexity. We present an algorithm for determining whether there exists an ordering of the values of the variables such that all arcs in a network are monotone, and show that this can be done in time, exponential only in the treewidth of the network. On the other hand, optimizing the number of monotone arcs is NP-complete and hard to approximate as well. We sketch a branch-and-bound exact algorithm to find an optimal solution for this problem.

1 Introduction

In many probabilistic networks [Pea88] that are used for classification in real problem domains, the variables of the network can be distinguished into observable input variables, non-observable intermediate variables and a single output variable. For example, in a medical domain the observable variables represent clinical evidence such as symptoms and test results, the output variable functions as a classification of a disease, and the intermediate variables model non-observable variables that are relevant for classification. The relations between observable symptoms and the classification variable are often monotone, e.g., higher values for the observable variable 'fever' makes higher values of the classification variable 'flu' more likely, independent of the value of other variables such as 'headache'. Such a network is *monotone in distribution* [vdGBF04] if higher-ordered configurations of the observable variables make higher-ordered outputs more (isotone) or less (antitone) likely.

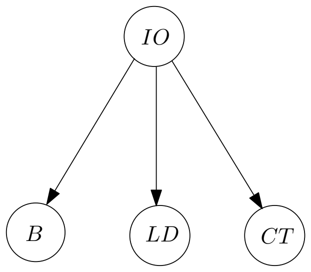


Figure 1: Small part of the Oesophagus network

When a domain expert indicates that a certain relation ought to be monotone, the joint probability distribution should be such that this property is reflected in the network. If monotonicity is violated, the probability distribution in the network can be revised in cooperation with the expert. Unfortunately, determining whether a network is monotone in distribution is, in general, highly intractable ([vdGBF04]). One approach to overcome this unfavorable complexity, is by approximating the decision (i.e, sometimes have 'undecidable' as outcome) like the algorithm discussed in [vdGBF04]. This algorithm uses qualitative influences (see e.g. [Wel90] for an introduction in qualitative networks or QPNs) that summarize the direction of the influence of variables by signs. However, the use of these signs of course requires an ordering on the values of the variables under consideration. Such an ordering might be implicit, for example *large* > *medium* > *small* or *true* > *false*. But in practice, there are often variables in a network which do not have such 'natural' orderings. As it is desirable to have as many as possible monotone influences (to minimize the offending context), it is important to *choose* an ordering for the values of these variables that maximizes the number of monotone arcs. Or, equivalently, minimizes the number of '?' signs in the corresponding QPN.

In Figure 1, a small part of the so-called Oesophageal Cancer network [vdGRW$^+$02] is shown. In this network, clinical evidence is used to determine the development stage of the tumor, and, as a consequence, the medical treatment. The network excerpt relates clinical evidence to a possible invasion of the tumor from the oesophageal wall into neighbouring organs. Laparoscopic evidence of an invasion of the tumor to the diaphragm, bronchoscopic evidence of an invasion of the bronchi, and evidence of a CT-scan influences the probability of an invasion of the trachea, mediastinum, diaphragm, heart, or none of these. The conditional probabilities are shown in Table 1.

The values of the nodes 'Invasion-Organs' (IO) and 'CT-organs' (CT) do not have an "natural" ordering. Nevertheless, when determining monotonicity in this part of the network such an ordering is highly relevant. For example, if we order the values of 'Invasion-Organs' and 'CT-organs' as *none* > *mediastinum* > *trachea* > *diaphragm* > *heart* then higher values for 'CT-Organs' predict lower values for 'Invasion-organs', but the other relations violate monotonicity. When we order the values for 'Invasion-Organs' as *diaphragm* > *none* > *mediastinum* > *heart* > *trachea*, then 'Bronchoscopy' (B) and 'Lapa-Diaphragm' (LD) become monotone, with respect to 'Invasion-organs', but we violate monotonicity with 'CT-organs'.

In this paper, we propose algorithms for deciding whether a network is locally monotone, and, if not, optimize the number of monotone arcs. We show that the decision variant of the latter problem is NP-complete and APX-hard as well. In Section 2, we introduce some notations and definitions. We discuss our algorithm for deciding whether a network

	CT-organs				
Invasion-Organs	none	trachea	mediastinum	diaphragm	heart
none	0.8	0.05	0.05	0.05	0.05
trachea	0.05	0.58	0.22	0.05	0.1
mediastinum	0.3	0.1	0.4	0.1	0.1
diaphragm	0.05	0.1	0.22	0.53	0.1
heart	0.05	0.1	0.22	0.1	0.53

	Bronchoscopy	
Invasion-Organs	yes	no
none	0.04	0.96
trachea	0.92	0.08
mediastinum	0.1	0.9
diaphragm	0.01	0.99
heart	0.25	0.75

	Lapa-Diaphragm	
Invasion-Organs	yes	no
none	0.02	0.98
trachea	0.02	0.98
mediastinum	0.02	0.98
diaphragm	0.8	0.2
heart	0.02	0.98

Table 1: Conditional probability tables for part of the Oesophagus network

is locally monotone in Section 3, and prove NP-completeness of the decision variants of the optimization problems in Section 4. In Section 5, we show that the optimization problems are hard to approximate as well, and we suggest a branch-and-bound strategy as an exact algorithm in Section 6. Finally, we conclude our paper in Section 7.

2 Preliminaries

Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a Bayesian network where $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ is an acyclic directed graph, and Γ, the set of conditional probability distributions, is composed of rational probabilities. Let Pr be the joint probability distribution of $\mathcal{B}$. The conditional probability distributions in Γ are assumed to be explicit, i.e., represented with look-up tables. For any variable $X \in \mathbf{V}$, let $\Omega(X)$ denote the set of values that X can take. A node Y is denoted as a predecessor of X if $(Y, X) \in \mathbf{A}$. The set of all predecessors of X is denoted as $\pi(X)$, the set of all children of X is denoted as $\sigma(X)$. If, for a node X, $\pi(X)$ is the set $\{Y_1, \ldots, Y_n\}$, the *configuration template* $\mathbf{Y}$ is defined as $\Omega(Y_1) \times \ldots \times \Omega(Y_n)$; a particular instantiation $\mathbf{y}$ of $Y_1, \ldots, Y_n$ will be denoted as a *configuration* of $\mathbf{Y}$.

2.1 Local monotonicity

Monotonicity can be defined as stochastical dominance (monotone in distribution) or in a modal sense (monotone in mode). In this paper, we discuss monotonicity in distribution only, and we focus on local effects, i.e., influences between two variables which are directly connected. A network is *locally monotone* if all qualitative influences along the arcs in the network are either positive or negative.

Definition 1 (local monotonicity). Let F be the cumulative distribution function for a node $X \in \mathbf{V}$, defined by $F(x) = Pr(X \leq x)$ for all $x \in \Omega(X)$. For any arc $(X, Y) \in \mathbf{A}$, let $\mathbf{Z}$ denote the configuration template $\pi(Y) \setminus X$, and let $\mathbf{z}$ denote an individual configuration of $\mathbf{Z}$. With (X, Y), a *positive* influence is associated if $x < x' \rightarrow F(y \,|\, x\mathbf{z}) \geq F(y \,|\, x'\mathbf{z})$ for all $y \in \Omega(Y), x, x' \in \Omega(X)$, and $\mathbf{z} \in \mathbf{Z}$. Similarly, a *negative* influence is associated with this arc if $x < x' \rightarrow F(y \,|\, x\mathbf{z}) \leq F(y \,|\, x'\mathbf{z})$ for all $y \in \Omega(Y), x, x' \in \Omega(X)$, and $\mathbf{z} \in \mathbf{Z}$. We will denote an arc associated with an positive or negative influence as a *isotone*, respectively *antitone* arc. $\mathcal{B} = (G, \Gamma)$ is *locally monotone* if all arcs in $\mathbf{A}$ are either isotone or antitone.

