The Computational Complexity of Probabilistic Networks

De computationale complexiteit van probabilistische netwerken

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof. dr. J.C. Stoof, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op 29 oktober 2009 des middags te 2.30 uur

door

Johan Henri Petrus Kwisthout

geboren op 19 mei 1976
te Breda

Promotoren:	prof. dr. J. van Leeuwen
	prof. dr. ir. L.C. van der Gaag
Co-promotoren:	dr. H.L. Bodlaender
	dr. G. Tel

This research was financially supported by the Netherlands Organisation for Scientific Research (NWO) in the framework of the *Algorithmic Complexity of Probabilistic Networks* project.

Voorwoord

Voor u ligt het resultaat van vier jaar promotieonderzoek. Zoals iedere promovendus heb ook ik de hoogte- én dieptepunten in dat promotietraject meegemaakt: van het vinden van een elegant bewijs voor een probleem, tot het vinden van een cruciale fout in een bewijs; van het jubelende *We are pleased to inform you that your paper has been selected...* tot het deprimerende *We are sorry to inform you...*; van de opluchting van *de eerste versie is af* tot het ontnuchterende *maar er moet nog veel aan gesleuteld worden.* Hoop, wanhoop, blijdschap, frustratie, opluchting en tal van andere gevoelens hebben in vier jaar de revue gepasseerd, waarbij het venijn 'm in de staart zat, maar: *all's well that ends well*, zoals William Shakespeare al schreef.

Een proefschrift schrijven is geen solistische bezigheid. Mijn dank gaat daarom uit naar al degenen die een bijdrage hebben geleverd aan de totstandkoming. In (ongeveer) chronologische volgorde zijn Peter V., Peter L., Emgad, Guido, Eelko, Thomas, Johan van R., en Bart mijn directe collega's geweest als mede-promovendi bij de groep Algorithmic Systems, waarvan Guido en Eelko een deel van de periode als kamergenoot. Dank aan de voorgangers voor hun hints en tips, en succes voor degenen die het *zwaard van Damocles* nog boven zich weten. Met mijn collega's was het altijd prettig van gedachten wisselen tijdens de lunch: over voetbal (zeker wanneer NAC van Ajax of Heerenveen had gewonnen), over stamboomonderzoek, de politieke actualiteit in Den Haag of in het Bestuursgebouw, de nieuwbouwplannen van de universiteit, en soms zelfs over informatica. Speciale dank voor Wilke die dagelijks trouw zijn appeltje met ons deelde.

Hans, Gerard, Jan en Linda: als (co-) promotoren hebben jullie een cruciale rol gespeeld in de totstandkoming van het proefschrift, waarbij een ieder een specifiek stuk kennis en ervaring in de begeleiding in bracht. Dank voor jullie inbreng: de kwaliteitscontrole, de uitleg en het begrip als ik het even de draad kwijt was, de handige LaTeX-tips, het zorgvuldige meelezen en daarbij *en passant* verbeteren van mijn schrijfstijl, het enthousiasme en de gemeende felicitaties als er papers geaccepteerd waren, de evenzeer gemeende opbeurende woorden als er papers afgewezen waren, de gelegenheid om in de praktijk ervaring op te doen met het voorbereiden en geven van onderwijs in verschillende vormen, de lovende beoordelingen bij de jaarlijkse R&O gesprekken, aan de andere kant soms ook de onverbloemde maar terechte kritiek als ik wat te zeer met de *franse slag* te werk was gegaan, en alle andere aspecten van begeleiding die ik vergeet.

Dank aan de leden van mijn leescommissie (J. Kok, P. van Emde Boas, R. Verbrugge, C. Witteveen en S. Renooij) voor het precieze lezen en becommen-

tariëren van mijn proefschrift, maar ook voor uw geduld en flexibiliteit toen het beloofde manuscript wat op zich liet wachten. Dank aan Bas Maes voor het ontwerpen van de mooie cover van dit proefschrift.

Ook anderen dan mijn directe collega's en begeleiders hebben een steentje bijgedragen. Natuurlijk de leden van de groep Decision Support Systems die vaak mijn reisgenoten waren naar buitenlandse conferenties. Maar ook mijn afstudeerbegeleider Mehdi Dastani die nog regelmatig kwam buurten en belangstelling toonde, en me inwijdde in de geheimen van het organiseren van een conferentie. Mijn afstudeerbegeleiders in Nijmegen, Pim Haselager en Ton Dijkstra, die accepteerden dat ik midden in mijn afstuderen bij Kunstmatige Intelligentie even alles aan de kant schoof om een promotietraject te starten, er voor zorgden dat ik de moed erin hield bij het in de vrije tijd schrijven van mijn tweede scriptie, maar ook daarna betrokken bleven bij het schrijven van een tijdschriftartikel en mijn perspectieven na de promotie. Collega's die interesse toonden in mijn werk: ik noem met name Leen Torenvliet, Bas Terwijn en Iris van Rooij. Maar ook buitenlandse 'grootmachten' als Lane Hemaspaandra en Finn V. Jensen die de moeite namen om mijn vragen te beantwoorden en tips te geven, Wolfgang Thomas en Erich Grädel voor hun uitnodiging om in Aken te participeren in hun onderzoek, en natuurlijk de vele onbekende reviewers die soms erg lovend en soms erg kritisch waren, maar in ieder geval ruim de tijd namen om mijn artikelen kritisch te lezen en opbouwende feedback te geven.

Maar ook dank voor degenen die wat meer op de achtergrond staan, maar toch een belangrijke bijdrage leveren aan het werk wat op een universiteit gedaan wordt: het Ondersteunend en Beheerspersoneel (OBP). Oftewel de secretariële ondersteuning, de balie, de bibliotheek, de catering en de schoonmakers.

Natuurlijk bestaat er ook een leven buiten de universiteit. Dank daarom aan mijn (schoon-)familie, vrienden en vriendinnen, en mijn politieke kameraden voor jullie steun, interesse, afleiding, en soms de meewarige blik en de herhaalde vragen 'wanneer ik nou precies professor ben' en of ik even naar hun PC kon kijken ('jij doet immers iets met computers').

Als laatste, maar voor mij het belangrijkste: dank aan mijn maatje Judith voor liefde, steun, en begrip, het samen delen van mooie momenten, het samen vieren van successen, het samen verwerken van tegenslagen. Zonder jou was ik zover misschien niet gekomen; in ieder geval had het minder waarde gehad.

Contents

1 Introduction **1**
1.1 Structure of this thesis 4

2 Preliminaries **7**
2.1 Computational complexity 7
2.2 Tree-decompositions and treewidth 12
2.3 Probabilistic Networks 14
2.4 Inference in probabilistic networks 17

I Analysis **23**

3 Sensitivity Analysis and Parameter Tuning **25**
3.1 Sensitivity analysis and tuning 27
3.1.1 Sensitivity analysis 27
3.1.2 Parameter tuning 29
3.1.3 Distance measures 30
3.2 Problem definitions 31
3.3 Completeness results 33
3.4 Restricted problem variants 37
3.4.1 Networks with bounded topologies 37

3.4.2 Bounded number of CPTs . . . 40
3.4.3 Parameterised Parameter Tuning . . . 42
3.5 Conclusion . . . 44

4 Monotonicity **45**
4.1 Monotonicity in probabilistic networks . . . 46
4.2 Loose Monotonicity in Distribution . . . 51
4.3 Weak Monotonicity in Mode . . . 54
4.4 Tunable Monotonicity in Distribution . . . 57
4.5 Conclusion . . . 60

5 Finding the kth MPE and kth Partial MAP **61**
5.1 Complexity classes described by metric Turing Machines . . . 64
5.2 Kth MPE . . . 66
5.3 K-th Partial MAP . . . 71
5.4 Conclusion . . . 76

II Abstraction **79**

6 Inference in Interval-based Networks **81**
6.1 Background: Qualitative Probabilistic Networks . . . 83
6.2 Enhanced QPNs . . . 86
6.3 Complexity of the problem . . . 89
6.3.1 Construction . . . 90
6.3.2 NP-hardness proof . . . 95
6.3.3 On the possible membership of NP . . . 95
6.4 Operator variants . . . 96
6.5 Conclusion . . . 99

7 Local Monotonicity **101**
7.1 Motivation . . . 102
7.2 Monotonicity and variable orderings . . . 103
7.2.1 Monotonicity functions and matrices . . . 106
7.2.2 Properties of monotonicity matrices . . . 107
7.3 Local Monotonicity . . . 108
7.3.1 Constructing monotonicity matrices . . . 109
7.3.2 Allowed sets and arc constraints . . . 109
7.3.3 Constructing a constraint satisfaction problem instance . 110

7.3.4 Deciding Local Monotonicity . . . 111
7.3.5 An example network . . . 112
7.4 Max-Local Monotonicity . . . 115
7.5 Non-Approximability . . . 119
7.5.1 Approximation preserving reductions and APX-hardness . 119
7.6 A branch-and-bound algorithm . . . 121
7.6.1 Initial heuristic - a lower bound . . . 122
7.6.2 Branching and bounding . . . 122
7.6.3 An example . . . 123
7.7 Conclusion . . . 124

III Inference **125**

8 Efficient Algorithms and Treewidth **127**
8.1 Approach . . . 129
8.2 Complexity of Inference . . . 131
8.3 Conclusion . . . 138

9 Conclusion **139**

Appendix **143**

Samenvatting **158**

Indices **162**
List of Complexity Classes . . . 162
List of Problems . . . 165
Index . . . 167

Curriculum vitae **169**

Titles in the IPA Dissertation Series **170**

Colofon **179**

Chapter 1

Introduction

In our daily life we are forced to reason with imperfect knowledge and bounded resources. We do not know all the relevant facts, we do not know which facts are relevant, and even if we would, normally we don't have the time to take everything into account. In general, our information is inconsistent, vague, or imprecise. To be helpful, computer programs that are designed to assist in decision making also need to deal with this uncertainty. Or, as Halpern (2003, p.1) puts it:

"Uncertainty is a fundamental and unavoidable feature of daily life; in order to deal with uncertainty intelligently, we need to be able to represent it and reason about it."

Probabilistic networks (Pearl, 1988; Jensen and Nielsen, 2007), also called *belief*

or *Bayesian* networks, are one of the formalisms that can be used to represent and reason about uncertain information. A probabilistic network represents a joint probability distribution on a set of stochastic variables, and is described by a directed acyclic graph and a set of conditional probabilities. The nodes of the graph represent the stochastic variables, the arcs (or lack of them) represent (in)dependencies in the joint probability distribution. The structure of the graph and the conditional probabilities in the network can be either *elicited* from domain experts, or *learned* from data. Probabilistic networks are often used in decision support systems, mainly—but not exclusively—in medical diagnosis systems (see e.g. Jensen et al., 2001; Kennett et al., 2001; Lucas et al., 2000; Wasyluk et al., 2001; Cofiño et al., 2002; van der Gaag et al., 2002; Kappen et al., 2003; Zhang et al., 2004; Geenen et al., 2006; Henriksen et al., 2007). In such systems, the network typically consists of *classification*, *observable*, and *intermediate* nodes.

For example, in the medical *Oesophagus Network* (Figure 1.1), a probabilistic network for supporting patient-specific therapy selection for oesophageal cancer (Van der Gaag et al., 2004), the tumour's *stage* is the main classification variable; whether the tumour is in an early or in a later stage, determines the selection of an appropriate therapy. The observable variables indicate diagnostic symptoms or test results, like the patient's ability to swallow food or the results of a CT-scan. Lastly, intermediate variables, for example presence of haematogenous metastases or the extent of lymph node metastases, cannot be directly observed.

Using a probabilistic network, one can infer the posterior probability distribution of a variable, given observed evidence for one or more variables. For example, given the findings of a CT-scan, we can calculate the probability distribution of the tumour stage. But probabilistic networks also allow us to reason the other way around: we can calculate the probability of a diagnostic symptom, given a particular tumour stage. Algorithms for these *inference* problems have been introduced and refined in the past decades (see e.g. Pearl, 1986; Lauritzen and Spiegelhalter, 1988; Shachter, 1986). These algorithms, however, all take exponential time in the worst case, and indeed Cooper (1990) proved NP-hardness of the inference problem. Furthermore, Dagum and Luby (1993) proved NP-hardness of *approximating* the inference problem as well. Similar hardness results were obtained for other problems related to probabilistic networks.

However, many problems in probabilistic networks are not 'merely' NP-hard.

In fact, completeness results have been obtained for classes that are assumed to be larger than NP. For example, the EXACT INFERENCE-problem has been proven to be #P-complete (Roth, 1996) and has a PP-complete decision variant (Littman et al., 2001). The PARTIAL MAP-problem is $\mathsf{NP}^{\mathsf{PP}}$-complete (Park and Darwiche, 2004) and the MONOTONICITY-problem is $\mathsf{co\text{-}NP}^{\mathsf{PP}}$-complete (Van der Gaag et al., 2004)[1]. These complexity results make probabilistic networks interesting from a complexity-theoretical viewpoint as well. In this thesis, we will present complexity results for a number of these problems, like tuning the parameters of a network, analysing the sensitivity of a network, verifying monotonicity properties, and enumerating explanations. For most of these problems (worst case exponential-time) algorithms are known; however, their exact complexity was not addressed before.

One might question the importance of knowing whether a problem is, for example, $\mathsf{NP}^{\mathsf{PP}}$-complete rather than 'merely' NP-hard, when both results will lead to the conclusion that the problem at hand is infeasible in general. Nevertheless, even when the problem is infeasible in general, it may or may not be feasible in special cases. For example, probabilistic inference can be done in polynomial time, if the network is a polytree. However, it is likely that an $\mathsf{NP}^{\mathsf{PP}}$-complete problem remains infeasible, even when inference is easy. For example, the $\mathsf{NP}^{\mathsf{PP}}$-complete PARTIAL MAP problem (which will be discussed in the next chapter) remains NP-complete, even when restricted to polytrees (Park and Darwiche, 2004). In Chapter 3 we will give a similar result for the PARAMETER TUNING problem. However, we show that a polynomial time algorithm for this problem is likely to exist when there are both restrictions on the structure of the network *and* on the number of parameters that are tuned. Thus, pinpointing the exact complexity of a problem may give us insight in the restrictions needed to make a problem feasible. Furthermore, it may aid in finding suitable approximation algorithms for the problems at hand (Majercik and Littman, 1998).

[1] These problems and the corresponding complexity classes will be discussed in more detail in the remainder of this thesis.

1.1 Structure of this thesis

In the next chapter, we introduce a number of concepts related to probabilistic networks, complexity theory, and tree-decompositions. We review known completeness results for several variants of the Inference problem, and we discuss other known complexity results on probabilistic networks. The remainder of the thesis consists of three parts. In the first part, a number of topics related to the *analysis* of probabilistic networks is discussed. In Chapter 3 we discuss the computational complexity of sensitivity analysis and parameter tuning in probabilistic networks. During the construction of a probabilistic network, one is often interested in the sensitivity of the output with respect to the parameters in the network. For example, it may be the case that a relatively small change in a particular parameter of the Oesophagus Network may change the most likely stage of the tumour. In such cases, it is important to elicit (or learn) the desired parameter with great care. Related to sensitivity analysis is *parameter tuning*. Often, one is interested in the amount of change needed in the parameter distribution to match some wanted constraint on the classification variable. Algorithms exist for sensitivity analysis and parameter tuning (Chan and Darwiche, 2004), yet the computational complexity of these problems has not been studied before. We prove, that sensitivity analysis and parameter tuning are $\mathsf{NP}^{\mathsf{PP}}$-complete, still are NP-complete when restricted to polytrees, and are PP-complete if the number of CPTs from which the parameters are taken (and the indegree of the corresponding variables) is bounded. An earlier version of this chapter appeared as Kwisthout and Van der Gaag (2008).

In Chapter 4 we discuss monotonicity analysis in probabilistic networks. When a domain expert indicates that a particular relation between one or more evidence variables and the classification variable should be monotone (e.g., more severe symptoms make a higher tumour stage more likely), then the network parameters should reflect this. However, since they are often estimated or learned from (imperfect) datasets, this is not always the case. Deciding whether the relation between sets of variables is monotone is infeasible in general: Van der Gaag et al. (2004) showed that this problem is $\mathsf{co\text{-}NP}^{\mathsf{PP}}$-complete and still is $\mathsf{co\text{-}NP}$-complete in polytrees. However, the direction of monotonicity relations (i.e., isotone or antitone) has to be specified beforehand. This restriction is sometimes too strong; however, we show that this problem remains $\mathsf{co\text{-}NP}$-complete even if a weaker notion of monotonicity is used. Furthermore, we show that tuning a network, in order to obtain monotonicity, is $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete. The

work in this chapter builds on material presented in Kwisthout (2007).

The first part of the thesis is concluded with a chapter on enumeration of most probable explanations (Chapter 5). While complexity results are known for finding the most probable explanation with either full or partial evidence, in practical applications one is often interested in *all* value assignments with a high likeliness. For example, in medical applications one would like to prescribe medication that covers not just the most probable cause of a disease, but rather a set of likely causes. We introduce the complexity classes $\mathsf{FP}^{\mathsf{PP}}$ and $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$, and show that enumerating most probable explanations is complete for $\mathsf{FP}^{\mathsf{PP}}$ (when full evidence is available) and for $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$ (when only partial evidence is available). An earlier version of this chapter appears as Kwisthout (2008).

In the second part, we discuss *abstraction* of probabilistic networks. QPNs are a qualitative abstraction of probabilistic networks, in which the conditional influences between variables are summarised in signs. While these networks have polynomial-time inference algorithms (van Kouwen et al., 2009), trade-offs modelled in the original network are sometimes lost in the abstraction. Enhanced models have been introduced (e.g., Renooij and van der Gaag, 2008) that allow a notion of strength. However, the complexity of inference in these enhanced models has not been studied yet. We show, that inference in the most general model (interval networks) is NP-hard if we allow that instances may not model actual probabilistic networks. These results appeared in an earlier version in Kwisthout and Tel (2006) and Kwisthout and Tel (2008). A related question in QPNs is the question how to *order* the values of a variable if they are not implicitly ordered already, like the values of a variable *race* or *blood type*. We show that optimising the order of these values in the network in such a way that as many influences as possible are monotone, has an NP-complete decision variant and is APX-hard, i.e., hard to approximate with arbitrary precision. Material from this chapter appeared earlier in Kwisthout et al. (2007).

In the final part, consisting of a single (yet unpublished) Chapter 8, we take a closer look at inference in probabilistic networks. There exist well-known algorithms for this problem (e.g. Shachter, 1986; Lauritzen and Spiegelhalter, 1988) that use time, exponential only in the treewidth of the network. We show that no general algorithm can solve arbitrary instances of the INFERENCE-problem with high treewidth, unless the Exponential Time Hypothesis fails. The thesis is concluded in Chapter 9.

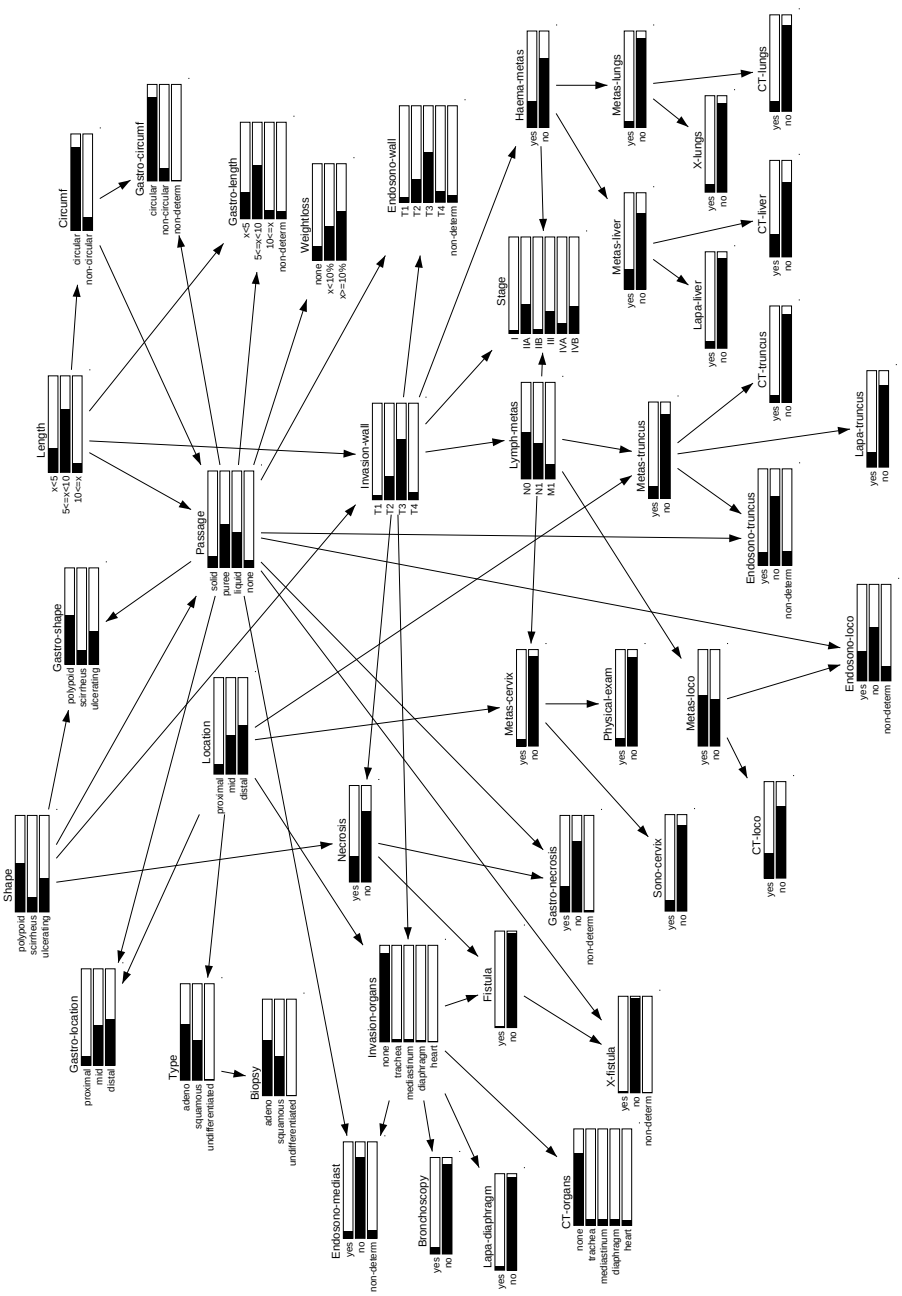


Figure 1.1 Diagnostic part of the Oesophagus network (Van der Gaag et al., 2004)

Chapter 2

Preliminaries

In this chapter we review basic concepts from computational complexity theory, graph theory, and probabilistic networks inasmuch as they are required for a good understanding in the ensuing chapters. Furthermore, we review in some detail the computational complexity of the INFERENCE-problem in probabilistic networks.

2.1 Computational complexity

In the remainder of this thesis, we assume that the reader is familiar with basic concepts of computational complexity theory, such as Turing Machines, the complexity classes P and NP, and NP-completeness proofs. While we do give

formal definitions of these concepts, we refer to classical textbooks like Garey and Johnson (1979) and Papadimitriou (1994) for a thorough introduction to these subjects.

A Turing Machine (hereafter TM), denoted by $\mathcal{M}$, consists of a finite (but arbitrarily large) one-dimensional tape, a read/write head and a state machine, and is formally defined as a 7-tuple $\langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$, in which Q is a finite set of states, Γ is the set of symbols which may occur on the tape, b is a designated *blank* symbol, $\Sigma \subseteq \Gamma \setminus \{b\}$ is a set of input symbols, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is a transition function (in which L denotes shifting the tape one position to the left, and R denotes shifting it one position to the right), q_0 is an initial state and F is a set of accepting states. In the remainder, we assume that $\Gamma = \{0, 1, b\}$ and $\Sigma = \{0, 1\}$, and we designate q_Y and q_N as accepting and rejecting states, respectively, with $F = \{q_Y\}$.

A particular TM $\mathcal{M}$ *decides* a language L if and only if, when presented with an input string x on its tape, it halts in the accepting state q_Y if $x \in L$ and it halts in the rejecting state q_N if $x \notin L$. If we only require that $\mathcal{M}$ accepts by halting in an accepting state if and only if $x \in L$ and either halts in a non-accepting state or does not halt at all if $x \notin L$, then $\mathcal{M}$ *recognises* a language L. A Turing Transducer (TT), denoted by $\mathcal{T}$, is a TM with an additional output tape. $\mathcal{T}$ *computes* a function f if and only if, when presented with an input string x, it halts in an accepting state with $f(x)$ on its output tape. If the transition function δ maps every tuple (q_i, γ_k) to at most one tuple (q_j, γ_l, p), then $\mathcal{M}$ ($\mathcal{T}$) is called a *deterministic* Turing Machine (Transducer), else it is termed as a *non-deterministic* Turing Machine (Transducer).

A non-deterministic TM accepts x if at least one of its possible computation paths accepts x; similarly, a non-deterministic TT computes $f(x)$ if at least one of its computation paths computes $f(x)$. The *time complexity* of deciding L by $\mathcal{M}$, respectively computing f by $\mathcal{T}$, is defined as the maximum number of steps that $\mathcal{M}$, respectively $\mathcal{T}$ uses, as a function of the size of the input x.

Formally, complexity classes are defined as classes of languages, where a language is an encoding of a computational problem. An example of such a problem is the Satisfiability problem: given a Boolean formula ϕ, is there a truth assignment to the variables in ϕ such that ϕ is satisfied? We will assume that there exists, for every problem, a reasonable encoding that translates arbitrary instances of

that problem to strings, such that the 'yes' instances form a language L and the 'no' instances are outside L. While we formally define complexity classes using languages, we may refer in the remainder to problems rather than to their encodings. We will thus write 'a problem Π is in class C' if there is a standard encoding from every instance of Π to a string in L where L is in C. We will use the notation $\text{bin}(x)$ if we want to refer to the *encoding* of an instance x, given a particular (fixed) encoding from instances to strings in L.

A problem Π is *hard* for a complexity class C if every problem in C can be reduced to Π. Unless explicitly stated otherwise, in the context of this thesis these reductions are polynomial-time *many-one* (or *Karp*) reductions. Π is polynomial-time many-one reducible to Π' if there exists a polynomial-time computable function f such that $x \in \Pi \Leftrightarrow f(x) \in \Pi'$. A problem Π is *complete* for a class C if it is both in C and hard for C. Such a problem may be regarded as being 'at least as hard' as any other problem in C: since we can reduce any problem in C to Π in polynomial time, a polynomial time algorithm for Π would imply a polynomial time algorithm for *every* problem in C.

The complexity class P (short for *polynomial time*) is the class of all languages that are decidable on a deterministic TM in a time which is polynomial in the length of the input string x. In contrast, the class NP (*non-deterministic polynomial time*) is the class of all languages that are decidable on a *non*-deterministic TM in a time which is polynomial in the length of the input string x. Alternatively NP can be defined as the class of all languages that can be *verified* in polynomial time, measured in the size of the input x, on a deterministic TM: for any problem $L \in \mathsf{NP}$, there exists a TM $\mathcal{M}$ that, when provided with a tuple (x, c) on its input tape, can verify in polynomial time that c is a 'proof' of the fact that $x \in L$; that is, there exists a c for which $\mathcal{M}$ accepts (x, c) in a time polynomial in the size of x, if and only if $x \in L$. We will call c a *certificate* or *witness* of membership of $x \in L$. Note that certificates are restricted to be of polynomially bounded size with respect to the length of the input.

Trivially, $\mathsf{P} \subseteq \mathsf{NP}$. Whether $\mathsf{P} = \mathsf{NP}$ is arguably the most important open problem in Computer Science presently. Note that if a polynomial-time algorithm would be found for an NP-complete problem, this would prove $\mathsf{P} = \mathsf{NP}$. However, it is widely believed (Sipser, 1992; Gasarch, 2002) that $\mathsf{P} \neq \mathsf{NP}$, thus an NP-completeness proof for a problem P would strongly suggest that no polynomial algorithm exists for P. It is common to use Satisfiability (see above) as

the standard example of an NP-complete problem; Satisfiability is therefore also called the *canonical* NP-complete problem. We will follow this example and use variants of this problem as canonical problems for various complexity classes.

The complexity class co-NP is defined as the class of all languages that, when provided with a tuple (x, c) on its input tape, can be *falsified* in polynomial time on a deterministic TM; for any problem $L \in$ co-NP, there exists a TM $\mathcal{M}$ that can verify that $x \notin L$ using the certificate c using time, polynomial in the length of x. It is known that P $\subseteq$ co-NP. It is not known whether NP $=$ co-NP, but it is widely believed that these two classes are not equal. In the remainder of this thesis, we denote the complement of a decision problem Π as not-Π, with 'yes' and 'no' answers reversed with respect to the original problem Π. Note that, by definition, if Π is in complexity class C, then not-Π is in co-C, and, likewise, if not-Π is in C, then Π is in co-C. The class #P is a function class; a function f is in #P if $f(x)$ computes the number of accepting paths for a particular non-deterministic TM when given x as input; thus #P is defined as the class of counting problems which have a decision variant in NP. The canonical complete problems for co-NP and #P are Tautology (given a formula ϕ, is it a tautology?) and #Sat (given a formula ϕ, how many truth assignments satisfy it?), respectively.

A *Probabilistic* TM (PTM) is similar to a non-deterministic TM, but the transitions are *probabilistic* rather than simply non-deterministic: for each transition, the next state is determined stochastically according to some probability distribution[1]. In the remainder of this thesis, we assume that a PTM has two possible next states q_1 and q_2 at each transition, and that the next state will be q_1 with some probability p and q_2 with probability $1 - p$. A PTM accepts a language L if the probability of ending in an accepting state, when presented an input x on its tape, is strictly larger than $\frac{1}{2}$ if and only if $x \in L$. If the transition probabilities are uniformly distributed, the machine accepts if the *majority* of its computation paths accepts. The class PP is defined as the class of languages decidable by a PTM in a time which is polynomial in the length of the input. It is known that both NP $\subseteq$ PP and co-NP $\subseteq$ PP. The canonical PP-complete

[1] In the original definition (Gill, 1977) of Probabilistic Turing Machines a 'fair random coin' was used to choose one out of two states, with uniform probability, as the next state. However it can be easily shown (see e.g. Arora and Barak, 2009) that this definition can be generalised to hold also for general random choices.

problem is Majsat: given a formula ϕ, does the majority of truth assignments satisfy it?