2.2 Interpretations

The above notions of monotonicity assumed an implicit *ordering* on the values of the variables involved. Such an ordering is often trivial (e.g., $x > \bar{x}$ and *always* > *sometimes* > *never*) but sometimes it is arbitrary, like an ordering of the values { *trachea*, *mediastinum*, *diaphragm*, *heart* }. Nevertheless, a certain ordering is necessary to determine whether the network is monotone, or to determine which parts of the network are violating monotonicity assumptions. Thus, for nodes where no *a priori* ordering is given, we want to order the values of these nodes in a way that maximizes the number of monotone arcs or the number of nodes with only monotone incoming arcs (depending on the specific application).

We define the notion of an *interpretation* of X to denote a certain ordering on $\Omega(X)$, the set of values of X. Note, that the number of distinct interpretations of a node with k values equals $k!$, the number of permutations of these values. Nevertheless, in practice, the number of values a variable can take is often small. For example, in the Alarm network [BSCC89], the number of values is at most four, and in the Oesophageal network [vdGRW$^+$02] it is at most six. In this paper, we assume that k is small and can be regarded as a fixed constant.

Definition 2 (interpretation). An *interpretation* of $X \in \mathbf{V}$, denoted I_X, is a total ordering on $\Omega(X)$. For arbitrary interpretations we will often use σ and τ. We use the superscript T to denote a reverse ordering: if $\sigma = (x_1 < x_2 < \ldots < x_n)$, then $\sigma^T = (x_n < \ldots < x_2 < x_1)$. The *interpretation set* $\mathbf{I_X}$ is defined as the set of all possible interpretations of X. Note that an arc is isotone for a given interpretation σ if and only if it is antitone for σ^T and vice versa, and that the interpretations in $\mathbf{I_X}$ are pairwise symmetric. In the remainder, when $\sigma, \tau \in \mathbf{I_X}$ are assumed distinct, then we also assume that $\sigma \neq \tau^T$. We use the shorthand notation $x_1 <_\sigma x_2$ to denote that $x_1 < x_2$ under interpretation σ.

2.3 Monotonicity functions and schemes

We define a *monotonicity function*, which determines whether a certain combination of interpretations for the two nodes of an arc makes the arc isotone or antitone. When a node has more than one predecessor (say $\pi(Y) = \{X_1, X_2\}$), the arc (X_1, Y) is monotone for a certain combination of interpretations $\sigma \in \mathbf{I}_{X_1}$ and $\tau \in \mathbf{I}_Y$, when it is isotone for all values[1] of X_2, or when it is antitone for all values of X_2. We define the monotonicity function of (X_1, Y) for a particular *given* value $x_2 \in \Omega(X_2)$ as a *partial monotonicity function*, to emphasize the *conditional* monotonicity of (X_1, Y).

Definition 3 ((partial) monotonicity function). Consider the arc $x_1 = (X_1, Y) \in \mathbf{A}$, where Y has auxiliary predecessors (say $x_2 \ldots x_n$), whose configuration template we denote with $\mathbf{Z}_N$. Assume $\sigma \in \mathbf{I}_{X_1}$ and $\tau \in \mathbf{I}_Y$. Then $M_{X_1Y}(\sigma, \tau)$ is true if and only if x_1 is either

[1]Note that the *ordering* of the elements in $\Omega(X_2)$ is irrelevant for the local monotonicity of (X_1, Y).

isotone (denoted $M^{+}_{X_1Y}$) or antitone (denoted $M^{-}_{X_1Y}$) for interpretations σ and τ, for all possible configurations of $\mathbf{Z}_N$. The *partial monotonicity function* $M_{X_1Y}(\sigma, \tau \,|\, \mathbf{z}_N)$ is true if and only if x_1 is isotone or antitone for interpretations σ and τ, given a specific configuration $\mathbf{z}_N$ of $\mathbf{Z}_N$.

Observe, that $M_{XY}(\sigma,\tau) = M_{XY}(\sigma^T,\tau) = M_{XY}(\sigma,\tau^T) = M_{XY}(\sigma^T,\tau^T)$ since $M_{XY} = M^{+}_{XY} \vee M^{-}_{XY}$, and $M^{+}_{XY}(\sigma,\tau) \leftrightarrow M^{-}_{XY}(\sigma^T,\tau)$. Partial monotonicity functions and schemes can be combined for multiple configurations of $\mathbf{Z}_N$. Informally, the combined partial monotonicity function for instantiation z_ϕ and z_ψ is true for a certain combination of interpretations, if the individual partial monotonicity functions are all isotone, or all antitone, for that combination.

Definition 4 (combining partial monotonicity functions). Consider again the arc x_1 as defined before, with $\mathbf{Z}_N$ as the configuration template of $\pi(Y) \setminus X_1$. Then, for $\delta \in \{+,-\}$,

$$M^{\delta}_{X_1Y}(\sigma,\tau \,|\, \mathbf{z}_\phi) \wedge M^{\delta}_{X_1Y}(\sigma,\tau \,|\, \mathbf{z}_\psi) = M^{\delta}_{X_1Y}(\sigma,\tau \,|\, \mathbf{z}_\phi, \mathbf{z}_\psi)$$

and consequently,

$$\bigwedge_{z_N \in Z_N} M^{\delta}_{X_1Y}(\sigma,\tau \,|\, \mathbf{z}_N) = M^{\delta}_{x_1Y}(\sigma,\tau)$$

With every monotonicity function M_{XY}, a binary matrix $\mathbf{M_{XY}}$ is associated, denoted as the *monotonicity scheme* of M_{XY}. Similarly, a *partial monotonicity scheme* $\mathbf{M_{XY|z_N}}$ is associated with the corresponding partial monotonicity function. These matrices have dimensions $\frac{1}{2} \mid \mathbf{I}_X \mid \times \frac{1}{2} \mid \mathbf{I}_Y \mid$, since the interpretations in $\mathbf{I}$ are pairwise symmetric. We will often illustrate these matrices using a grid, where shaded areas denote monotone combinations of interpretations in $\mathbf{I}_X$ and $\mathbf{I}_Y$. If the monotonicity scheme $\mathbf{M_{XY}}$ *factorizes* over values of $\mathbf{I}_X$ and $\mathbf{I}_Y$ than $\mathbf{M_{XY}}$ will be denoted as a *factorizing monotonicity scheme.* An arc (X, Y) with such a factorizing monotonicity scheme will be denoted as a *factorizing arc.*

Definition 5 (factorizing monotonicity scheme). $\mathbf{M_{XY}}$ is called *factorizing* over $\mathbf{I}_X$ and $\mathbf{I}_Y$ if there exist subsets $\mathbf{I^+_X} \subseteq \mathbf{I_X}$ and $\mathbf{I^+_Y} \subseteq \mathbf{I_Y}$ such that $M_{XY}(\sigma,\tau)$ is true if and only if $\sigma \in \mathbf{I^+_X}$ and $\tau \in \mathbf{I^+_Y}$.

2.4 Properties of monotonicity schemes

The definition of monotonicity implies that certain monotonicity schemes are impossible, thus restricting the number of possible interpretations in the network. More in particular, if we look at the columns of a monotonicity scheme $\mathbf{M_{XY}}$, where each column represents an interpretation $\sigma \in \mathbf{I}_X$, we will see that two columns are either equal or disjoint. That is, either $M_{XY}(\sigma,\tau) = M_{XY}(\sigma',\tau)$ for all $\tau \in \mathbf{I}_Y$, or $M_{XY}(\sigma,\tau) \to \neg M_{XY}(\sigma',\tau)$ for all $\tau \in \mathbf{I}_Y$. This property can be used to speed up the construction of a monotonicity scheme.

First observe the following: if an arc $X \to Y$ is monotone for two distinct interpretations $\sigma, \sigma' \in \mathbf{I}_X$ given an interpretation $\tau \in \mathbf{I}_Y$, then there are at least two equal columns in the joint probability table, i.e., $\Pr(y_k \,|\, x_i) = \Pr(y_k \,|\, x_j)$ for all $y_k \in \Omega(Y)$. But then, $M_{XY}(\sigma,\tau) = M_{XY}(\sigma',\tau)$ for *all* interpretations $\tau \in \mathbf{I}_Y$. Of course, in two distinct interpretations[2] there exist i and j such that $x_i \leq x_j$ in one interpretation and $x_j \leq x_i$ in the other.

Lemma 6. *Assume $\sigma, \sigma' \in \mathbf{I}_X$ are distinct, and $\tau \in \mathbf{I}_Y$. Then, if $M_{XY}(\sigma,\tau) \wedge M_{XY}(\sigma',\tau)$, then there exist $x_i, x_j \in \Omega(X)$ such that $Pr(y_k \,|\, x_i) = Pr(y_k \,|\, x_j)$ for all $y_k \in \Omega(Y)$.*

[2]Note that we defined σ and σ' to be distinct only if also $\sigma' \neq \sigma^T$.

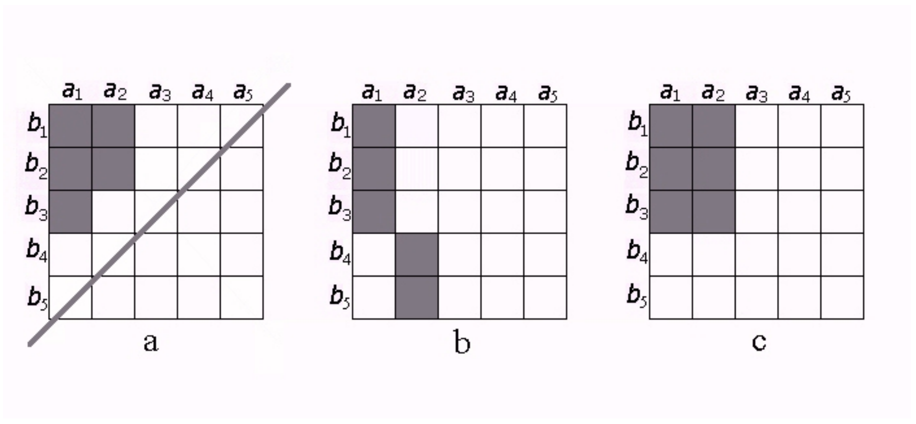


Figure 2: a is impossible: columns are either disjoint (b) or equal (c)

Proof of Lemma 6. Since $M_{XY}(\sigma, \tau) = M_{XY}(\sigma^T, \tau) = M_{XY}(\sigma, \tau^T) = M_{XY}(\sigma^T, \tau^T)$ we can assume, without loss of generality, that $M_{XY}(\sigma, \tau)$ and $M_{XY}(\sigma', \tau)$ both denote isotone relations. Since σ and σ' are distinct, there exist i and j such that for $x_i, x_j \in \Omega(X)$ $x_i <_\sigma x_j$ and $x_j <_{\sigma'} x_i$. Furthermore, there exist partial orderings $\rho, \upsilon_1, \upsilon_2$ with respect to $\Omega(X)$ such that $\sigma = \rho x_i \upsilon_1$ and $\sigma' = \rho x_j \upsilon_2$. For the cumulative distribution function $F(Y \mid X)$ it holds that $F(Y \mid x_i) \leq F(Y \mid x_j)$, but also that $F(Y \mid x_j) \leq F(Y \mid x_i)$. But then $\Pr(y_k \mid x_i) = \Pr(y_k \mid x_j)$ for all $y_k \in \Omega(B)$. □

From this lemma, it follows as a corollary that monotonicity functions are symmetric: The corresponding monotonicity scheme can be described by a disjunct set of factorizing sub-schemes, and the constraints describing these sub-schemes are symmetric. This symmetry property is formalised in corollary 7.

Corollary 7 (Symmetry of monotonicity schemes). *Assume* $\sigma, \sigma' \in \mathbf{I_X}, \tau,\ \tau' \in \mathbf{I_Y}$. *Then,* $M_{XY}(\sigma, \tau) \wedge M_{XY}(\sigma', \tau) \wedge M_{XY}(\sigma, \tau') \Rightarrow M_{XY}(\sigma', \tau')$

3 Local Monotonicity

The first problem we discuss is the problem of determining whether a network can be made locally monotone. This problem can be stated as follows:

Local Monotonicity
Instance: Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a Bayesian network where Γ is composed of rational probabilities. Let $\Omega(X)$ denote the set of values that $X \in \mathbf{V}$, with $k = \max_X(\mid \Omega(X) \mid)$, and let $k!$ denote the maximal number of interpretations of the values of a variable in the network.
Question: Is there an interpretation I_X for all $X \in \mathbf{V}$ such that $\mathcal{B}$ is locally monotone in distribution?

If the number of values per variable is arbitrary large, then the number of interpretations increases exponentially. In practice, however, the number of values per variable is limited.

In this paper we assume that k, the maximum number of values per node, is fixed. In the remainder of this section, we present an algorithm, with running time (for fixed k) exponential only in the treewidth[3] of the graph. The algorithm consists of a *preprocessing phase*, in which we construct a tree decomposition of (a reduced subset of) the graph, and a *dynamic programming phase*. In the preprocessing phase, we calculate monotonicity schemes of all arcs, calculate *allowed sets* of all variables and *arc constraints* between variables, reduce the graph using these allowed sets and arc constraints, and construct a tree decomposition of this reduced graph. In the next section, we show how monotonicity schemes are constructed, and in Section 3.2 we discuss the calculation of allowed sets and arc constraints. Using these monotonicity schemes, allowed sets, and arc constraints, we reduce $\mathcal{B}$ to a constraint satisfaction problem $\mathcal{C}$, and show how a tree decomposition T_C can be constructed (Section 3.3). We present a dynamic program to decide LOCAL MONOTONICITY in Section 3.4, and illustrate the algorithm on an example network in Section 3.5.

3.1 Constructing monotonicity schemes

Trivially, computing a monotonicity scheme for any node takes $O((k!)^2)$, since there are $k!$ interpretations for both ends of the arc, and computing local monotonicity takes time, linear in the size of the conditional probability table. Notice that the complexity of calculating schemes for an arc whose endpoint has multiple predecessors, is proportional in the size of the input (i.e., the conditional probability table). If all other variables in the graph have an arc towards this endpoint, there are $\prod_{i=1}^{n} \mid X_i \mid$ configurations that we need to consider. However, the conditional probability table has size $O(\prod_{i=1}^{n} \mid X_i \mid)$ as well, since we assumed explicit probability representation.

This running time can be reduced to $O(\frac{1}{2}(k!)^2)$ in the worst case, and $O(2(k!))$ in the best case, by exploiting the following observation. If a relation $X \to Y$ is monotone for two distinct interpretations $\sigma, \sigma' \in \mathbf{I}_X$ given an interpretation $\tau \in \mathbf{I}_Y$, then there are at least two equal columns in the joint probability table, i.e., $\Pr(y_k \mid x_i) = \Pr(y_k \mid x_j)$ for all $y_k \in \Omega(Y)$. But then, $M_{XY}(\sigma, \tau') = M_{XY}(\sigma', \tau')$ for *all* interpretations $\tau' \in \mathbf{I}_Y$. Of course, in two distinct interpretations[4] there exist i and j such that $x_i \le x_j$ in one interpretation and $x_j \le x_i$ in the other. From this property follows, that two columns in a monotonicity scheme are either equal or disjoint. It suffices to observe that there exists a $\tau \in \mathbf{I}_Y$ such that $M_{XY}(\sigma, \tau) = M_{XY}(\sigma', \tau) = $ TRUE to conclude that this is the case for *all* $\tau \in \mathbf{I}_Y$.