Another concept from complexity theory that we will use in this thesis is the *Oracle Machine*. An Oracle Machine is a Turing Machine (or Transducer) which is enhanced with an oracle tape, two designated oracle states q_{O_Y} and q_{O_N}, and an oracle for deciding membership queries for a particular language L_O. Apart from its usual operations, the TM can write a string x on the oracle tape and query the oracle. The oracle then decides whether $x \in L_O$ in a single state transition and puts the TM in state q_{O_Y} or q_{O_N}, depending on the 'yes'/'no' outcome of the decision. We can regard the oracle as a 'black box' that can answer membership queries in one step. We will write $\mathcal{M}^{\mathsf{C}}$ to denote an Oracle Machine with access to an oracle that decides languages in C. A similar notation is used for complexity classes. For example, $\mathsf{NP}^{\mathsf{SAT}}$ is defined as the class of languages which are decidable in polynomial time on a non-deterministic Turing Machine with access to an oracle deciding Satisfiability instances. In general, if an oracle can solve problems that are complete for some class C (like the PP-complete Inference-problem), then we will write NP^{C} (in the example $\mathsf{NP}^{\mathsf{PP}}$, rather than $\mathsf{NP}^{\mathsf{INF}}$). Note that $\mathsf{A}^{\mathsf{co\text{-}C}} = \mathsf{A}^{\mathsf{C}}$, since both accepting and rejecting answers of the oracle can be used.

The *Counting Hierarchy* (CH) (Wagner, 1986; Torán, 1991) is a hierarchy of classes which is constructed inductively as follows.

Definition 2.1 (Counting Hierarchy). *The Counting Hierarchy is a partial order on complexity classes such that:*

1. $\mathsf{P}, \mathsf{NP}, \mathsf{co\text{-}NP}, \mathsf{PP} \in \mathsf{CH}$;
2. *if* $\mathsf{C} \in \mathsf{CH}$, *then also* $\mathsf{C}^{\mathsf{NP}}, \mathsf{C}^{\mathsf{PP}} \in \mathsf{CH}$.

Examples of classes in CH are for example PP, $\mathsf{NP}^{\mathsf{PP}}$, and $\mathsf{co\text{-}NP}^{\mathsf{PP}}$. Canonical complete problems for these complexity classes are defined by Wagner (Wagner, 1986) using quantified Satisfiability variants. For the classes above, the canonical complete problems are Majsat, E-Majsat, and A-Majsat, respectively, as defined below. In all problems, we consider a Boolean formula ϕ with n variables $x_i, i = 1, \ldots, n, n \geq 1$, and we introduce quantifiers to bind subsets of these variables.

Majsat
Instance: Let ϕ be a Boolean formula and let $\mathbf{X}$ denote its set of variables.
Question: Does the majority of the truth assignments to $\mathbf{X}$ satisfy ϕ?

E-Majsat
Instance: Let ϕ be a Boolean formula with n variables $x_i, i = 1, \ldots, n, n \geq 1$, grouped into two disjoint sets $\mathbf{X_E} = \{x_1, \ldots, x_k\}$ and $\mathbf{X_M} = \{x_{k+1}, \ldots, x_n\}$ for some $1 \leq k \leq n$.
Question: Is there a truth assignment to $\mathbf{X_E}$ such that the majority of the truth assignments to $\mathbf{X_M}$ satisfy ϕ?

A-Majsat
Instance: Let ϕ be a Boolean formula with n variables $x_i, i = 1, \ldots, n, n \geq 1$, grouped into two disjoint sets $\mathbf{X_A} = \{x_1, \ldots, x_k\}$ and $\mathbf{X_M} = \{x_{k+1}, \ldots, x_n\}$ for some $1 \leq k \leq n$.
Question: For every truth assignment to $\mathbf{X_A}$, is ϕ satisfied for the majority of the truth assignments to $\mathbf{X_M}$?

A common approach to prove membership of a problem in a particular class NP^{C} is to show the existence of a sufficiently small certificate that can be used to *verify* existence of a solution in polynomial time, using an oracle for C. To prove membership in $\mathsf{co\text{-}NP}^{\mathsf{C}}$, one can try to show the existence of a certificate that can be used to construct a counterexample (i.e., *falsify* a claim) in polynomial time, using an oracle for C.

2.2 Tree-decompositions and treewidth

In this section, we review some concepts from graph theory, in particular tree-decompositions. We refer to Robertson and Seymour (1986) for more background. A *tree-decomposition* of an undirected graph is defined as follows.

Definition 2.2 (tree-decomposition). *A tree-decomposition of a graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *is a pair* $\langle T, \mathcal{X} \rangle$*, where* $T = (I, F)$ *is a tree and* $\mathcal{X} = \{\mathbf{X_i} \mid i \in I\}$ *is a family of subsets (called bags) of* $\mathbf{V}$*, one for each node of* T*, such that*

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

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

For every fixed number w, there is an algorithm using $\mathcal{O}(n)$ time, that, given a graph $\mathbf{G}$ with n nodes, decides if the treewidth of $\mathbf{G}$ is at most w and, if so, constructs a tree-decomposition of $\mathbf{G}$ with width w (Bodlaender, 1996). While this linear algorithm is not practically feasible due to its very large constant, efficient heuristics and exact algorithms for small values of n are known; see e.g. Bodlaender (2006).

A so-called *nice* tree-decomposition is a tree-decomposition with a particularly simple structure. In particular, a nice tree is rooted and every node in the tree has at most two children. A node in a nice tree-decomposition is either a leaf node, an insert node, a forget node, or a join node.

- A leaf node i is a leaf of T with $\mid \mathbf{X_i} \mid = 1$.
- An insert node i has one child j with $\mathbf{X_i} = \mathbf{X_j} \cup \{Y\}$, where $Y \in \mathbf{V} \setminus \mathbf{X_j}$.
- A forget node i has one child j with $\mathbf{X_i} = \mathbf{X_j} \setminus \{Y\}$ where $Y \in \mathbf{X_j}$.
- A join node i has two children j, k where $\mathbf{X_i} = \mathbf{X_j} = \mathbf{X_k}$.

Every tree-decomposition T with treewidth w and b bags can be converted to a nice tree-decomposition of the same width and $\mathcal{O}(w \cdot b)$ bags in time $\mathcal{O}(f(w) \cdot b)$ for a polynomial function f (Bodlaender, 1997; Kloks, 1994).

2.3 Probabilistic Networks

In this section, a brief review of the basic concepts of probabilistic networks is given. The interested reader is referred to textbooks like Pearl (1988) or Jensen and Nielsen (2007) for a more thorough introduction.

A probabilistic network models a set of stochastic variables, the (in-)dependencies among these variables, and a joint probability distribution over these variables. The variables and (in-)dependencies are modelled in the network by a directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$. A node $X \in \mathbf{V}$ is called a parent of $Y \in \mathbf{V}$, and Y is called a child of X, if $(X, Y) \in \mathbf{A}$. The set of all parents of Y is denoted as $\pi(Y)$, the set of all children of X is denoted as $\sigma(X)$. The transitive and reflexive closure of the set of parents of Y, denoted by $\pi^*(Y)$, is called the set of *ancestors* of Y; the transitive and reflexive closure of the set of children of Y, denoted by $\sigma^*(X)$, is called the set of *descendants* of Y.

We associate with each node $X \in \mathbf{V}$ a finite set $\Omega(X)$, denoting the set of values that X can take. We will use upper case letters to denote individual nodes in the network, upper case bold letters to denote sets of nodes, lower case letters to denote value assignments to nodes, and lower case bold letters to denote joint value assignments to sets of nodes. If not specified otherwise, we will use indexed lower case letters $x_1, x_2, \ldots, x_n$ to denote the values of a node X, and assume that $\Omega(X)$ is ordered such that $x_i < x_j$ whenever $i < j$. When X is a binary node, we will use x and $\bar{x}$ to denote the two values of x and assume that $\bar{x} < x$. We will use the notation $X = x$, respectively $\mathbf{X} = \mathbf{x}$, to denote a specific yet unspecified (joint) value assignment to X and $\mathbf{X}$, respectively. The set of all joint values that $\mathbf{X}$ can take will be denoted with $\Omega(\mathbf{X})$. We will use $\mathbf{C}$ to denote the set of output or classification nodes in the network. Likewise, we will use $\mathbf{E}$ to denote the set of evidence nodes, i.e., the set of nodes for which a particular joint value assignment is observed.

A *joint probability distribution* over $\mathbf{V}$ is a function on a Boolean algebra of propositions spanned by $\mathbf{V}$, defined in Van der Gaag (1990, p.85) as follows.

Definition 2.3 (joint probability distribution). *Let ξ denote a Boolean algebra of propositions spanned by $\mathbf{V}$. The function $\Pr : \xi \rightarrow [0, 1]$ is a joint probability distribution on $\mathbf{V}$ if the following conditions hold:*

- $0 \leq \Pr(a) \leq 1$, *for all* $a \in \xi$;
- $\Pr(\text{TRUE}) = 1$;
- $\Pr(\text{FALSE}) = 0$;
- *for all* $a, b \in \xi$, *if* $a \wedge b \equiv \text{FALSE}$ *then* $\Pr(a \vee b) = \Pr(a) + \Pr(b)$.

The *probability of* $X = x$ will be written as $\Pr(X = x)$; we will write $\Pr(x)$ if no ambiguity can occur. The *conditional* probability of x given a joint value assignment $\mathbf{y}$ to a set $\mathbf{Y}$, written as $\Pr(x \,|\, \mathbf{y})$, is defined as $\frac{\Pr(x,\mathbf{y})}{\Pr(\mathbf{y})}$ (provided that $\Pr(\mathbf{y}) > 0$). If $\Pr(\mathbf{x}, \mathbf{y}) = \Pr(\mathbf{x}) \cdot \Pr(\mathbf{y})$ then $\mathbf{x}$ and $\mathbf{y}$ are called independent in the joint probability distribution Pr. The joint value assignments $\mathbf{x}$ and $\mathbf{y}$ are called *conditionally independent* given $\mathbf{z}$ if $\Pr(\mathbf{x}, \mathbf{y} \,|\, \mathbf{z}) = \Pr(\mathbf{x} \,|\, \mathbf{z}) \cdot \Pr(\mathbf{y} \,|\, \mathbf{z})$. The *cumulative distribution function* F for a node $X \in \mathbf{V}$ is defined by $F(x) = \Pr(X \leq x)$ for all $x \in \Omega(X)$.

Some useful properties of probability distributions are the chain rule, marginalisation property, conditioning property, and Bayes' rule, stated below. Let Pr be a joint probability distribution on $\mathbf{V} = \{X_1, \ldots, X_n\}, n \geq 1$, let $X \in \mathbf{V}$ and let $\mathbf{Y} \subseteq \mathbf{V}$. We have that:

- $\Pr(x_1, \ldots, x_n) = \Pr(x_n \,|\, x_1, \ldots, x_{n-1}) \cdot \ldots \cdot \Pr(x_2 \,|\, x_1) \cdot \Pr(x_1)$ (chain rule).
- $\Pr(\mathbf{y}) = \sum_{x_i \in \Omega(X)} \Pr(\mathbf{y} \wedge x_i)$ (marginalisation).
- $\Pr(\mathbf{y}) = \sum_{x_i \in \Omega(X)} \Pr(\mathbf{y} \,|\, x_i) \cdot \Pr(x_i)$ (conditioning).
- $\Pr(\mathbf{y} \,|\, \mathbf{x}) = \frac{\Pr(\mathbf{x}|\mathbf{y}) \cdot \Pr(\mathbf{y})}{\Pr(\mathbf{x})}$, provided that $\Pr(\mathbf{y}) > 0$ and $\Pr(\mathbf{x}) > 0$ (Bayes' rule).

A probabilistic network is formally defined as follows.

Definition 2.4 (probabilistic network). *A probabilistic network, denoted by* $\mathcal{B}$, *is a tuple* $(\mathbf{G}, \Gamma)$, *where* $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ *is a directed acyclic graph, and* $\Gamma = \{\Pr_X \mid X \in \mathbf{V}\}$ *denotes a set of conditional probability distributions* $\Pr(X \,|\, \mathbf{y})$ *for each joint value assignment* $\mathbf{y}$ *to the parents of* X *in* $\mathbf{G}$.

A probabilistic network uniquely defines a joint probability distribution $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(X_i \mid \pi(X_i))$ over its variables (Kiiveri et al., 1984). In a probabilistic network, the structure of the graph encodes (in-)dependencies between the variables. Conditional (in-)dependence between two sets of variables $\mathbf{X}$ and $\mathbf{Y}$, given a third set $\mathbf{Z}$, is captured by the notions of *d-separation* and *d-connection*. Consider any probabilistic network $\mathcal{B} = (\mathbf{G}, \Gamma)$.

Definition 2.5 (d-separation). *Let a trail t from X to Y be defined as a simple path from X to Y in the underlying undirected graph, and let $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ be disjunct subsets of $\mathbf{V}$. We call $\mathbf{X}$ and $\mathbf{Y}$ d-separated given $\mathbf{Z}$ if all trails from any $X \in \mathbf{X}$ to any $Y \in \mathbf{Y}$ are blocked by $\mathbf{Z}$. A trail t from X to Y is blocked by $\mathbf{Z}$ if at least one of the following situations occurs:*

- *t contains $\{(P, Z), (Z, Q)\}$ (called a* chain*) for some $Z \in \mathbf{Z}$;*
- *t contains (Z, P) and (Z, Q) (called a* fork*) for some $Z \in \mathbf{Z}$;*
- *t contains (P, M) and (Q, M) (called a* collider*) for some M that is* not *in $\mathbf{Z}$, nor has any descendent in $\mathbf{Z}$.*

If two sets $\mathbf{X}$ and $\mathbf{Y}$ are not d-separated given $\mathbf{Z}$, then there are d-connected.

If $\mathbf{X}$ and $\mathbf{Y}$ are d-separated given $\mathbf{Z}$, then they are taken to be conditionally independent given $\mathbf{Z}$.

The size of a probabilistic network, denoted by $\|\mathcal{B}\|$, is defined as the number of bits needed to encode $\mathcal{B}$ by a reasonable encoding. The complexity of a probabilistic network is usually measured by the treewidth of a triangulation of the moralised graph of the network. Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a probabilistic network, where $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ is a directed acyclic graph with the set of vertices $\mathbf{V} = \{V_1, \ldots, V_n\}$ and the set of arcs $\mathbf{A}$. The *moralisation* $\mathbf{G_M}$ of $\mathbf{G}$ is the undirected graph obtained from $\mathbf{G}$ by adding arcs connecting all pairs of parents of a variable, and then dropping all arc directions. A *triangulation* of $\mathbf{G_M}$ is any graph $\mathbf{G_T}$, such that $\mathbf{E}(\mathbf{G_M}) \subseteq \mathbf{E}(\mathbf{G_T})$ and $\mathbf{G_T}$ is *chordal*, i.e. every cycle in $\mathbf{G_T}$ of more than three nodes has an edge connecting two non-adjacent nodes in the cycle. The treewidth of this chordal graph, denoted as $\mathrm{tw}(\mathbf{G_T})$, is the size of the largest clique in $\mathbf{G_T}$ minus one.