3.2 Calculating allowed sets and arc constraints

If all arcs $(Y \in \pi(X), X)$ and $(X, Z \in \sigma(X))$ have a factorizing monotonicity scheme, an interpretation I_X for X that is an element of $(\bigcap_{Y \in \pi(X)} \mathbf{M_{YX}}) \cap (\bigcap_{Z \in \sigma(X)} \mathbf{M_{XZ}})$ is always an interpretation that can be chosen for X without violating local monotonicity of the network. Of course, not all monotonicity schemes are factorizing. If $\pi(X)_f$ and $\sigma(X)_f$ denote the predecessors, respectively children of X such that $(Y \in \pi(X)_f, X)$, respectively $(X, Z \in \sigma(X)_f)$ are arcs with factorizing monotonicity schemes, we will denote $\mathcal{M}_X$ as the *allowed set* of X, where $\mathcal{M}_X = (\bigcap_{Y \in \pi(X)_f} \mathbf{M_{YX}}) \cap (\bigcap_{Z \in \sigma(X)_f} \mathbf{M_{XZ}}) \cap \mathbf{I_X}$. Note, that the allowed set consists of interpretations that can be chosen, if all arcs *without* factorizing monotonicity schemes would be removed. In other words, there exists a network $\mathbf{G}' = (\mathbf{V}, \mathbf{A}')$ where $\mathbf{A}'$ is

[3]See for example [Bod06] for a discussion of treewidth and tree decompositions.

[4]Note that we defined σ and σ' to be distinct, only if also $\sigma' \neq \sigma^T$.

```
procedure CalculateMonotonicityScheme($X$, $Y$)
$m \leftarrow \mid \mathbf{I}(X) \mid$;
$n \leftarrow \mid \mathbf{I}(Y) \mid$;
for $i = 1$ to $m$
    $j \leftarrow 1$;
    while ($j \leq n$ and $\neg M_{XY}(I_X[i], I_Y[j])$) do $j++$;
        $k \leftarrow 1$;
        while ($k \leq i$ and $\neg M_{XY}(I_X[k], I_Y[j])$) do $k++$;
        if $k < i$ then
           $\mathbf{M_{XY}}[i] \leftarrow \mathbf{M_{XY}}[k]$;
           $j \leftarrow m$;
        else while ($j \leq n$) do $\mathbf{M_{XY}}[i][j] \leftarrow M_{XY}(I_X[i], I_Y[j])$;
        endif
end procedure
```

Figure 3: A procedure for calculating monotonicity schemes

the (possibly empty) set of arcs with factorizing monotonicity schemes, and the allowed set of $X \in \mathbf{V}$ is the set of interpretations that can be chosen without violating monotonicity of $\mathbf{G}'$.

On the other hand, the arcs in $\mathbf{A} \setminus \mathbf{A}'$ have *non*-factorizing monotonicity schemes. For each arc $(X, Y) \in \mathbf{A} \setminus \mathbf{A}'$, the set of tuples ($\sigma \in \mathbf{I}_X$, $\tau \in \mathbf{I}_Y$) such that $M_{XY}(\sigma, \tau)$, denoted as $\mathcal{A}_{XY}$, will be defined as the *arc constraints* of (X, Y). For arcs in $\mathbf{A}'$ (the arcs with factorizing monotonicity schemes), $\mathcal{A}_{XY}$ is defined as $\mathcal{M}_X \times \mathcal{M}_Y$, i.e., these arcs do not have further constraints than those implied by the allowed sets of their endpoints. The combination of allowed set and arc constraints fully defines the possible choices for interpretations that do not violate local monotonicity. If the allowed set is empty for any variable in $\mathbf{V}$, or there is an arc in $\mathbf{A}$ whose arc constraints cannot be satisfied, then $\mathcal{B}$ is not locally monotone.

3.3 Constructing a tree decomposition

Using these allowed sets and arc constraints, we can reduce $\mathcal{B}$ to a constraint satisfaction problem (CSP) $\mathcal{C}$. A CSP is a 3-tuple $< \mathbf{V_C}, \mathbf{D}, \mathbf{C} >$, where $\mathbf{V_C}$ denotes a set of variables, $\mathbf{D}$ denotes their domain, and $\mathbf{C}$ denotes a set of constraints, which are defined as tuples denoting variables and restrictions on these variables. Trivially, our variable set $\mathbf{V_C}$ can be chosen to be equal to $\mathbf{V}$, but we can further reduce $\mathbf{V_C}$. Variables that are not endpoints of any arc constraints can be satisfied trivially by assigning any interpretation in their allowed set to that variable. So, we define $\mathbf{V_C}$ as $\{X \mid \neg\exists_{Y \in \pi X}(\mathcal{A}_{YX} = \mathcal{M}_Y \times \mathcal{M}_X \wedge \neg\exists_{Y \in \sigma X}\mathcal{A}_{XY} = \mathcal{M}_X \times \mathcal{M}_Y)\}$. The domain $\mathbf{D}$ of these variables is, for each variable X, the allowed set $\mathcal{M}_X$. The constraint set $\mathbf{C}$ consists of tuples $< t, R >$ where t are the endpoints of arcs (X, Y) and R is a disjunction of constraints in $\mathcal{A}_{XY}$. Obviously, if a thus constructed CSP $\mathcal{C}$ has a solution, then $\mathcal{B}$ is locally monotone. Observe that $\mathcal{C}$ has no solution if the allowed set $\mathcal{M}_X$ is empty for any variable X.

To solve $\mathcal{C}$, we construct a tree decomposition[5] T_C of the underlying subgraph induced by

[5]Also known as a *junction tree* in the literature on probabilistic networks.

$\mathbf{V_C}$. A tree decomposition is defined as follows.

Definition 8 (Tree decomposition [RS86]). A *tree decomposition* of a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is a pair $< \mathbf{X}, T >$, where $T = (I, F)$ is a tree, and $\mathbf{X} = \{X_i \mid i \in I\}$ is a family of subsets (or *bags*) of $\mathbf{V}$, one for each node of T, such that

- $\bigcup_{i \in I} X_i = \mathbf{V}$,
- for all edges $(V, W) \in \mathbf{E}$ there exists an $i \in I$ with $V \in X_i$ and $W \in X_i$, and
- for all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition $((I, F), \{X_i \mid i \in I\})$ is $\max\limits_{i \in I} \mid X_i \mid -1$. The *treewidth* of $\mathbf{G}$ is the minimum width over all tree decompositions of $\mathbf{G}$.

To facilitate the analysis, we will assume that T_C is a so-called *nice* tree decomposition (see [Bod97]). Every tree decomposition T can be converted in linear time to a nice tree decomposition of the same width and a size linear in T, where T is rooted and binary and has the following four node types:

- Leaf nodes i are leaves of T and have $\mid X_i \mid = 1$.
- Introduce nodes i have one child j with $X_i = X_j \cup \{Y\}$ for some vertex $Y \in \mathbf{V}$.
- Forget nodes i have one child j with $X_i = X_j - \{Y\}$ for some vertex $Y \in \mathbf{V}$.
- Join nodes i have two children j_1, j_2 where $X_i = X_{j_1} = X_{j_2}$.

For each fixed W, a tree decomposition of a graph with treewidth W can be constructed in $O(n)$ time with the algorithm of [Bod96]. While this linear algorithm may not be very practical, efficient heuristics and exact algorithms for small values of n are known; see e.g. [Bod06]. In the next section, we will discuss a dynamic programming algorithm on T_C.

3.4 Deciding Local Monotonicity

Here we present a dynamic programming algorithm that works on the tree decomposition T_C and decides LOCAL MONOTONICITY in $O(m \cdot (k!)^W \cdot \Gamma)$, where $O(\Gamma)$ is the size of the joint probability distribution of $\mathcal{B}$, W is the treewidth of $\mathbf{V_C}$, m is the number of arcs in $\mathbf{G}$, and k is the maximal number of values per node in $\mathbf{V_C}$. For each bag $i \in I$, we construct a set of possible combinations of interpretations for the variables in that bag, according to the following scheme:

- Leaf nodes i: the possible combinations is the allowed set $\mathcal{M}$ for the variable in i.
- Introduce nodes i: The set of possible combinations in i is the Cartesian product of the 'introduced' variable $\mathcal{M}_Y$ and the set of all possible combinations in the child j, that satisfy the arc constraints $\mathcal{A}_{XY}$ and $\mathcal{A}_{YX}$ for all variables $X \in X_j$.
- Forget nodes: The set of possible combinations in i contains all possible combinations in j, where we ignore the 'forget' variable $\mathcal{M}_Y$

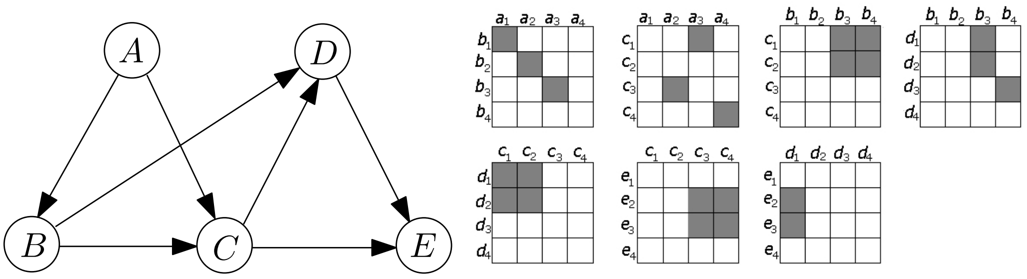


Figure 4: Example graph with monotonicity schemes

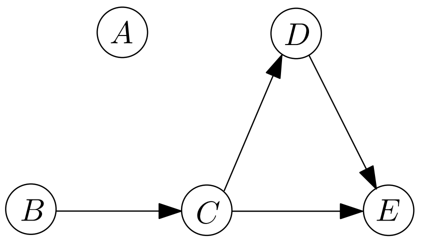
- Join nodes: The set of possible combinations in i contains the intersection of the possible combinations in j_1 and j_2.

If, during the iteration over the bags, the set of possible combinations becomes empty, then the network is not locally monotone; in the other case a satisfying interpretation for all variables exists. Such an interpretation can be found by a slightly adapted constructive algorithm. For a complexity analysis, only the introduce nodes are relevant. Observe that there are $(k!)^{|X_j|}$ possible combinations in X_j. There are at most $k!$ possible interpretations in the allowed set of the introduced variable $\mathcal{M}_Y$, and there are at most $\mid X_j \mid$ arcs (and thus sets of arc constraints) from Y to variables in X_j. Due to the symmetry of the monotonicity schemes, we can construct the possible combinations in X_i by testing all $(k!) \cdot (k!)^{|X_j|} = (k!)^{|X_j|+1} = (k!)^{|X_i|}$ elements of the Cartesian product against the X_j arc constraints, which can be done in $O(m \cdot (k!)^W)$. Calculating the monotonicity schemes and the allowed sets for all variables can be done $O((k!)^2 \cdot n) \cdot O(\Gamma)$ and calculating the arc constraints in $O((k!)^2 \cdot m)$, hence the running time of the algorithm is $O(m \cdot (k!)^W \cdot \Gamma)$.

3.5 An Example Network

In Figure 4 a small example network with monotonicity schemes is given to illustrate the algorithm. To construct a tree decomposition T_C of the reduced graph, we first construct the allowed sets of all variables. In the initial situation, the allowed set for each variable X is the set $\mathbf{I_X}$. In the next step, we calculate for each node the allowed set, by considering arcs with factorizing monotonicity schemes and intersecting the original allowed sets of both vertices covered by these arcs with the factorizing rows (for arcs entering X) respectively columns (for arcs leaving X). This step is illustrated in Figure 5 where the factorizing arcs and the allowed sets induced by the monotonicity schemes of these arcs are shown.

Furthermore, there are non-factorizing arcs which induce arc constraints. In the network, these arcs are (A, B), (A, C) and (B, D) and the corresponding arc constraints are $\mathcal{A}_{AB} = \{(a_1, b_2), (a_2, b_3), (a_3, b_4)\}$, $\mathcal{A}_{AC} = \{(a_2, c_3), (a_3, c_1), (a_4, c_4)\}$, and $\mathcal{A}_{BD} = \{(b_3, d_1), (b_3, d_2), (b_4, d_1)\}$. The variables that are endpoints of non-factorizing arcs are A, B, C and D and they induce the underlying graph in Figure 6. From this graph, we construct a nice tree decomposition as in Figure 7. We calculate, for each bag (1 to 5), the possible combinations of interpretations of the variables in that bag, starting with the leaf node.



$$\begin{aligned} A &= \{a_1, a_2, a_3, a_4\} \\ B &= \{b_3, b_4\} \\ C &= \{c_1, c_2\} \\ D &= \{d_1\} \\ E &= \{e_2, e_3\} \end{aligned}$$

Figure 5: factorizing arcs and allowed sets

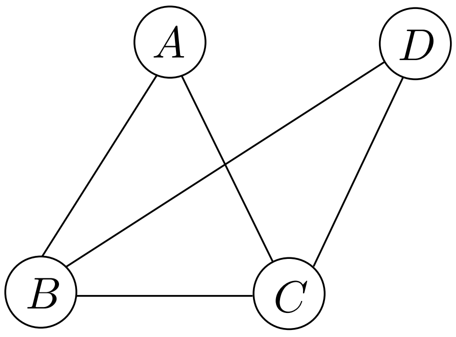


Figure 6: The underlying graph

i	node type	possible combinations	comments
5	leaf	(d_1)	allowed set of D
4	introduce	(d_1, c_1), (d_1, c_2)	factorizing arc
3	introduce	(d_1, c_1, b_3)	b_4 and c_2 are impossible
2	forget	(c_1, b_3)	just forget d_1
1	introduce	(a_3, c_1, b_3)	solution found!

Since none of the sets is empty, the graph is locally monotone, with as interpretation of the variables $(a_3, b_3, c_1, d_1, e_2)$ or $(a_3, b_3, c_1, d_1, e_3)$.

4 Max-Local Monotonicity

In this section, we formalize the problem of optimizing the number of monotone arcs, and show that it is NP-complete, i.e., infeasible in general. A similar complexity result is established for the derived problem of optimizing the number of nodes with only monotone incoming arcs. Both problems can be used as a measure for the size of the monotonicity-violating context. Furthermore, we prove that these problems — apart from infeasible to solve exactly — are hard to approximate as well. In the remainder of this section, we assume that the reader is familiar with NP-completeness proofs; more background can be found in textbooks like [GJ79] and [Pap94].

In the formal problem definitions, we assume that the (conditional) probabilities in the network are specified using rationals, rather than reals, to ensure an efficient coding of these

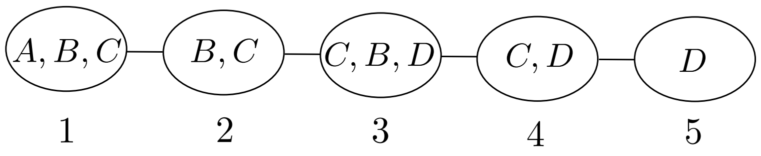


Figure 7: A tree decomposition

probabilities. Since these probabilities are often specified by experts or approximated using learning methods, this is a realistic constraint. Furthermore, we assume that the conditional probabilities in the network are coded explicitly, i.e., using look-up tables, rather than using some computable function. Lastly, for technical reasons we formulate our problems as decision problems (returning 'yes' or 'no'), rather than functions (returning a number).

Max-Local Monotonicity
Instance: Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a Bayesian network where Γ is composed of rational probabilities, and let Pr be its joint probability distribution. Let $\Omega(X)$ denote the set of values that $X \in \mathbf{V}$ can take, and let k be a positive integer $\leq | \mathbf{A} |$.
Question: Is there an interpretation I_X for all $X \in \mathbf{V}$ such that the number of arcs in G that are monotone in distribution is at least k?

Max-Nodes-Local Monotonicity
Instance: Let $\mathcal{B}$ and $\Omega(X)$ be as above, and let k be a positive integer $\leq | \mathbf{V} |$.
Question: Is there an interpretation I_X for all $X \in V(G)$ such that the number of nodes in G that have only incoming arcs that are monotone in distribution, is at least k?