Finally, a word on the representation of numerical values. To avoid problems related to the finite representation and manipulation of real numbers (see e.g. Bodlaender et al. (2002)), we assume that all computations are done on rationals. We assume that all parameter probabilities in our network are rational numbers, thus ensuring that all calculated probabilities are rational numbers as well. This is a realistic assumption, since the probabilities are normally either assessed by domain experts or estimated by a learning algorithm from data instances. For similar reasons, we assume that functions like $\ln(x)$, when necessary, are approximated within a finite precision, polynomial in the length of the encoding of x. Furthermore, we assume that the conditional probabilities in the network are given explicitly in look-up tables (conditional probability tables or CPTs), rather than using some computable function. In our hardness proofs, moreover, ϵ is a particular sufficiently small value that can yet be encoded using space polynomial in the size of the input; we fix $\epsilon = \frac{1}{2^{n^2}}$ which is sufficiently small for our means, yet can be encoded using a number of bits polynomial in n.

2.4 Inference in probabilistic networks

Arguably the most important computational problem related to probabilistic networks, is determining the posterior probability distribution $\Pr(\mathbf{C} \mid \mathbf{e})$ of some set of variables $\mathbf{C}$ given evidence $\mathbf{e}$ for other variables in the network. The probability distribution of any variable can in essence be calculated using the chain rule, marginalisation, and conditioning. However, this calculation can take a time which is exponential in the size of the network, since $\Pr(\mathbf{C}) = \sum_{\mathbf{y} \in \Omega(\mathbf{Y})} \Pr(\mathbf{C} \wedge \mathbf{y})$. Typical algorithms for probabilistic inference are the *message passing* (Pearl, 1988), *variable elimination* (Dechter, 1998), and *clique tree propagation*[2] (Lauritzen and Spiegelhalter, 1988) approaches. When the network structure is restricted to be a polytree (or *singly connected* graph), inference can be done in polynomial time, for example using the message passing algorithm (Pearl, 1988). However, for multiply connected graphs, no polynomial-time algorithm is known.

One approach to inference in multiply connected graphs is enhancing the message passing algorithm with loop cutset conditioning. A *loop cutset* of a graph $\mathbf{G}$ is a set S such that for all simple loops in $\mathbf{G}$, S contains a vertex that is on

[2] Also called *join tree* propagation.

that loop and is not a collider. In loop cutset conditioning, a loop cutset of $\mathbf{G}$ is determined, and Pearl's (1988) message passing algorithm for singly connected graphs is applied, conditioned on each possible joint value assignment for that loop cutset. This algorithm is exponential in the size of the loop cutset of $\mathbf{G}$. As a second approach, variable elimination algorithms try to minimise the number of computations needed to compute posterior probabilities in the network by determining the optimal order of marginalisation. Clique tree propagation algorithms use yet another approach, and transform a multiply connected network into an equivalent singly connected network by clustering nodes. The latter algorithms are exponential in the treewidth of the moralised graph of $\mathbf{G}$. In practical applications (e.g., applications built with the Hugin (Jensen et al., 1990), Genie (Druzdzel, 1999), or Dazzle (Schrage et al., 2005) toolboxes) typically the clique tree propagation algorithm or one of its variants (e.g. the Shenoy-Shafer (Shenoy and Shafer, 1986) algorithm) is used.

No polynomial-time algorithms are known for the inference problem in general, and indeed various problem variants (discussed below) have been shown to be NP-complete (Cooper, 1990), PP-complete (Littman et al., 2001) and #P-complete (Roth, 1996), respectively. The construction needed to prove hardness is essentially the same for all three problems and uses a reduction from the corresponding canonical Satisfiability-variant. Since we will use this construction throughout this thesis, and the published proofs are sometimes brief or based on other reductions[3], we will review these hardness proofs in detail here. We will first give the relevant problem definitions. For historical reasons, we refer to the PP-complete variant when we discuss Inference in general.

Positive Inference
Instance: Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a probabilistic network, and let Pr be its joint probability distribution. Let $\mathbf{C} \subseteq \mathbf{V}$ denote a set of output variables with joint value assignment $\mathbf{c}$, and let $\mathbf{E} \subseteq \mathbf{V}$ denote a set of evidence variables with joint value assignment $\mathbf{e}$.
Question: Is $\Pr(\mathbf{c} \mid \mathbf{e}) > 0$?

[3]Cooper's (1990) proof uses a reduction from Vertex Cover; Littman et al. (2001) asserts PP-completeness of the decision variant of the inference problem as a corollary from the #P-completeness; Roth's (1996) proof uses a reduction from Monotone 2SAT to preserve approximation results; furthermore the problem is "strictly ... not in #P" (Roth, 1996, p.280), but "easily seen to be equivalent to a problem in this class."

Inference
Instance: Let $\mathbf{c}$ and $\mathbf{e}$ be given as in Positive Inference. Furthermore, let $0 \leq q \leq 1$.
Question: Is $\Pr(\mathbf{c} \mid \mathbf{e}) > q$?

Exact Inference
Instance: Let $\mathbf{c}$ and $\mathbf{e}$ be given as in Positive Inference.
Output: The probability $\Pr(\mathbf{c} \mid \mathbf{e})$.

Note that the first two problems are decision problems and that the last one is a function problem, i.e., a function that can be computed by a Turing Transducer. We will first discuss membership of NP, PP, and #P, respectively, for these problems.

Lemma 2.6. Positive Inference *is in* NP.

Proof. To prove membership in NP, it suffices to give a polynomial-time verifiable certificate of membership. Let $\{Y_1, \ldots, Y_n\}$ denote the variables in $\mathbf{V} \setminus \{\mathbf{C} \cup \mathbf{E}\}$. Recall from Section 2.3 that $\Pr(\mathbf{c}, \mathbf{e}) = \sum_{y_1} \ldots \sum_{y_n} \Pr(Y_1 = y_1, \ldots, Y_n = y_n, \mathbf{c}, \mathbf{e})$ according to the marginalisation property. To show that $\Pr(\mathbf{c} \mid \mathbf{e}) > 0$ we need to show that there exists at least one term $\Pr(y_1, \ldots, y_n, \mathbf{c}, \mathbf{e}) > 0$. A certificate for this purpose consists of the corresponding entries in the CPTs, and can trivially be verified in polynomial time. □

Lemma 2.7. Inference *is in* PP.

Proof. To prove membership in PP, we need to show that Inference can be decided by a Probabilistic Turing Machine $\mathcal{M}$ in polynomial time. To facilitate our proof, we first show how to compute $\Pr(\mathbf{c})$ probabilistically; for brevity we assume no evidence, the proof with evidence goes analogously. $\mathcal{M}$ computes a joint probability $\Pr(y_1, \ldots, y_n)$ by iterating over i using a topological sort of the graph, and choosing a value for each variable Y_i conform the probability distribution in its CPT given the values that are already assigned to the parents of Y_i. Each computation path then corresponds to a specific joint value assignment to the variables in the network, and the probability of arriving in a particular state corresponds with the probability of that assignment. After iteration, we

accept with probability $\frac{1}{2} + (1-q) \cdot \epsilon$, if the joint value assignment to $Y_1, \ldots, Y_n$ is consistent with $\mathbf{c}$, and we accept with probability $\frac{1}{2} - q \cdot \epsilon$ if the joint value assignment is *not* consistent with $\mathbf{c}$. The probability of entering an accepting state is hence $\Pr(\mathbf{c}) \cdot (\frac{1}{2} + (1-q)\epsilon) + (1 - \Pr(\mathbf{c})) \cdot (\frac{1}{2} - q \cdot \epsilon) = \frac{1}{2} + \Pr(\mathbf{c}) \cdot \epsilon - q \cdot \epsilon$. Now, the probability of arriving in an accepting state is strictly larger than $\frac{1}{2}$ if and only if $\Pr(\mathbf{c}) > q$. □

For Exact Inference, showing membership in #P is problematic. Recall that #P is defined as the class of counting problems which have a decision variant in NP; a problem is in #P if it computes the number of accepting paths on a particular TM given an input x. Since Exact Inference is technically not a counting problem[4], we believe that Roth's (1996) claim that Exact Inference is #P-complete is actually too strong.

To prove hardness results for these three problems, we will use (here and in the remaining chapters) a proof technique due to Park and Darwiche (2004). Using their technique, Park and Darwiche proved that the decision variant of the problem of finding the most likely joint value assignment to a set of variables given *partial* evidence for the complement of that set (the Partial MAP problem), is $\mathsf{NP}^{\mathsf{PP}}$-complete. In the proof, a probabilistic network $\mathcal{B}_\phi$ is constructed from a given Boolean formula ϕ with n variables. For each propositional variable x_i in ϕ, a binary stochastic variable X_i is added to $\mathcal{B}_\phi$, with possible values true and false and a uniform probability distribution. For each logical operator in ϕ, an additional binary variable in $\mathcal{B}_\phi$ is introduced, whose parents are the variables that correspond to the input of the operator, and whose conditional probability table is equal to the truth table of that operator. For example, the value true of a stochastic variable mimicking the *and*-operator would have a conditional probability of 1 if and only if both its parents have the value true, and 0 otherwise. The top-level operator in ϕ is denoted as V_ϕ. In Figure 2.1 the network $\mathcal{B}_\phi$ is shown for the formula $\neg(x_1 \vee x_2) \wedge \neg x_3$.

Now, for any particular truth assignment $\mathbf{x}$ to the set of all propositional variables $\mathbf{X}$ in the formula ϕ we have that the probability of the value true of V_ϕ, given the joint value assignment to the stochastic variables matching that truth assignment, equals 1 if $\mathbf{x}$ satisfies ϕ, and 0 if $\mathbf{x}$ does not satisfy ϕ. Without any

[4] If we would be allowed to interpret 'counting' liberally to also include *weighted* counting, then membership would follow from the marginalisation property, since we sum over the certificates $\Pr(y_1, \ldots, y_n) > 0$ of the decision problem Positive Inference to compute $\Pr(X = x)$.

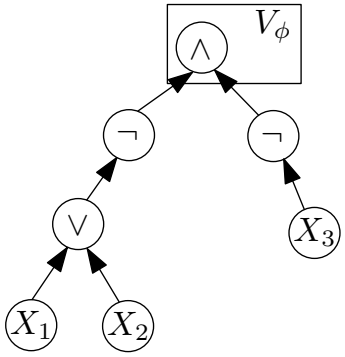


Figure 2.1 The probabilistic network corresponding to $\neg(x_1 \vee x_2) \wedge \neg x_3$

given joint value assignment, the prior probability of V_ϕ is $\frac{\#_\phi}{2^n}$, where $\#_\phi$ is the number of satisfying truth assignments of the set of propositional variables $\mathbf{X}$. Note that the above network $\mathcal{B}_\phi$ can be constructed from ϕ in polynomial time.

Lemma 2.8. Positive Inference *is* NP*-hard.*

Proof. We reduce Satisfiability to Positive Inference. Let ϕ be a Satisfiability-instance and let $\mathcal{B}_\phi$ be the network as constructed above. Now, $\Pr(V_\phi = \text{TRUE}) > 0$ if and only if there is a satisfying truth assignment to ϕ. □

Lemma 2.9. Inference *is* PP*-hard.*

Proof. We reduce Majsat to Inference. Let ϕ be a Majsat-instance and let $\mathcal{B}_\phi$ be the network as constructed above. Now, $\Pr(V_\phi = \text{TRUE}) > \frac{1}{2}$ if and only if the majority of truth assignments satisfy ϕ. □

Lemma 2.10. Exact Inference *is* #P*-hard.*

Proof. We reduce #Sat to Exact Inference, using a parsimoniously polynomial-time many-one reduction, i.e., a reduction that preserves the number of solutions. Let ϕ be a #Sat-instance and let $\mathcal{B}_\phi$ be the network as constructed above. Now, $\Pr(V_\phi = \text{TRUE}) = \frac{l}{2^n}$ if and only if l truth assignments satisfy ϕ. □

We will later use this technique to show hardness of certain natural problems related to the construction and use of probabilistic networks for classes like $\mathsf{NP}^{\mathsf{PP}}$, $\mathsf{co\text{-}NP}^{\mathsf{PP}}$, and P^{PP}.

Part I

Analysis

Chapter 3

Sensitivity Analysis and Parameter Tuning

In this chapter, we discuss sensitivity analysis and parameter tuning in probabilistic networks. The sensitivity of the output of a probabilistic network to small changes in its parameters has been studied by various researchers (Castillo et al., 1997; Laskey, 1995; Coupé and van der Gaag, 2002; Chan and Darwiche, 2004; van der Gaag and Renooij, 2001; Charitos and van der Gaag, 2006). Whether the parameter probabilities of a network are assessed by domain experts or estimated from data, they inevitably include some inaccuracies. In a sensitivity analysis of the network, the parameter probabilities are varied within a plausible range and the effect of the variation is studied on the output computed from the network, be it a posterior probability or the most likely value of an output variable.

The results of a sensitivity analysis are used, for example, to establish the robustness of the network's output. The results are used also in engineering a probabilistic network, for example to distinguish between parameters which allow some imprecision and parameters which should be determined as accurately as possible (Coupé, Jensen, Kjærulff and van der Gaag, 2000). Another use is in carefully *tuning* the parameter probabilities of a network to arrive at some desired model behaviour (Chan and Darwiche, 2005).

Research efforts in sensitivity analysis and parameter tuning for probabilistic networks have resulted in a variety of fundamental insights and computational methods. While the majority of these insights and methods pertain to a one-way sensitivity analysis in which the effect of varying a single parameter probability on a single output probability or output value is studied, recently there also has been some pioneering work on extending these insights to higher-order analyses (Chan and Darwiche, 2004; Coupé, Jensen, Kjærulff and van der Gaag, 2000).

The currently available algorithms for sensitivity analysis and parameter tuning of probabilistic networks have a running time that is exponential in the size of the network and in the number of CPTs from which the parameters are taken. This observation suggests that these problems could be intractable in general. The actual computational complexity of the problems, however, has not been studied until now. In this chapter we define several variants of the tuning problem for probabilistic networks and show that these variants are all $\mathsf{NP}^{\mathsf{PP}}$-complete in general. We further show that the tuning problem remains NP-complete even if the topological structure of the network under study is restricted to a polytree, and is still PP-complete if both the number of conditional probability tables involved and the indegree of the corresponding nodes are bounded by a constant.

Given the unfavourable complexity results, even for restricted cases of the tuning problem, we cannot expect to arrive at efficient, general computational methods for sensitivity analysis and parameter tuning for probabilistic networks. Our complexity results in fact suggest that further research should concentrate on tuning a limited number of parameters in networks where inference is tractable.

The chapter is organised as follows. After briefly reviewing the basic concepts involved in sensitivity analysis and parameter tuning in Section 3.1, we formally define several variants of the tuning problem in Section 3.2. We give a general completeness proof for these problems in Section 3.3. We further address some

special, restricted cases of these problems in Section 3.4. The chapter ends with some concluding observations in Section 3.5.

3.1 Sensitivity analysis and tuning

Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a probabilistic network as defined in Chapter 2. In sensitivity analysis and parameter tuning, we are interested in the behaviour of the output of $\mathcal{B}$ upon varying its *parameters*. We will denote a particular set of parameter probabilities as $\mathbf{O} \subseteq \Gamma$, and use O to denote a single parameter[1]. We will use $\mathbf{o}$ and o to denote a *combination of values* of the set of parameters $\mathbf{O}$ and a value of a single parameter O, respectively. We denote with $\Pr_{\mathbf{o}}$ the joint probability distribution defined by the network $\mathcal{B}$, in which the parameters $\mathbf{O}$ have been assigned the combination of values $\mathbf{o}$. In sensitivity analysis and parameter tuning, we are interested in the effect of changes in the values of parameters $\mathbf{O}$ on an output probability for a designated variable C, given evidence $\mathbf{e}$. The *sensitivity function* $f_{\Pr(c|\mathbf{e})}(\mathbf{O})$ expresses the conditional probability of the output for the value c of C in terms of the parameter set $\mathbf{O}$. We will omit the subscript if no ambiguity can occur.

3.1.1 Sensitivity analysis

In a *one-way sensitivity analysis*, we measure the sensitivity of an output probability of interest with respect to a single parameter. The parameter under consideration is systematically varied from 0 to 1 and the other parameters from the same CPT are co-varied such that their mutual proportional relationship is kept constant (van der Gaag et al., 2007). Thus, if the parameter $O = \Pr(b_i \,|\, \mathbf{a})$ (denoting the conditional probability of the value b_i of the variable B given a particular configuration $\mathbf{a}$ of B's parents) is varied from 0 to 1, the other

[1] In the literature on sensitivity analysis, typically X is used to denote parameters. Throughout this thesis, X is used as a network node modelling a Boolean variable. To avoid confusion between variables, probabilities, and parameters, we will not use the (perhaps more obvious) letters P or X to denote parameters.

parameters $\Pr(b_j \,|\, \mathbf{a})$ for the variable B are varied such that

$$\Pr(b_j \,|\, \mathbf{a})(O) = \Pr(b_j \,|\, \mathbf{a}) \cdot \frac{1 - O}{1 - \Pr(b_i \,|\, \mathbf{a})}$$

for any value b_j other than b_i. Under the condition of co-variation, the sensitivity function $f(O)$ is a quotient of two linear functions (Coupé and van der Gaag, 2002), that is, it takes the form

$$f(O) = \frac{c_1 \cdot O + c_2}{c_3 \cdot O + c_4}$$

where the constants $c_1, \ldots, c_4$ can be calculated from the other parameter probabilities in the network. Note that $f(O)$ reaches its maximum either at $O = 0$ or at $O = 1$.

A one-way sensitivity analysis can be extended to measure the effect of the simultaneous variation of *two* parameters on the output (Coupé, van der Gaag and Habbema, 2000). The sensitivity function then generalises to

$$f(O_1, O_2) = \frac{c_5 \cdot O_1 \cdot O_2 + c_6 \cdot O_1 + c_7 \cdot O_2 + c_8}{c_9 \cdot O_1 \cdot O_2 + c_{10} \cdot O_1 + c_{11} \cdot O_2 + c_{12}}$$

In this function, the terms $c_5 \cdot O_1 \cdot O_2$ and $c_9 \cdot O_1 \cdot O_2$ capture the interaction effect of the parameters on the output variable. This can be further generalised to *n-way* sensitivity analyses (Coupé, Jensen, Kjærulff and van der Gaag, 2000; Chan and Darwiche, 2004) where multiple parameters are varied simultaneously. While higher-order analyses can reveal synergistic effects of variation, the results are often difficult to interpret (van der Gaag et al., 2007).

For performing a one-way sensitivity analysis, efficient algorithms are available that build on the observation that for establishing the sensitivity of an output probability it suffices to determine the constants in the associated sensitivity function. The simplest method for this purpose is to compute, from the network, the probability of interest for up to three values for the parameter under study; using the functional form of the function to be established, a system of linear equations is obtained, which is subsequently solved (Coupé and van der Gaag, 2002). For the network computations involved, any standard propagation algorithm can be used. A more efficient method determines the required

constants by propagating information through a junction tree, similar to the standard junction-tree propagation algorithm discussed in Chapter 2 (Kjærulff and van der Gaag, 2002). This method requires a very small number of inward and outward propagations in the tree to determine the constants of all sensitivity functions that relate the probability of interest to any one of the network parameters, or to determine the sensitivity functions for any output probability in terms of a single parameter. Both algorithms are exponential in the size of the network, yet have a polynomial running time for networks of bounded treewidth.

3.1.2 Parameter tuning

Closely related to *analysing* the effect of variation of parameters on the output—and often the next step after performing such an analysis—is *tuning* the parameters, such that the output has the desired properties. The output may need to satisfy particular constraints, e.g. $\Pr(c \mid \mathbf{e}) \geq q$, $\Pr(c_1 \mid \mathbf{e}) / \Pr(c_2 \mid \mathbf{e}) \geq q$ or $\Pr(c_1 \mid \mathbf{e}) - \Pr(c_2 \mid \mathbf{e}) \geq q$, for a particular value q. There are a number of algorithms to determine the solution space for a set of parameters given such constraints (Chan and Darwiche, 2004). The computational complexity of these algorithms is always exponential in the treewidth w of the moralised graph (i.e., the size of the largest clique in the junction tree minus one), yet varies from $\mathcal{O}(c^w)$ for single parameter tuning, to $\mathcal{O}(n \cdot \prod_{i=1}^{k} F(O_i) \cdot c^w)$ for tuning n parameters, where c is a constant, k is the number of CPTs that include at least one of the parameters being varied, and $F(O_i)$ denotes the size of the i-th CPT. Note that the tuning problem is somewhat related to the inference problem in so-called *credal networks* (Cozman, 2000), where each variable is associated with sets of probability measures, rather than single probabilities as in Bayesian networks. Computing upper (or lower) bounds on the probability distribution of a variable has been proven $\mathsf{NP}^{\mathsf{PP}}$-complete (Cozman et al., 2004; de Campos and Cozman, 2005). In some sense, computing these bounds can be envisaged by non-deterministically selecting a fixed value within a credal set, and then using inference to verify that upper bound.

3.1.3 Distance measures

Often, we want to select a combination of values for the parameters that satisfies constraints on the output probability of interest, yet has minimal impact on the other probabilities computed from the network. In other cases, we want the change in the original values to be as small as possible. In other words, we want to find a tuning that not merely satisfies the constraints, but is also *optimal*, either with respect to the amount of parameter change needed or with respect to the change in the joint probability distribution induced by the parameter change. Here we review two typical distance measures for joint probability distributions, namely those proposed by Kullback and Leibler (1951), and, more recently, by Chan and Darwiche (2005), respectively.

The distance measure introduced by Chan and Darwiche (2005), denoted by D_{CD}, defines the distance between two joint probability distributions $\Pr_{\mathbf{o}}$ and $\Pr_{\mathbf{o}'}$ as:

$$D_{CD}(\Pr_{\mathbf{o}}, \Pr_{\mathbf{o}'}) \stackrel{def}{=} \ln \max_{\omega} \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)} - \ln \min_{\omega} \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)}$$

where ω ranges over the joint value assignments to the variables in the network. The Kullback-Leibler measure (Kullback and Leibler, 1951), denoted by D_{KL}, is defined as:

$$D_{KL}(\Pr_{\mathbf{o}}, \Pr_{\mathbf{o}'}) \stackrel{def}{=} \sum_{\omega} \Pr_{\mathbf{o}}(\omega) \ln \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)}$$

Both distance measures are undefined if $\Pr_{\mathbf{o}'}(\omega) = 0$ and $\Pr_{\mathbf{o}}(\omega) \neq 0$. If $\Pr_{\mathbf{o}'}(\omega) = \Pr_{\mathbf{o}}(\omega) = 0$, then $\ln \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)} = 0$ by definition. Calculating either distance between two distributions is intractable in general. Calculating D_{CD} has an NP-complete decision variant, and calculating D_{KL} has a PP-complete decision variant; proofs of these claims can be found in the appendix.

The *Euclidean distance* is a convenient way to measure the *amount of change* needed in $\mathbf{o}$ to go from $\Pr_{\mathbf{o}}$ to $\Pr_{\mathbf{o}'}$. This distance, denoted by D_E, is defined as:

$$D_E(\mathbf{o}, \mathbf{o}') \stackrel{def}{=} \sqrt{\sum_{o_i \in \mathbf{o}, o_i' \in \mathbf{o}'} (o_i - o_i')^2}$$

The Euclidean distance depends only on the parameters that are changed and can be calculated in $\mathcal{O}(|\,\mathbf{O}\,|)$ steps, assuming that taking the square root of a number takes a polynomial number of steps.

3.2 Problem definitions

In the previous section, we have encountered a number of computational problems related to sensitivity analysis and parameter tuning. To prove hardness results, we will first define decision problems related to these questions. These decision problems capture various problem variations. Because of the formulation as decision problems, all problems are in fact tuning problems. We distinguish between tuning to satisfy constraints on the probability distribution of an output variable C given evidence $\mathbf{e}$, and on the mode (i.e., the most likely value, or if no unique most likely value exists the most likely value that has the lowest order in $\Omega(C)$).

Parameter Tuning
Instance: Probabilistic network $\mathcal{B} = (\mathbf{G}, \Gamma)$, where $\mathbf{O} \subseteq \Gamma$ denotes a set of parameters in the network. Let C denote the output variable and let c be a particular value of C. Furthermore, let $\mathbf{E}$ denote a set of evidence variables with the joint value assignment $\mathbf{e}$, and let $0 \leq q \leq 1$.
Question: Is there a combination of values $\mathbf{o}$ for the parameters in $\mathbf{O}$ such that $\Pr_{\mathbf{o}}(c \mid \mathbf{e}) \geq q$?

Parameter Tuning Range
Instance: As in Parameter Tuning.
Question: Are there combinations of values $\mathbf{o}$ and $\mathbf{o}'$ for the parameters in $\mathbf{O}$ such that $\Pr_{\mathbf{o}}(c \mid \mathbf{e}) - \Pr_{\mathbf{o}'}(c \mid \mathbf{e}) \geq q$?

Evidence Parameter Tuning Range
Instance: As in Parameter Tuning; furthermore let $\mathbf{e_1}$ and $\mathbf{e_2}$ denote two particular joint value assignments to the evidence variables in $\mathbf{E}$.
Question: Is there a combination of values $\mathbf{o}$ for the parameters in $\mathbf{O}$ such that $\Pr_{\mathbf{o}}(c \mid \mathbf{e_1}) - \Pr_{\mathbf{o}}(c \mid \mathbf{e_2}) \geq q$?

Minimal Parameter Tuning Range
Instance: As in Parameter Tuning; furthermore let $r \in \mathbb{Q}^+$.
Question: Are there combinations of values $\mathbf{o}$ and $\mathbf{o}'$ for the parameters in $\mathbf{O}$ such that $D_E(\mathbf{o}, \mathbf{o}') \leq r$ and $\Pr_{\mathbf{o}}(c \mid \mathbf{e}) - \Pr_{\mathbf{o}'}(c \mid \mathbf{e}) \geq q$?

Minimal Change Parameter Tuning Range
Instance: As in Parameter Tuning; furthermore let $s \in \mathbb{Q}^+$, and let D denote a distance measure for joint probability distributions as reviewed in Section 3.1.
Question: Are there combinations of values $\mathbf{o}$ and $\mathbf{o}'$ for the parameters in $\mathbf{O}$ such that $D(\mathbf{o}, \mathbf{o}') \leq s$ and $\Pr_{\mathbf{o}}(c \mid \mathbf{e}) - \Pr_{\mathbf{o}'}(c \mid \mathbf{e}) \geq q$?

Mode Tuning
Instance: As in Parameter Tuning; furthermore let $\top(\Pr(C \mid \mathbf{e}))$ denote the mode of $\Pr(C \mid \mathbf{e})$.
Question: Are there combinations of values $\mathbf{o}$ and $\mathbf{o}'$ for the parameters in $\mathbf{O}$ such that $\top(\Pr_{\mathbf{o}}(C \mid \mathbf{e})) \neq \top(\Pr_{\mathbf{o}'}(C \mid \mathbf{e}))$?

We define Evidence Mode Tuning, Minimal Parameter Mode Tuning, and Minimal Change Mode Tuning similar to the Parameter Tuning variants mentioned above, but now defined in terms of the mode.

Note that trivial solutions to Parameter Tuning and the other problem variants may exist for particular subsets $\mathbf{O}$ of the set of parameter probabilities Γ. For example, if $\mathbf{O}$ consists of all conditional probabilities $\Pr(C = c \mid \pi(C))$, for all joint value assignments $\pi(C)$ to the parents of C, then a trivial solution would set all these parameters to q. If the number of parameters in $\mathbf{O}$ is logarithmic in the total number of parameter probabilities, i.e., if $\mid \mathbf{O} \mid \leq p(\log \mid \Gamma \mid)$ for some polynomial function p, then all problem variants are in $\mathsf{PP}^{\mathsf{PP}}$, since we can try all combinations of parameter settings to 0 or 1 in polynomial time, using a PP-oracle. Recall that the sensitivity function is maximal when $o = 0$ or $o = 1$ for every parameter o, therefore it suffices to try all possible relevant combinations (i.e., $o = 0$ or $o = 1$) to determine whether one of these combinations of values satisfies the constraints on the probability or the mode of the output variable of interest. This can be done in polynomial time.

3.3 Completeness results

To show $\mathsf{NP}^{\mathsf{PP}}$-completeness of Parameter Tuning Range, we will first show that Parameter Tuning Range is $\mathsf{NP}^{\mathsf{PP}}$-hard. The $\mathsf{NP}^{\mathsf{PP}}$-hardness of the other problems can be derived with minimal changes to the proof construction. We prove $\mathsf{NP}^{\mathsf{PP}}$-hardness of Parameter Tuning Range by a reduction from E-Majsat, using the technique of Park and Darwiche discussed in Chapter 2. Recall the definition of E-Majsat from Chapter 2:

E-Majsat
Instance: Let ϕ be a Boolean formula with n variables $x_i, i = 1, \ldots, n, n \geq 1$, grouped into two disjoint sets $\mathbf{X_E} = \{x_1, \ldots, x_k\}$ and $\mathbf{X_M} = \{x_{k+1}, \ldots, x_n\}$ for some $k \leq n$.
Question: Is there a truth assignment to $\mathbf{X_E}$ such that with the majority of the truth assignments to $\mathbf{X_M}$, ϕ is satisfied?

We construct a probabilistic network $\mathcal{B}_\phi$ from the Boolean formula ϕ of an E-Majsat instance. For each variable x_i in the formula ϕ, we create a matching node X_i in $\mathbf{V}$ for the network $\mathcal{B}_\phi$, with possible values true and false with a uniform prior distribution. These nodes are roots in the network $\mathcal{B}_\phi$. We use $O_i = \Pr(X_i = \text{true})$ to denote the *parameter* of X_i. For each logical operator in ϕ, we create an additional node in the network, whose parents capture the corresponding sub-formulas (or a single sub-formula in case of a negation operator) and whose CPT is equal to the truth table of that operator. For example, a node W_i, with parents W_j and W_k, corresponding to the $\wedge$-operator would have a conditional probability $\Pr(W_i = \text{true} \mid W_j, W_k) = 1$ if and only if both W_j and W_k have the value true, and $\Pr(W_i = \text{true} \mid W_j, W_k) = 0$ otherwise. We denote the node that is associated with the top-level operator in ϕ with V_ϕ. Furthermore, we add a node S with values true and false with a uniform prior probability distribution, where $O_S = \Pr(S = \text{true})$ is the parameter of S. Lastly, we have an output node C, with parents S and V_ϕ and with values true and false, whose CPT is equal to the truth table of the $\wedge$-operator. The set of parameters $\mathbf{O}$ in the Parameter Tuning Range-problem is now defined to be $\{O_1, \ldots, O_k\} \cup \{O_S\}$, i.e., the parameters of the nodes in the set $\mathbf{X_E}$ of the E-Majsat instance and the parameter of S. We set $q = \frac{1}{2} + \frac{1}{2^n}$ and assume no evidence variables, i.e., $\mathbf{E} = \varnothing$.