Furthermore, we note that a network where some nodes are fixed (i.e., an interpretation is given) can be translated in an equivalent network with non-fixed interpretations, where the number of monotone arcs will be optimal if and only if that particular interpretation is chosen. F or example, if the ordering of a node C with values $\{low, mid, high\}$ and degree n is to be fixed at $low < mid < high$, we can enforce this condition by adding $n+1$ dummy nodes D with $\Omega(D) = \{T, F\}$ and arcs from C to these nodes that are only monotone if C has ordering $low < mid < high$, for example $\Pr(T \mid low) = 0.2$, $\Pr(T \mid mid) = 0.4$, $\Pr(T \mid high) = 0.6$. It can be easily verified that the optimal number of monotone arcs enforces the given ordering on C. In a similar way, a partial order can be guaranteed, e.g., the variable 'Stereo Sound' with values $\{none, left, right, both\}$ where no obvious ordering for '*left*' and '*right*' exists, but $none \prec left \prec both$ and $none \prec right \prec both$.

In our hardness proof, we use the Graph 3-Colorability problem, defined in [GJ79]. In this problem, the instance is an undirected graph $G = (\mathbf{V}, \mathbf{E})$, and we want to know whether there is a function $f : \mathbf{V} \to \{1, 2, 3\}$ such that $f(U) \neq f(V)$ whenever $(U, V) \in \mathbf{E}$, i.e., all nodes can be colored with three colors such that no adjacent nodes have the same color.

4.1 NP-completeness proof

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be an instance of the Graph 3-Colorability problem. From this undirected graph $\mathbf{G}$, we construct the directed graph $\mathbf{G'} = (\mathbf{V'}, \mathbf{A})$ as follows (See Figure 8):

$\mathbf{Pr}(E \mid y_1, X)$			
	e_1	e_2	e_3
x_1	0.42	0.30	0.28
x_2	0.28	0.44	0.28
x_3	0.28	0.30	0.42

$\mathbf{Pr}(E \mid y_2, X)$			
	e_1	e_2	e_3
v_1	0.44	0.28	0.28
v_2	0.28	0.44	0.28
v_3	0.28	0.28	0.44

$\mathbf{Pr}(X \mid y_3, X)$			
	e_1	e_2	e_3
v_1	0.44	0.28	0.28
v_2	0.28	0.44	0.28
v_3	0.28	0.28	0.44

Table 2: Conditional probability table for node E_1 with incoming arcs from X and Y

- for $X \in \mathbf{V}$, $\mathbf{V}'$ contains a node X.
- for $(X, Y) \in \mathbf{E}$, $\mathbf{V}'$ contains nodes $E_1, E_2, E_3, E_4, E_5, E_6$.
- for $(X, Y) \in \mathbf{E}$, $\in \mathbf{A}$ contains $(X, E_1), \ldots, (X, E_6), (Y, E_1), \ldots, (Y, E_6)$.
- $\Omega(X) = \{x_1, x_2, x_3\}$ for all $X \in \mathbf{V}'$.

We number the interpretations of all nodes in $\mathbf{V}'$ as follows:

- $i_1 = x_2 < x_1 < x_3$.
- $i_2 = x_1 < x_2 < x_3$.
- $i_3 = x_1 < x_3 < x_2$.

Now, for all nodes E_i we construct a conditional probability table such that $M(I_X, I_{E_i})$ has the following monotonicity scheme:

E_i	E_1	E_2	E_3	E_4	E_5	E_6
I_X	$\{i_1, i_2\}$	$\{i_1, i_2\}$	$\{i_1, i_3\}$	$\{i_1, i_3\}$	$\{i_2, i_3\}$	$\{i_2, i_3\}$
I_{E_i}	$\{i_2, i_3\}$	$\{i_1, i_3\}$	$\{i_1, i_2\}$	$\{i_2, i_3\}$	$\{i_1, i_2\}$	$\{i_1, i_3\}$

and the probability table for the arc (Y, E_i) is such that (Y, E_i) is a monotone relation if and only if $I_Y = I_{E_i}$. An example of such a table is given in Table 2; the other tables can be generated likewise. Observe, that a graphical representation of these schemes would be a 2×2 square, which is *transposed* from the origin. We claim that, in the thus constructed network, there is a maximum of eight arcs that have a monotone relation, if $I_X = I_Y$, and nine arcs if $I_X \neq I_Y$. We assume, without loss of generality, that $I_Y = i_1$. If we choose $I_{E_i} = i_1$ for all E_i, then all six outgoing arcs from Y to E_i have monotone relations. Now there are two cases:

- $I_X = i_1$. There are two monotone relations: $(X,\ E_2)$ and $(X,\ E_3)$. Both E_2 and E_3 have only monotone incoming arcs.
- $I_X = i_2$ or i_3. There are *three* monotone relations: either $(X,\ E_2)$, $(X,\ E_5)$ and $(X,\ E_6)$; or $(X,\ E_3)$, $(X,\ E_5)$ and $(X,\ E_6)$, which all have only monotone incoming arcs.

Note that there is no way to make *more* than three monotone arcs. We will use this construct to prove NP-hardness.

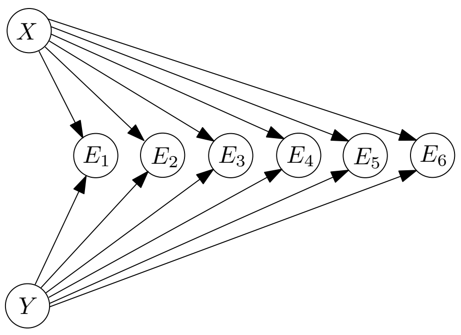


Figure 8: Construction with 6 extra nodes

Theorem 9. *Max-Local Monotonicity and Max-Nodes-Local Monotonicity are NP-complete.*

Proof. Membership of NP is trivial for both problems. Using a certificate that consists of interpretations for all vertices, we can easily test whether at least k arcs are monotone in distribution, or at least k nodes have the property that all incoming arcs are monotone in distribution. To prove *NP*-hardness, we construct a transformation from the Graph 3-Colorability problem. Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be an instance of this problem, and let $\mathbf{G}' = (\mathbf{V}', \mathbf{A})$ be the directed acyclic graph the is constructed from this instance, as described above. If and only if $9 \cdot \mid \mathbf{E} \mid$ arcs in $\mathbf{G}'$ are monotone, then all nodes X and Y that were adjacent in $\mathbf{G}$, have different interpretations, hence $\mathbf{G}$ would be 3-colorable. Since $\mathbf{G}' = (\mathbf{V}', \mathbf{A})$ can be computed from $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ in polynomial time, we have a polynomial-time transformation from Graph 3-Colorability to the Max-Local Monotonicity problem, which proves NP-hardness of the latter. A similar argument holds for the number of nodes with only monotone incoming arcs, therefore Max-Nodes-Local Monotonicity is NP-hard as well. □

Note that the above proof construct ensures that the problems remain NP-complete if k, the maximal number of values per node, is at most three.

5 Non-Approximability of Max-Local Monotonicity

For an optimization problem that is NP-complete, the question arises, how close the solution can be approximated in polynomial time. In this section, we will show that Max-Local Monotonicity is *APX-hard*, which means that it cannot be approximated to a factor arbitrarily close to 1 (unless P = NP). The result will be obtained by an *L-reduction* from Max-3-Colorability.

In Subsection 5.1 we recall the definitions of APX-hardness. In Subsection 5.2 we introduce L-reductions. In Subsection 5.3 we show that the reduction given in the previous section actually is an L-reduction, and consequently, Max-Local Monotonicity is APX-hard.

5.1 The Class APX-Hard

The results from this subsection are taken from Crescenzi and for more details we refer to his work [Cre97]. A maximization problem is characterized by the set of its instances and a measurement function m on solutions. If y is a (feasible) solution for instance x, $m(x,\, y)$ gives its *value*. The value of the *best* solution to instance x is denoted $\mathbf{opt}(x)$, and the quality of any solution y can be measured by the ratio $R(x,\, y) = \frac{\mathbf{opt}(x)}{m(x,\, y)}$.