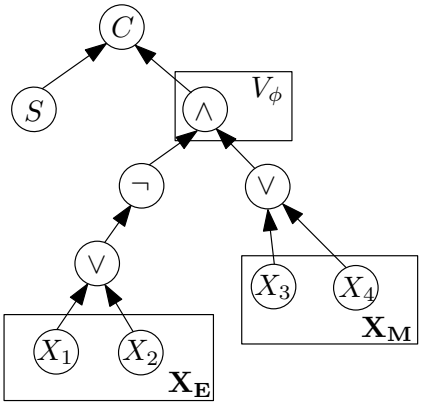


Figure 3.1 Example of construction of a probabilistic network $\mathcal{B}_{\text{ex}}$ for a Boolean formula ϕ_{ex}

Figure 3.1 shows the graphical structure of the probabilistic network constructed for the E-MAJSAT instance $(\phi_{\text{ex}}, \mathbf{X_E}, \mathbf{X_M})$, where $\phi_{\text{ex}} = \neg(x_1 \vee x_2) \wedge (x_3 \vee x_4)$, $\mathbf{X_E} = \{x_1, x_2\}$, and $\mathbf{X_M} = \{x_3, x_4\}$. Note that this E-MAJSAT instance is satisfiable with $x_1 = x_2 =$ FALSE. For that truth assignment to $\mathbf{X_E}$, strictly more than half of the possible truth assignments to $\mathbf{X_M}$ satisfy the formula; in fact, only $x_3 = x_4 =$ FALSE will fail to satisfy ϕ_{ex}.

Theorem 3.1. PARAMETER TUNING RANGE *is* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

Proof. Membership of $\mathsf{NP}^{\mathsf{PP}}$ can be proved as follows. Given non-deterministically chosen $\mathbf{o}'$ and $\mathbf{o}$, we can verify for any given value assignment c and joint value assignment $\mathbf{e}$ whether $\Pr_{\mathbf{o}}(c \mid \mathbf{e}) - \Pr_{\mathbf{o}'}(c \mid \mathbf{e}) \geq q$ in polynomial time, given an oracle that decides INFERENCE. Since INFERENCE is PP-complete, this proves membership of $\mathsf{NP}^{\mathsf{PP}}$.

To prove hardness, we construct a polynomial time transformation from E-MAJSAT. Let $(\phi, \mathbf{X_E}, \mathbf{X_M})$ be an instance of E-MAJSAT, and let $\mathcal{B}_\phi$ be the probabilistic network constructed from ϕ as described above. Let $\mathbf{O} = \{O_1, \ldots, O_k\} \cup \{O_S\}$. Further, let $q = \frac{1}{2} + \frac{1}{2^n}$ and let $\mathbf{E} = \varnothing$. Trivially, there exists a combination of parameter values $\mathbf{o}'$ such that $\Pr_{\mathbf{o}'}(C = \text{TRUE}) = 0$, namely all assignments in which $O_S = 0$: at least one of the parents of C then has the value FALSE with probability 1 and thus $\Pr_{\mathbf{o}'}(C = \text{TRUE}) = 0$. If $\mathbf{o}$ includes

$O_S = 1$, then $\Pr_{\mathbf{o}}(C = \text{TRUE})$ depends on the values of $O_1, \ldots, O_k$. More in particular, there exist values such that $\Pr_{\mathbf{o}}(C = \text{TRUE}) \geq \frac{1}{2} + \frac{1}{2^n}$ if and only if $(\phi, \mathbf{X_E}, \mathbf{X_M})$ has a solution. From a solution $\mathbf{x_e}$ to $(\phi, \mathbf{X_E}, \mathbf{X_M})$, we can construct a solution $\mathbf{o}$ by assigning the value 1 to O_S, the value 1 to all parameters in $\{O_1, \ldots, O_k\}$ for which the corresponding variable in $\mathbf{x_e}$ is set to TRUE and the value 0 where it is set to FALSE. Given this combination of parameter values, $\Pr_{\mathbf{o}}(V_\phi = \text{TRUE} \,|\, \mathbf{x_M}) = 1$ for every joint value assignment $\mathbf{x_M}$ to $\mathbf{X_M}$ that corresponds to a truth assignment that satisfies ϕ. Since the majority of the truth assignments to $\mathbf{X_M}$ satisfy ϕ, $\Pr_{\mathbf{o}}(V_\phi = \text{TRUE}) \geq \frac{1}{2} + \frac{1}{2^n}$. On the other hand, if $(\phi, \mathbf{X_E}, \mathbf{X_M})$ is *not* satisfiable, then $\Pr_{\mathbf{o}}(C = \text{TRUE})$ will be at most $\frac{1}{2}$ for *any* parameter setting. Due to the nature of the CPTs of the operator variables which mimic the truth tables of the operators, $\Pr_{\mathbf{o}}(C = \text{TRUE} \,|\, \mathbf{x_M}) = 1$ for a combination of values $\mathbf{o}$ for the parameters and a joint value assignment $\mathbf{x_M}$ that correspond with a satisfying truth assignment to ϕ. If there does not exist a truth assignment to the variables in $\mathbf{X_E}$ such that the majority of the truth assignments to the variables in $\mathbf{X_M}$ satisfies ϕ, then there cannot be a value assignment to $\mathbf{O}$ such that $\Pr_{\mathbf{o}}(C = \text{TRUE}) \geq \frac{1}{2} + \frac{1}{2^n}$.

Thus, if we can decide whether there exist two sets of parameter settings $\mathbf{o}$ and $\mathbf{o}'$ such that in this network $\mathcal{B}_\phi$, $\Pr_{\mathbf{o}}(C = \text{TRUE}) - \Pr_{\mathbf{o}'}(C = \text{TRUE}) \geq \frac{1}{2} + \frac{1}{2^n}$, then we can answer $(\phi, \mathbf{X_E}, \mathbf{X_M})$ as well. This reduces E-Majsat to Parameter Tuning Range. □

Note that the constructed proof shows that Parameter Tuning Range remains $\mathsf{NP}^{\mathsf{PP}}$-complete, even if we restrict the set of parameters to include only prior probabilities, if all variables are binary, if all nodes have indegree at most 2, if the output is a singleton variable, and if there is no evidence. We now show completeness results for the other problems.

Proposition 3.2. *All tuning problems defined in Section 3.2 are* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

Proof. We first observe that all problems are in $\mathsf{NP}^{\mathsf{PP}}$: given $\mathbf{o}$, $\mathbf{o}'$, q, r and s, we can verify all claims in polynomial time using a PP oracle, since inference is in PP and in fact PP-complete (Roth, 1996). For example, with the use of the oracle, we can verify in polynomial time whether $\Pr_{\mathbf{o}}(c \mid \mathbf{e}) - \Pr_{\mathbf{o}'}(c \mid \mathbf{e}) \geq q$, for a given $\mathbf{o}$, $\mathbf{o}'$, and q. Likewise, we can calculate the Euclidean distance of $\mathbf{o}$ and $\mathbf{o}'$ in polynomial time and verify that it is less than r. Deciding whether a distance between two joint probability distributions is smaller than s is NP-complete for

the distance D_{CD} defined by Chan and Darwiche (2005) and PP-complete for the distance D_{KL} defined by Kullback and Leibler (1951). Thus, we can non-deterministically choose a combination of values for $\mathbf{O}$ and check (using a PP oracle) that the distance is smaller than s. Therefore, all problems are in $\mathsf{NP}^{\mathsf{PP}}$.

We now show how the above construction can be adjusted to prove $\mathsf{NP}^{\mathsf{PP}}$-hardness for all other tuning problems defined in Section 3.2.

- Parameter Tuning: From the above construct, we leave out the nodes S and C, and let $\mathbf{O} = \{O_1, \ldots, O_k\}$. Then there is a combination of values $\mathbf{o}$ such that $\Pr_{\mathbf{o}}(V_\phi = \text{true}) \geq \frac{1}{2} + \frac{1}{2^n}$ if and only if $(\phi, \mathbf{X_E}, \mathbf{X_M})$ has a solution.

- Evidence Parameter Tuning Range: From the above construct, we replace S with a singleton evidence variable E with values true and false and a uniform prior distribution; we denote $E = \text{true}$ as e_1 and $E = \text{false}$ as e_2. Let $\mathbf{O} = \{O_1, \ldots, O_k\}$. Then, $\Pr_{\mathbf{o}}(C = \text{true} \mid E = e_2) = 0$ for all possible parameter settings $\mathbf{o}$. On the other hand, $\Pr_{\mathbf{o}}(V_\phi = \text{true}) \geq \frac{1}{2} + \frac{1}{2^n}$ and thus $\Pr_{\mathbf{o}}(C = \text{true} \mid E = e_1) \geq \frac{1}{2} + \frac{1}{2^n}$ if and only if $(\phi, \mathbf{X_E}, \mathbf{X_M})$ has a solution.

- Minimal Parameter Tuning Range and Minimal Change Parameter Tuning Range: These problems have Parameter Tuning Range as a special case with $r, s = \infty$. Hardness thus follows by restriction.

- Mode Tuning: Since C has two values, we have that $\Pr(C = \text{false}) = 1 - \Pr(C = \text{true})$. For the mode of $\Pr(C)$, we have that $\top(C) = \text{true}$ if $\Pr(C = \text{true}) > \frac{1}{2}$ and $\top(C) = \text{false}$ if $\Pr(C = \text{false}) \geq \frac{1}{2}$. If $O_S = 0$ then $\top(C) = \text{false}$. There is a combination of values $\mathbf{o}$ such that $\Pr(C = \text{true}) \geq \frac{1}{2} + \frac{1}{2^n}$ and hence $\top(C) = \text{true}$ if and only if $(\phi, \mathbf{X_E}, \mathbf{X_M})$ has a solution.

For Evidence Mode Tuning, Minimal Parameter Mode Tuning, and Minimal Change Mode Tuning, apply modifications to the construction similar to those for the corresponding Parameter Tuning-problems. □

3.4 Restricted problem variants

In the previous section, we have shown that, in the general case, Parameter Tuning Range is $\mathsf{NP}^{\mathsf{PP}}$-complete. In this section, the complexity of the problem is studied for restricted classes of instances. In particular, we will discuss tuning problems in networks with bounded topologies and tuning problems with a bounded number of CPTs containing parameters to be tuned. Finally, we will discuss a parameterised variant of Parameter Tuning Range.

3.4.1 Networks with bounded topologies

In this section we show that restrictions on the topology of the network alone do not suffice to make the various tuning problems tractable. In fact, Parameter Tuning Range remains hard, even if $\mathcal{B}$ is a polytree. Similar results can be easily derived for the other problems. We now prove NP-completeness of Parameter Tuning Range on polytrees. To this end, we reduce Maxsat to Parameter Tuning Range on polytrees, using a slightly modified proof from Park and Darwiche (2004). The (unweighted) Maxsat-problem is defined as follows:

Maxsat
Instance: Let ϕ be a Boolean formula in CNF format, let $\mathbf{C}_\phi = \{C_1, \ldots, C_m\}, m \geq 1$, be its clauses and $\mathbf{V}_\phi = \{X_1, \ldots, X_n\}, n \geq 1$, its variables, and let $1 \leq k \leq m$.
Question: Is there an assignment to the variables in $\mathbf{V}_\phi$ such that at least k clauses in ϕ are satisfied?

From ϕ, we will construct a polytree network $\mathcal{B}_\phi$ as follows. For each variable in the formula, we create a root node in the network with values true and false and a uniform prior probability distribution. We denote the parameter of X_i as O_i as in the previous construct. We define a *clause selector node* S_0 with values $c_1, \ldots, c_m$ with a uniform probability distribution, i.e. $\Pr(S_0 = c_i) = \frac{1}{m}$. Furthermore, we define *clause satisfaction nodes* S_i, with values $c_0, \ldots, c_m$, associated with each node X_i. Every node S_i has X_i and S_{i-1} as parents. The CPT for node $S_i, i \geq 1$, is defined as follows.

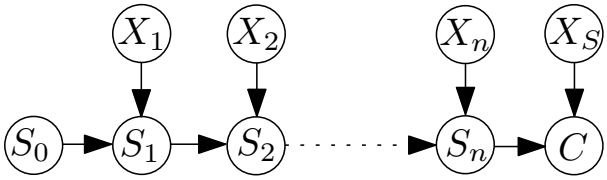


Figure 3.2 Hardness construction of Parameter Tuning restricted to polytrees

$\Pr(C = \text{TRUE} \mid X_S, S_n)$		
S_n	$X_S = \text{TRUE}$	$X_S = \text{FALSE}$
c_0	1	0
otherwise	0	0

Table 3.3 CPT for $\Pr(C \mid X_S, S_n)$

$$\Pr(S_i \mid X_i, S_{i-1}) = \begin{cases} 1 & \text{if } S_i = S_{i-1} = c_0 \\ 1 & \text{if } S_i = c_0 \text{ and } S_{i-1} = c_j, \text{ and } x_i \text{ satisfies the } j\text{th clause} \\ 1 & \text{if } S_i = S_{i-1} = c_j, \text{ and } x_i \text{ does not satisfy the } j\text{th clause} \\ 0 & \text{otherwise} \end{cases}$$

A value assignment $c_j, j > 0$ to the clause selector node S_0 models the selecting of a particular clause c_j. If this clause was not satisfied by the first $i-1$ nodes (i.e., if $S_{i-1} \neq c_0$) and the clause *is* satisfied by the value assignment to X_i, then $S_i = c_0$, otherwise $S_i = S_{i-1}$. If $S_i = c_0$, then $S_{i+1} = c_0$ and eventually $S_n = c_0$ for the clause j.

Lastly, we define a root node X_S, with values TRUE and FALSE, with a uniform probability distribution and a parameter O_S, and a node C with values TRUE and FALSE, with parents X_S and S_n. The CPT for C is given in Table 3.3. The intuitive idea behind this CPT is that $\Pr(C = \text{TRUE}) = 1$ if and only if both $X_S = \text{TRUE}$ and all clauses $\mathbf{C}_\phi$ are satisfied. See Figure 3.2 for the topology of this network.

Theorem 3.3. Parameter Tuning Range *remains* NP*-complete if* $\mathcal{B}$ *is restricted to polytrees.*

Proof. Membership of NP is immediate, since we can decide Inference in polynomial time on polytrees. Given $\mathbf{o}'$ and $\mathbf{o}$, we can thus verify for any given value assignment c and joint value assignment $\mathbf{e}$ whether $\Pr_{\mathbf{o}'}(c \,|\, \mathbf{e}) - \Pr_{\mathbf{o}}(c \,|\, \mathbf{e}) \geq q$ in polynomial time.