The best solution can be hard to find, but an algorithm with *performance guarantee* r will find a solution that is within a factor r of the best one. Formally, if A is an algorithm that returns a solution $A(x)$ for instance x, A has performance guarantee r if $m(x,\, A(x)) \geq \mathbf{opt}(x)/r$, or equivalently, $R(x,\, A(x)) \leq r$ for every instance x.

A relevant distinction exists between problems that can be approximated with *some* ratio (class APX) and problems that can be approximated with *every* ratio (PTAS). A *polynomial time approximation scheme* is an algorithm that can be parameterized with any desired performance guarantee $r > 1$.

Definition 10. Problem Mx belongs to APX if there exists an $r > 1$ and a polynomial time algorithm A that has performance guarantee r.
Problem Mx belongs to PTAS if for every $r > 0$ there exists a polynomial time algorithm A_r that has performance guarantee r.

(The running time of A_r must be polynomial in the input size for fixed r, but may grow large when r gets closer to 1.) PTAS membership is a stronger property than APX membership, and if P $\neq$ NP, Max-3-Colorability (see below) is in APX but not in PTAS. Actually, Max-3-Colorability is *APX-hard* [PY91].

An optimization problem Mx to which Max-3-Colorability can be reduced is also APX-hard, and has no polynomial time approximation scheme (unless P = NP). We shall discuss approximation preserving reductions in the next subsection, including the L-reduction that we use to prove APX-hardness of Max-Local Monotonicity

5.2 Reductions: The L-Reduction

The general idea of a reduction is to see if problem MaxA can be solved using a MaxB *oracle*; again we heavily rely on Crescenzi [Cre97]. The reduction uses efficiently computable functions f and g, where f maps an instance of problem MaxA to an instance of problem MaxB and g maps a solution to problem MaxB to a solution of MaxA. If B is an algorithm that computes solutions to instances of MaxB, $A(x) = g(B(f(x)))$ is an algorithm that computes solutions to instances of MaxA. The reduction should allow conclusions about the performance of this algorithm, assuming a certain performance of B. In the following, let ϕ mean a function from $\mathbf{Q}_{>1}$ to $\mathbf{Q}_{>1}$.

The reduction $(f,\, g)$ is an *A-reduction* if an assumed performance guarantee r of B implies some performance guarantee $\phi(r)$ for A:

$$R_B(f(x),\, y) \leq r \Rightarrow R_A(x,\, g(y)) \leq \phi(r).$$

If such a reduction exists and MaxB is in APX, also MaxA is in APX. However, an A-reduction does *not imply* that if MaxB can be arbitrarily well approximated (PTAS), then also MaxA can be arbitrarily well approximated. The definition of the A-reduction does not imply that even if r get very close to 1, then also $\phi(r)$ gets arbitrarily close to 1. (For example, if $\phi(r) = r + 1$, even a PTAS for MaxB would not approximate MaxA better than with performance guarantee 2.)

PTAS membership is preserved with a very similar reduction, the P-reduction, where a desired performance r for A is considered, and the necessary performance for B can be computed as $\phi(r)$. Formally, the reduction is a *P-reduction* if there exists a ϕ such that

$$R_B(f(x), y) \leq \phi(r) \Rightarrow R_A(x, g(y)) \leq r.$$

If such a reduction exists and MaxB is in PTAS, also MaxA is in PTAS. Indeed, assume we desire to approximate MaxB with some performance guarantee r. Because MaxB is in PTAS, MaxB can be approximated (by algorithm $B_{\phi(r)}$) with performance guarantee $\phi(r)$, and now the algorithm $A_r(x) = g(B_{\phi(r)}(f(x)))$ computes the desires approximation.

Conversely, if a P-reduction exists and MaxA is APX-hard, then also MaxB is APX-hard. We will show in the next subsection that our reduction in Section 4 is a P-reduction. It is quite straightforward to show that for our reduction, the *absolute error* in the computed coloring is equal to the absolute error in the computed Max-Local Monotonicity instance. Here the absolute error of a computed solution y is defined as $E(x, y) = \mathbf{opt}(x) - m(x, y)$. But a small *absolute error* can still mean a very bad *ratio* if the value of the optimal solution of the original problem is small. Therefore, conclusions from absolute errors can be drawn only if the value of an optimal solution is lower bounded. The reduction is an *L-reduction* if there exist α, β such that:

1. $\mathbf{opt}_A(x) \geq \mathbf{opt}_B(f(x))/\alpha$;
2. $E_A(x, g(y)) \leq \beta \cdot E_B(f(x), y))$.

An L-reduction is also a P-reduction [Cre97] because performance $\phi(r) = (1 - \frac{1}{\alpha\beta}(1 - \frac{1}{r}))^{-1}$ for problem MaxB implies ratio r for problem MaxA.

5.3 Max-3-Colorability L-Reduces to Max-Local Monotonicity

Max-3-Colorability asks to color graph G, with n nodes and m edges, using three colors, maximizing the number of bichrome edges.

Lemma 11. *$\mathbf{opt}_{\mathrm{M3C}}(G) \geq 2/3m$.*

Proof. A probabilistic argument uses that in a random coloring, each edge has probability 2/3 to be bichrome. Consequently, the expected number of bichrome edges in a coloring is $2/3 \cdot m$, so the best possible coloring has at least $2/3 \cdot m$ bichrome edges.

A constructive argument colors the nodes greedily, giving each node the color that occurs least often among its (already colored) neighbors. This makes monochrome at most 1/3 of the edges that become completely colored in this step. Consequently, the resulting coloring has at least $2/3 \cdot m$ bichrome edges. □

Let $f(G)$ denote the probabilistic network constructed in Section 4. This network has $n + 6m$ nodes and $12m$ edges, of which at most $9m$ can be made monotone. A solution I for this Max-Local Monotonicity instance is an interpretation; our function g colors each node in G with the interpretation of the corresponding node in $f(G)$.

Corollary 12. *$\mathbf{opt}_{\mathrm{M3C}}(G) \geq \mathbf{opt}_{\mathrm{MLM}}(f(G))/13.5$.*

Proof. Follows from $\mathbf{opt}_{\mathrm{M3C}}(G) \geq 2/3 \cdot m$ and $\mathbf{opt}_{\mathrm{MLM}}(f(G)) \leq 9m$. □

The construction guarantees that in $f(G)$, exactly $8m+\mathbf{opt}(G)$ edges can be made monotone. Any solution with $8m+k$ monotone edges necessarily has exactly $m-k$ of the edge widgets with 8 monotone edges, and k of them with 9 monotone edges. Function g then returns a coloring with k bichrome edges.

Corollary 13. $E_{\mathrm{M3C}}(G, g(I)) = E_{\mathrm{MLM}}(f(G), I)$.

We conclude that our reduction (f, g) is a P-reduction of Max-3-Colorability to Max-Local Monotonicity. As Max-3-Colorability is an APX-hard problem, Max-Local Monotonicity is also APX-hard, which means that it does not have a polynomial time approximation scheme (unless P = NP). We do not know if Max-Local Monotonicity is in APX.

We finally remark that the formal definition of the class APX-hard is based on a slightly more general type of reduction, where also the computation of the functions f and g (and their time complexity) can depend on the required performance r. Because these reductions are not needed to prove non-approximability of Max-Local Monotonicity, we refer to [Cre97] for details.

6 A branch-and-bound algorithm

In the previous section we proved that there does not exist a PTAS for Max-Local Monotonicity unless P = NP. However, there might exist approximations for Max-Local Monotonicity that are within a fixed ratio r. Nevertheless, r may be very large and such approximations may not be particularly useful. Therefore, we now construct an exact algorithm for this problem, based on a so-called branch-and-bound strategy (see for example [WN88]). In such a strategy, the set of possible solutions is partitioned (the branch step), and upper (or lower, for minimalization problems) bounds for this partition are calculated. Whenever these bounds are lower than or equal to the current best solution (i.e., further exploration of these branches will not lead to a better solution) the branch is terminated, and other, yet unvisited branches are explored. This procedure continues until all branches terminate (we can return an optimal solution), or a given ratio between current best solution and upper bound is reached (we can return a 'good enough' solution).

6.1 Initial heuristic - a lower bound

In this section we discuss how a lower bound on the number of monotone arcs can be calculated in polynomial time (for fixed k). To compute a lower bound heuristic, we consider only arcs that have *factorizing* monotonicity schemes. For a network $\mathbf{G}$ we construct $\mathbf{G}' = (\mathbf{V}, \mathbf{A}')$ where $\mathbf{A}'$ is the (possibly empty) set of arcs with factorizing monotonicity schemes, and the allowed set of all $Z \in \mathbf{V}$ is the set of interpretations that can be chosen without violating monotonicity of $\mathbf{G}'$. Now, we can calculate a lower bound for the maximal number of arcs in $\mathbf{G}$ that can be made monotone as follows. We initialise $\mathcal{M}_Z$ to $\mathbf{I_Z}$ for all $Z \in \mathbf{V}'$ and A^+ to the empty set, and iteratively consider arcs in $\mathbf{A}'$. If an arc does not cause any allowed set to become empty, it is added to A^+, and $\mathcal{M}$ is adapted for both endpoints of that arc. On the other hand, if the arc does lead to an empty allowed set, it is dismissed. After considering all arcs in $\mathbf{A}'$, $\mid A^+ \mid$ is a lower bound on the optimal number of monotone arcs in $\mathbf{G}$.

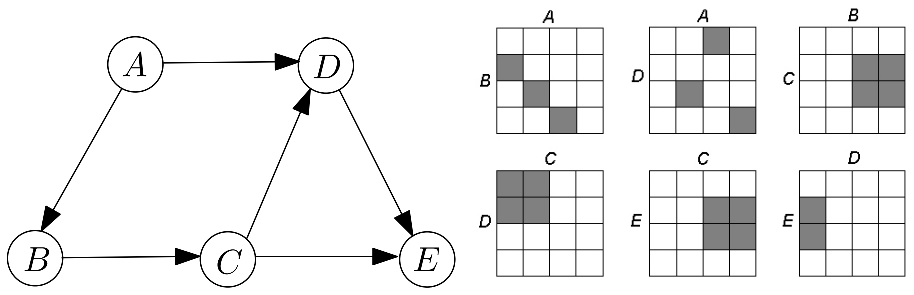


Figure 9: An example graph

6.2 Branching and bounding

Using this lower bound, we consecutively branch on the possible interpretations of the nodes, terminating branches whose upper bound is not higher than the current best solution (or lower bound). While different strategies can be followed to choose a node to branch on at any step in the algorithm, a reasonable heuristic is to pick the node that has the highest degree of all unexplored nodes. We fix the interpretation of the variable we branch on (i.e., the allowed set is a singleton, corresponding with the branch value) and calculate how many factorizing arcs remain monotone in the network. This value is added to the number of non-factorizing arcs; this is an upper bound for the total number of monotone arcs in the network.

Of course, there are many degrees of freedom in this branch-and-bound strategy. We chose to compute rather loose bounds; one can compute tighter bounds by considering a number of non-factorizing arcs that can be made monotone. Nevertheless, the constraints imposed by these arcs might require re-evaluation of all allowed sets in the network, so there is a tradeoff between the tightness of the bounds - and thus the number and depth of the branches - and the time needed to calculate such bounds.

6.3 An example

We will use the graph in Figure 9 as a example to sketch our branch-and-bound algorithm. We assume that, for every variable, only four interpretations are relevant; we will denote a particular interpretation with indexed lowercase variables, e.g., $\mathbf{I_C} = \{c_1, c_2, c_3, c_4\}$. On the right part of Figure 9, the monotonicity schemes for the arcs in the graph are shown. For example, (A, B) is monotone if $I_A = a_1$ and $I_B = b_2$.

We start with the heuristic lower bound calculated in Section 6.1. The factorizing arcs are (B, C), (C, D), (C, E), and (D, E), and if we consider these in this order and calculate the allowed sets for all nodes, we will find that we can make at least three arcs monotone, namely (B, C), (C, D), and (D, E). The lower bound will thus be three in this example. Now we branch on one of the nodes with maximal degree, say C, and explore the branches $I_C = c_1$, $I_C = c_2$, $I_C = c_3$, and $I_C = c_4$, terminating branches with an upper bound lower than three. Eventually, the algorithm will find the optimal solutions $\{I_A = a_3, I_B = b_4, I_C = c_3, I_D = d_1, I_E = e_2 \vee e_3\}$.

7 Conclusion

Optimizing the number of monotone arcs in a network, and thus minimizing the number of '?'s in the corresponding QPN, is a computationally hard problem, and hard to approximate as well, even when the number of values per variable is no more than three. The infeasibility of this problem corresponds to the even harder problem of determining whether a network is *globally* monotone [vdGBF04], or whether the values of the variables can be ordered such that this is the case [Kwi07]. We proposed a branch-and-bound approach to calculate optimal orderings. This approach may work rather well in practice with 'real world' networks, provided that the number of values per node is small. For example, in the ALARM-network [BSCC89] the maximum number of values per variable is four, and our implementation of the algorithm will find the optimal ordering in a few seconds. However, for networks where some nodes have a large range of possible values, this approach will be infeasible. Other methods must be used in such cases to calculate or approximate an optimal solution.

Acknowledgements

This research has been (partly) supported by the Netherlands Organisation for Scientific Research (NWO). An earlier version of this paper appeared in the Proceedings of the Ninth ECSQARU conference [KBT07]. The authors wish to thank Linda van der Gaag and Silja Renooij for their insightful comments on this subject and access to the information stored in the Oesophageal Cancer network, and Jesper Nederlof for his work on the implementation of the algorithm.

References

[Bod96] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[Bod97] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proceedings of the Twenty-second International Symposium on Mathematical Foundations of Computer Science*, volume LNCS 1295, pages 19–36. Springer-Verlag, 1997.

[Bod06] Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In Fedor V. Fomin, editor, *Proceedings 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2006)*, pages 1–14. Springer, Lecture Notes in Computer Science, volume 4271, 2006.

[BSCC89] I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on AI and Medicine*, pages 247–256. Springer-Verlag, 1989.

[Cre97] P. Crescenzi. A short guide to approximation preserving reductions. In *12th Annual IEEE Conference on Computational Complexity (CCC'97)*, pages 262–273. IEEE, 1997.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, 1979.

[KBT07] J. Kwisthout, H. Bodlaender, and G. Tel. Local monotonicity in probabilistic networks. In K. Mellouli, editor, *Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty. October 31-November 2, 2007, Hammamet, Tunisia*, volume 4724 of *LNCS*, pages 548–559. Springer-Verlag, 2007.

[Kwi07] J. Kwisthout. The computational complexity of monotonicity in probabilistic networks. In E. Csuhaj-Varj and Z. Esik, editors, *Sixteenth International Symposium on Fundamentals of Computation Theory, August 27-30, 2007, Budapest, Hungary*, volume 4639 of *LNCS*, pages 388–399. Springer-Verlag, 2007.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

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

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[RS86] N. Robertson and P.D. Seymour. Graph minors II: Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[vdGBF04] L. C. van der Gaag, H.L. Bodlaender, and A. Feelders. Monotonicity in Bayesian networks. In *Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 569–576. AUAI Press, 2004.

[vdGRW$^+$02] L. C. van der Gaag, S. Renooij, C. L. M. Witteman, B. M. P. Aleman, and B. G. Taa. Probabilities for a probabilistic network: a case study in oesophageal cancer. *Artificial Intelligence in Medicine*, 25:123–148, 2002.

[Wel90] M. P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44(3):257–303, 1990.

[WN88] L. A. Wolsey and G. L. Nemhauser. *Integer and Combinatorial Optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, 1988.