To prove NP-hardness, we reduce Maxsat to Parameter Tuning Range. Let (ϕ, k) be an instance of Maxsat. From the clauses $\mathbf{C}_\phi$ and variables $\mathbf{V}_\phi$, we construct $\mathcal{B}_\phi$ with parameters $\{O_1, \ldots, O_n\}$ as discussed above; we set $q = \frac{k}{n}$. We assume no evidence variables, i.e., $\mathbf{E} = \varnothing$. The reduction is obviously polynomial-time computable. Similarly as in the previous proof, if $O_S = 0$ then $\Pr(C = \text{true}) = 0$ for any combination of values to the parameters O_1 to O_n. If $O_S = 1$ then we observe the following. For every value assignment c_j to S_0, the probability distribution of S_i is as follows. $\Pr(S_i = c_0 \mid S_0 = c_j, X_i) = 1$ if the joint value assignment to $X_1, \ldots, X_i$ satisfies clause C_j, and 0 otherwise; $\Pr(S_i = c_j \,|\, S_0 = c_j, X_i) = 1$ if this joint value assignment does *not* satisfy clause C_j.

The marginal probability $\Pr(S_i)$ is dependent of the parameter O_i of node X_i. For $O_i = 0$ or $O_i = 1$, either $\Pr(S_i = c_0) = 1$ or $\Pr(S_i = c_j) = 1$ for some j; for intermediate values of O_i the probability mass is shared between $\Pr(S_i = c_0)$ and $\Pr(S_i = c_j)$. But then $\Pr(S_n = c_0 \mid S_0 = c_j) = 1$ given a particular value c_j of the clause selection node S_0, if and only if the combination of values to the parameters O_1 to O_n is such that clause c_j is satisfied by the matching joint value assignment to $X_1, \ldots, X_n$. Due to the conditional probability table of C and $O_S = 1$, $\Pr_{\mathbf{o}}(C = \text{true}) = 1$ if and only if the parameter setting $\mathbf{o}$ satisfies that clause. Summing over S_0 yields $\Pr_{\mathbf{o}}(C = \text{true}) = \frac{k}{n}$, where k is the number of clauses that is satisfied by $\mathbf{o}$. Thus, a Parameter Tuning Range query with value $q = \frac{k}{n}$ would solve the Maxsat-problem. This proves NP-hardness of Parameter Tuning Range on polytrees. □

3.4.2 Bounded number of CPTs

In the previous section we have shown that a restriction on the topology of the network in itself does not suffice to make parameter tuning tractable. In this section we show that bounding the number of CPTs containing parameters in $\mathbf{O}$ in itself is not sufficient either. If both the number of CPTs containing one or more parameters in the set $\mathbf{O}$ is bounded by a constant k (independent of the total number of parameter probabilities) and the indegree of the corresponding nodes is bounded as well, then Parameter Tuning is PP-complete. PP-hardness follows immediately since Parameter Tuning has Inference as a trivial special case (for zero parameters). We will prove membership of PP for this problem for a single parameter in a root node and show that the result also holds for a k-bounded number of CPTs with at most m parents each. Similar observations can be made for the other tuning problems defined in Section 3.2.

Theorem 3.4. Parameter Tuning *is* PP*-complete if the number of CPTs containing parameters in the set* $\mathbf{O}$ *and the indegree of the corresponding nodes are bounded by a constant.*

Proof. First assume $k = 1$, i.e., all n parameters in the set $\mathbf{O}$ are taken from the CPT of a single node X. Furthermore, assume for now that X is a root node. To solve Parameter Tuning we need to decide whether there exists a combination of values for the parameters in $\mathbf{O}$ such that $\Pr(C = c) \geq q$. Conditioning on X gives $\sum_i \Pr(C = c \mid X = x_i) \cdot \Pr(X = x_i)$. Since $\sum_i \Pr(X = x_i) = 1$, the probability $\Pr(C = c)$ is maximal for $\Pr(X = x_i) = 1$ for a particular x_i. Thus, if we want to decide whether $\Pr(C = c) \geq q$ for a particular combination of values of the parameters, then it suffices to iterate over all parameters x_i and combinations of values $\mathbf{o}$ in which $\Pr(X = x_i) = 1$ for a particular parameter x_i and successively verify whether $\Pr(C = c) \geq q$ for such a combination. Note that if the parameters subject to tuning do not include all parameter probabilities in the CPT, then we need to test whether $\Pr(C = c) \geq q$ for the combination of values in which all tunable parameters have the value 0 as well.

Using this observation, we construct a Probabilistic Turing Machine $\mathcal{M}$ by combining several machines $\mathcal{M}_i$ accepting Inference instances $\mathcal{B}_i$[2], in which the parameter $\Pr(X = x_i) = 1$ (see Figure 3.4). At its first branching step, $\mathcal{M}$

[2]See Chapter 2 for the definition of such Probabilistic Turing Machines.

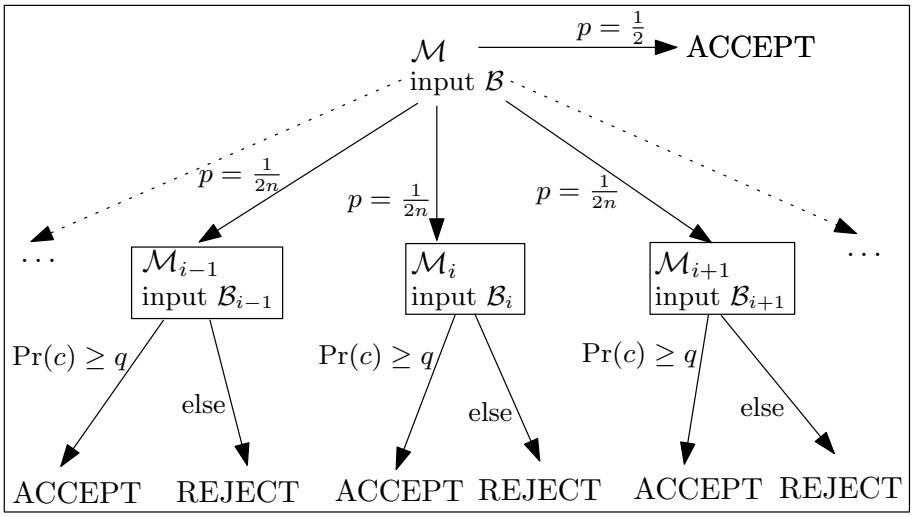


Figure 3.4 Construction of a Probabilistic Turing Machine $\mathcal{M}$ to decide Parameter Tuning with a bounded number of CPTs

---

either accepts with probability $\frac{1}{2}$, or runs, with probability $\frac{1}{2n}$, one of n Probabilistic Turing Machines $\mathcal{M}_i, i = 1, \ldots, n$, which on input $\mathcal{B}_i$ and q accept if and only if $\Pr(C = c) \geq q$. If any $\mathcal{M}_i$ accepts, then $\mathcal{M}$ accepts. $\mathcal{M}$ accepts with probability strictly larger than $\frac{1}{2}$ if and only if the Parameter Tuning instance has a solution.

If X is *not* a root node, then we must, in addition, branch over each joint value assignment to the parents of X, since we need to determine whether $\Pr(C = c) \geq q$ for a combination of values $\mathbf{o}$ in which $\Pr(X = x_i \,|\, \pi(X)) = 1$. Note that since the number of parent configurations—and hence the number of entries in the CPT—is exponential in m, we need to construct a number of Probabilistic Turing Machines $\mathcal{M}_i$ that is exponential in m.

For k CPTs with at most n tunable parameters and m incoming arcs each, we need to construct a Probabilistic Turing Machine $\mathcal{M}$ which branches over each CPT, since we must verify whether $\Pr(C = c) \geq q$ for a particular combination of values in *one* CPT, for *all* combinations of values in the other CPTs. The number of machines $\mathcal{M}_i$ we need to construct is also exponential in the number of CPTs k.

In summary, we need to construct a combined Probabilistic Turing Machine consisting of $\mathcal{O}(n^{m^k})$ Probabilistic Turing Machines accepting instances of Inference. For arbitrary m or k, this number is exponential in n. For bounded m and k, however, this is a polynomial number of machines and computation takes polynomial time. Thus, Parameter Tuning is in PP for a bounded number k of CPTs containing parameters and a bounded indegree m of the corresponding nodes. □

3.4.3 Parameterised Parameter Tuning

We have seen that Parameter Tuning is $\mathsf{NP}^{\mathsf{PP}}$-complete in general, remains NP-complete in polytrees and still is PP-complete when the number of CPTs containing parameters and the indegree of the corresponding nodes are bounded. In this section, we will investigate whether Parameter Tuning has an efficient *parameterisation*. While the intractability results strongly suggest that an algorithm with a running time polynomial in the number of parameters to be tuned is unlikely, it may be possible to obtain efficient algorithms if we are allowed to change only *a small subset* of fixed size of all parameters in $\mathbf{O}$. A problem is called *fixed parameter tractable* (Downey and Fellows, 1999) if it can be solved in polynomial time for every fixed value of a particular *problem parameter*[3], i.e. if the running time is $\mathcal{O}(f(k) \cdot n^c)$ for an arbitrary function f and a constant c, independent of n. For example, the Vertex Cover-problem (does a graph G with n vertices have a vertex cover of size at most k?) has an $\mathcal{O}(kn + c^k)$ algorithm and thus is exponential only in k, rather than in n. Hence, when the problem parameter k has a fixed value, the running time of this algorithm is polynomial in n and the problem is fixed parameter tractable. For other problems, such as the Clique-problem (does a graph G have a clique of size at least k?), it can be proven that the existence of a fixed parameter tractable algorithm is highly unlikely.

Downey and Fellows (1999) developed a theory of parameterised complexity and introduced the complexity classes FPT and the W-hierarchy. FPT and W[1] (the lowest level of the W-hierarchy) play a similar role in parameterised complexity theory as P and NP do in ordinary complexity theory. Using the commonly

[3]We will explicitly discriminate between a network parameter (an entry in a CPT) and a problem parameter (a property of the problem) to avoid confusion.

believed assumption that FPT $\neq$ W[1], proving W[1]-hardness for a particular problem is a very strong indicator that the problem is intractable, even for small values of the problem parameter under consideration. For example, Clique is known to be W[1]-hard (Downey and Fellows, 1995).

Proving W[1]-hardness[4] can be done by an *fpt-reduction* from a known W[1]-hard problem. An fpt-reduction (Downey and Fellows, 1999) is a mapping R from a parameterised problem (A, k) to a parameterised problem (B, k), computable using a fixed-parameter algorithm (i.e., exponential only in k). In the remainder, we will prove that Parameter Tuning is W[1]-hard, by reducing it (using an fpt-reduction) from the known W[1]-hard problem Positive 3Sat (Downey and Fellows, 1999). We will first introduce a parameterised variant of the Parameter Tuning-problem, in which the number of parameters that are changed is parameterised.

Parameterised Parameter Tuning
Instance: As in Parameter Tuning; furthermore let $k \leq \mid \mathbf{O} \mid$.
Question: Is there a subset $\mathbf{O}_k$ of $\mathbf{O}$ which has no more than k parameters, for which there exists a combination of values $\mathbf{o_k}$ such that $\Pr_{\mathbf{o_k}}(c \mid \mathbf{e}) \geq q$?

We will reduce Parameterized Parameter Tuning from Positive 3Sat, defined as follows.

Positive 3Sat
Instance: Boolean formula ϕ in 3cnf form with $n \geq 1$ variables and $m \geq 1$ literals, integer $1 \leq k \leq m$.
Question: Is there a satisfying truth assignment to the variables in ϕ such that no more than k literals are set to true?

It can be easily shown that the proof construction in Section 3.4.1 (illustrated in Figure 3.2) with minor adaptations suffices to fpt-reduce Positive 3Sat to Parameterized Parameter Tuning. Let $\mathcal{B}_\phi$ be the network constructed from ϕ as in Section 3.4.1, but without the X_S and C nodes and their corresponding CPTs. Instead of a uniform distribution for the nodes X_i in the network, we

[4] In fact, proving W[i]-hardness for any level i of the W-hierarchy.

define $\Pr(X_i = \text{TRUE}) = 0$ for all nodes X_i. Trivially, we can choose at most k parameters and assign them a value of 1 such that $\Pr(S_n = c_0) = 1$ if and only if we can also solve the Positive 3Sat-problem.

Corollary 3.5. Parameterized Parameter Tuning *is* $\mathsf{W}[1]$*-hard.*

3.5 Conclusion

In this chapter, we studied the computational complexity of several variants of parameter tuning. Existing algorithms for sensitivity analysis and parameter tuning (Chan and Darwiche, 2004, e.g.,) have a running time which is exponential in both the treewidth of the graph and in the number of CPTs from which the parameters are taken. We have shown that parameter tuning remains NP-hard even if the network is restricted to be a polytree, and remains PP-hard if the number of CPTs from which the parameters are taken is bounded. Furthermore, parameter tuning is not fixed parameter tractable in the number of network parameters that are tuned unless $\mathsf{FPT} = \mathsf{W}[1]$. We conclude, that Parameter Tuning is in general only tractable if *both* probabilistic inference is easy *and* the number of CPTs involved is bounded. Networks of restricted topology such as *naive Bayesian classifiers* and *tree augmented networks* (Friedman et al., 1997) may allow polynomial-time algorithms. We leave the question whether such polynomial-time algorithms exist in these special cases for further research.

Chapter 4

Monotonicity

Monotonicity in probabilistic networks is often studied in the context of probabilistic classification, where a network is constructed of evidence nodes (modelling for example observable symptoms and test results), non-observable intermediate nodes, and one or more output nodes. An output node C is isotone in a set of evidence nodes $\mathbf{E}$, if higher ordered joint value assignments to the nodes in the set $\mathbf{E}$ cannot lead to lower values of C. Antitonicity is defined in an analogous way. We distinguish between monotonicity in *mode*, where we consider the most likely value of C, and monotonicity in *distribution*, where we consider the probability distribution over C.

Monotonicity analysis is typically relevant in the construction phase of a network. The parameter probabilities associated with the network are often elicited from experts, or learned from data. In both cases the network properties induced

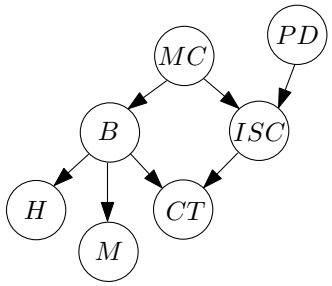
by these probabilities might not reflect the actual domain knowledge and experience. For example, medical experience related to brain tumours may dictate that more severe symptoms will not decrease the likelihood of metastatic cancer. When a domain expert insists that a particular relation ought to be monotone, the network should be such that this property is reflected in the joint probability distribution defined by the network (van der Gaag et al., 2006). If monotonicity is violated, the probability distribution in the network may be revised in cooperation with the expert to meet the monotonicity constraint.

Van der Gaag et al. (2004) established co-$\mathsf{NP}^{\mathsf{PP}}$-completeness of monotonicity analysis in probabilistic networks. The hardness result was proven for isotone relations; hardness for antitone relations, or a combination of isotone and antitone relations, followed as a corollary, given that the *direction* of the monotonicity (isotonicity or antitonicity) is *specified* for each relation prior to the analysis. For example, one might want to determine whether node C is both isotone in E_1 and antitone in E_2. When the directions are specified by the experts and need to be verified in the network, these directions are indeed known beforehand. It might be important, however, to also study *which* monotonicities hold in a network. Thus, we may also be interested in the question whether C is monotone in the set of nodes $\{E_1, E_2\}$ *without* specifying antitonicity or isotonicity.

In this chapter, we study the computational complexity of a number of variants of the monotonicity problem that use a less strict notion of monotonicity in the sense that isotonicity and antitonicity need not be specified a priori. We will further combine monotonicity with the parameter tuning problem from the previous chapter and study the problem of tuning a network in order to make it monotone. In the remainder of this chapter, we start by introducing our monotonicity variants in Section 4.1. Computational complexity results of the *loose* and *weak* monotonicity problems are given in Sections 4.2 and 4.3. In Section 4.4 we discuss *tunable* monotonicity. Finally, in Section 4.5 the results are discussed and the chapter is concluded.

4.1 Monotonicity in probabilistic networks

Before formalising the problems of which we want to determine the computational complexity, we first introduce some definitions and notations. Throughout



$\Pr(mc) = 0.20$
$\Pr(pd) = 0.10$

$\Pr(b|mc) = 0.20$
$\Pr(b|\overline{mc}) = 0.05$

$\Pr(M = \text{norm}|b) = 0.50$
$\Pr(M = \text{imp}|b) = 0.40$
$\Pr(M = \text{malf}|b) = 0.10$
$\Pr(M = \text{norm}|\overline{b}) = 0.70$
$\Pr(M = \text{imp}|\overline{b}) = 0.25$
$\Pr(M = \text{malf}|\overline{b}) = 0.05$

$\Pr(H = \text{sev}|b) = 0.70$
$\Pr(H = \text{mod}|b) = 0.25$
$\Pr(H = \text{abs}|b) = 0.05$
$\Pr(H = \text{sev}|\overline{b}) = 0.30$
$\Pr(H = \text{mod}|\overline{b}) = 0.20$
$\Pr(H = \text{abs}|\overline{b}) = 0.50$

$\Pr(isc|mc, pd) = 0.95$
$\Pr(isc|mc, \overline{pd}) = 0.80$
$\Pr(isc|\overline{mc}, pd) = 0.70$
$\Pr(isc|\overline{mc}, \overline{pd}) = 0.20$

$\Pr(CT = \text{tum}|b, isc) = 0.80$
$\Pr(CT = \text{tum}|b, \overline{isc}) = 0.90$
$\Pr(CT = \text{tum}|\overline{b}, isc) = 0.05$
$\Pr(CT = \text{tum}|\overline{b}, \overline{isc}) = 0.10$

$\Pr(CT = \text{fract}|b, isc) = 0.18$
$\Pr(CT = \text{fract}|b, \overline{isc}) = 0.01$
$\Pr(CT = \text{fract}|\overline{b}, isc) = 0.55$
$\Pr(CT = \text{fract}|\overline{b}, \overline{isc}) = 0.40$

$\Pr(CT = \text{les}|b, isc) = 0.02$
$\Pr(CT = \text{les}|b, \overline{isc}) = 0.09$
$\Pr(CT = \text{les}|\overline{b}, isc) = 0.40$
$\Pr(CT = \text{les}|\overline{b}, \overline{isc}) = 0.50$

Figure 4.1 The *Brain tumour* network

this chapter, we will refer to the *Brain tumour* network, shown in Figure 4.1, as a running example. This network, adapted from Cooper (1984), captures some fictitious and incomplete medical knowledge related to metastatic cancer; we assume that all variables are ordered as they appear in the CPTs, with lower ordered values appearing before higher values. The presence of metastatic cancer (modelled by the node MC) typically induces the development of a brain tumour (B), and an increased level of serum calcium (ISC). The latter can also be caused by Paget's disease (PD). A brain tumour is likely to increase the severity of headaches (H). Long-term memory (M) is probably impaired, or even malfunctioning. Furthermore, it is likely that a CT-scan (CT) of the head will reveal a tumour if it is present, but it may also reveal other anomalies like a fracture or a lesion, which might explain an increased serum calcium. Note that in this network MC, PD, B, and ISC are binary nodes, while M, H, and CT have multiple values.

In probabilistic networks, monotonicity relations between two sets of nodes can be defined in terms of stochastic dominance (monotonicity in distribution) or in a modal sense (monotonicity in mode). Furthermore, monotonicity can be defined globally or on a local scale, that is, concerning relations between the endpoints of an arc only; the latter is relevant when constructing qualitative probabilistic

networks (see Chapter 6). In this chapter, we only discuss global monotonicity; the reader can refer to Chapter 7 for a discussion of local monotonicity. In the remainder we study monotonicity relations between a singleton output node C and a set of observable nodes $\mathbf{E} = \{E_1, \ldots, E_n\}, n \geq 1$; our results can be generalised, however, to hold between any two sets of nodes. We assume an ordering $<$ on the values $\Omega(E)$ of a node E; for a set of nodes $\mathbf{E}$, the orderings on $E \in \mathbf{E}$ define a partial order $\prec$ on the joint value assignments to $\mathbf{E}$.

We define *strong*, *weak*, and *loose* notions of monotonicity in mode and in distribution between the output node C and a set of evidence nodes. The strong notions are similar to the notions of monotonicity in mode and in distribution discussed in Van der Gaag et al. (2004).

Definition 4.1 (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 $x \in \Omega(X)$. Let $\top(X)$ denote the mode of a node $X \in \mathbf{V}$, i.e., the most likely value of X, or in case of multiple values with equal posterior probability, the value that has the lowest order in $\Omega(X)$. Let $C \in \mathbf{V}$, and let $\mathbf{E} \subseteq \mathbf{V} \setminus \{C\}$; let $\mathbf{E}^+$ and $\mathbf{E}^-$ be subsets of $\mathbf{E}$ such that $\mathbf{E}^+ \cup \mathbf{E}^- = \mathbf{E}$ and $\mathbf{E}^+ \cap \mathbf{E}^- = \varnothing$.*

- *C is* strongly monotone in distribution *in $\mathbf{E}$, if both*
  - *for all $c \in \Omega(C)$ and all joint value assignments $\mathbf{e}^+, \mathbf{e}^{+\prime}$ to $\mathbf{E}^+$, $\mathbf{e}^+ \preceq \mathbf{e}^{+\prime} \rightarrow F(c\,|\,\mathbf{e}^+) \geq F(c\,|\,\mathbf{e}^{+\prime})$ (isotonicity),* and
  - *for all $c \in \Omega(C)$ and all joint value assignments $\mathbf{e}^-, \mathbf{e}^{-\prime}$ to $\mathbf{E}^-$, $\mathbf{e}^- \preceq \mathbf{e}^{-\prime} \rightarrow F(c\,|\,\mathbf{e}^-) \leq F(c\,|\,\mathbf{e}^{-\prime})$ (antitonicity)*

  C is strongly monotone in mode *in $\mathbf{E}$, if both*
  - *for all joint value assignments $\mathbf{e}^+, \mathbf{e}^{+\prime}$ to $\mathbf{E}^+$, $\mathbf{e}^+ \preceq \mathbf{e}^{+\prime} \rightarrow \top(C\,|\,\mathbf{e}^+) \leq \top(C\,|\,\mathbf{e}^{+\prime})$ (isotonicity),* and
  - *for all joint value assignments $\mathbf{e}^-, \mathbf{e}^{-\prime}$ to $\mathbf{E}^-$, $\mathbf{e}^- \preceq \mathbf{e}^{-\prime} \rightarrow \top(C\,|\,\mathbf{e}^-) \geq \top(C\,|\,\mathbf{e}^{-\prime})$ (antitonicity)*
- *C is* weakly monotone in distribution in $\mathbf{E}$ *if, for all nodes $E_i \in \mathbf{E}$, either*
  - *for all $c \in \Omega(C)$ and all joint value assignments e_i, e_i' to E_i and $\mathbf{e_j}$ to $\mathbf{E_j} = \mathbf{E} \setminus \{E_i\}$, $e_i < e_i' \rightarrow F(c\,|\,e_i, \mathbf{e_j}) \geq F(c\,|\,e_i', \mathbf{e_j})$ (isotonicity),* or

  - *for all* $c \in \Omega(C)$ *and all joint value assignments* e_i, e_i' *to* E_i *and* $\mathbf{e_j}$ *to* $\mathbf{E_j} = \mathbf{E} \setminus \{E_i\}$,
    $e_i < e_i' \rightarrow F(c \,|\, e_i, \mathbf{e_j}) \leq F(c \,|\, e_i', \mathbf{e_j})$ *(antitonicity)*

- C *is* weakly monotone in mode in $\mathbf{E}$ *if, for all nodes* $E_i \in \mathbf{E}$*, either*

  - *for all joint value assignments* e_i, e_i' *to* E_i *and* $\mathbf{e_j}$ *to* $\mathbf{E_j} = \mathbf{E} \setminus \{E_i\}$,
    $e_i < e_i' \rightarrow \top(C \,|\, e_i, \mathbf{e_j}) \leq \top(C \,|\, e_i', \mathbf{e_j})$ *(isotonicity),* or
  - *for all joint value assignments* e_i, e_i' *to* E_i *and* $\mathbf{e_j}$ *to* $\mathbf{E_j} = \mathbf{E} \setminus \{E_i\}$,
    $e_i < e_i' \rightarrow \top(C \,|\, e_i, \mathbf{e_j}) \geq \top(C \,|\, e_i', \mathbf{e_j})$ *(antitonicity)*

- C *is* loosely monotone in distribution in $\mathbf{E}$*, if, for all nodes* $E_i \in \mathbf{E}$*, all* $c \in \Omega(C)$ *and all joint value assignments* $\mathbf{e_j}$ *to* $\mathbf{E_j} = \mathbf{E} \setminus \{E_i\}$*, either*

  - $e_i < e_i' \rightarrow F(c \,|\, e_i, \mathbf{e_j}) \geq F(c \,|\, e_i', \mathbf{e_j})$ *for each* $e_i, e_i' \in \Omega(E_i)$ or
  - $e_i < e_i' \rightarrow F(c \,|\, e_i, \mathbf{e_j}) \leq F(c \,|\, e_i', \mathbf{e_j})$ *for each* $e_i, e_i' \in \Omega(E_i)$.

- C *is* loosely monotone in mode in $\mathbf{E}$*, if, for all nodes* $E_i \in \mathbf{E}$ *and all joint value assignments* $\mathbf{e_j}$ *to* $\mathbf{E_j} = \mathbf{E} \setminus \{E_i\}$*, either*

  - $e_i < e_i' \rightarrow \top(C \,|\, e_i, \mathbf{e_j}) \geq \top(C \,|\, e_i', \mathbf{e_j})$ *for each* $e_i, e_i' \in \Omega(E_i)$ or
  - $e_i < e_i' \rightarrow \top(C \,|\, e_i, \mathbf{e_j}) \leq \top(C \,|\, e_i', \mathbf{e_j})$ *for each* $e_i, e_i' \in \Omega(E_i)$.

In our running example, we might want to study whether the relation between MC and $\{M, H\}$ is monotone, e.g., whether the probabilities in the network are such that more severe symptoms cannot make metastatic cancer less likely. It turns out that MC is weakly monotone in distribution in $\{M, H\}$, since MC is isotone in distribution in H (more severe headaches make metastatic cancer *more* likely) and antitone in distribution in M (better functioning long term memory makes metastatic cancer *less* likely). It further turns out that MC is strongly monotone in mode in $\{M, H\}$ since higher ordered joint value assignments to $\{M, H\}$ cannot lead to a lower ordered mode of MC.

Note that if an output node is strongly monotone in some set $\mathbf{E}$, then it is also weakly monotone in this set. The reverse property does not hold just like that: while the strong variant of monotonicity assumes a partition of $\mathbf{E}$ into $\mathbf{E}^+$ and $\mathbf{E}^-$, in the weak variant it is not known a priori which relations ought to be isotone or antitone. Further note that, if the direction of the effect of $E_i \in \mathbf{E}$ on C may depend on the joint value assignment to the other nodes

in $\mathbf{E}$, then C is loosely monotone in $\mathbf{E}$. Weak monotonicity thus implies loose monotonicity. Furthermore, if all nodes in $\mathbf{E}$ are binary, then it follows directly from the definition that C is loosely monotone in $\mathbf{E}$.

If monotonicity is violated in a network where it shouldn't be, it is important to minimise the amount of change (in terms of the distance between two joint probability distributions or, alternatively, in the number of probabilities that need to be tuned; see the previous chapter) to enforce the monotonicity.

We conclude this section by formally defining the problems for which we will discuss their complexity. In the sequel, we will append $-D$ and $-M$ to the problem's name to explicitly distinguish between monotonicity in distribution and monotonicity in mode, respectively.

Strong Monotonicity
Instance: Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a probabilistic network, and let Pr be its joint probability distribution. Let C denote the output node, and let $\mathbf{E}$ denote a set of evidence nodes, partitioned into two disjoint subsets $\mathbf{E}^+$ and $\mathbf{E}^-$.
Question: Is C isotone in $\mathbf{E}^+$ and antitone in $\mathbf{E}^-$?

Weak Monotonicity
Instance: As in Strong Monotonicity, but now no partition of $\mathbf{E}$ is given.
Question: Is C weakly monotone in $\mathbf{E}$?

Loose Monotonicity
Instance: As in Weak Monotonicity.
Question: Is C loosely monotone in $\mathbf{E}$?

Tunable Monotonicity
Instance: As in Strong (Weak, Loose) Monotonicity; furthermore, let $\mathbf{O} \subseteq \Gamma$ denote a set of parameters in the network $\mathcal{B}$.
Question: Is there a combination of values $\mathbf{o}$ for $\mathbf{O}$ such that C is strongly (weakly, loosely) monotone in $\mathbf{E}$ in the network $\mathcal{B}_{\mathbf{o}}$ with the parameter probabilities $\mathbf{o}$?

We will discuss the loose and weak monotonicity variants of the monotonicity problem in Sections 4.2 and 4.3, respectively, and the tuning variants in Section 4.4.

4.2 Loose Monotonicity in Distribution

In this section, we discuss the computational complexity of the Loose Monotonicity-D-problem. As in the previous chapter, we will use the proof technique by Park and Darwiche to show that Loose Monotonicity-D is co-$\mathsf{NP}^{\mathsf{PP}}$-complete. To prove co-$\mathsf{NP}^{\mathsf{PP}}$-hardness of Loose Monotonicity-D, we construct a reduction from the A-Majsat problem, the relevant co-$\mathsf{NP}^{\mathsf{PP}}$-complete satisfiability variant discussed in Chapter 2. We do not know whether Loose Monotonicity-M is co-$\mathsf{NP}^{\mathsf{PP}}$-complete.

A-Majsat
Instance: Let ϕ be a Boolean formula with n variables $x_i, i = 1, \ldots, n, n \geq 1$, grouped into two disjoint sets $\mathbf{X_A} = \{x_1, \ldots, x_k\}$ and $\mathbf{X_M} = \{x_{k+1}, \ldots, x_n\}$ for some $1 \leq k \leq n$.
Question: For every truth assignment to $\mathbf{X_A}$, is ϕ satisfied for the majority of the truth assignments to $\mathbf{X_M}$?

Below, we will prove that any instance $(\phi, \mathbf{X_A}, \mathbf{X_M})$ of A-Majsat can be translated to a Loose Monotonicity-D instance $(\mathcal{B}, C, \mathbf{E})$ that is loosely monotone in distribution if and only if $(\phi, \mathbf{X_A}, \mathbf{X_M})$ is satisfiable. As an example, we consider the formula $\phi_{\text{ex}} = \neg(x_1 \wedge x_2) \vee (x_3 \vee x_4)$, where $\mathbf{X_A} = \{x_1, x_2\}$ and $\mathbf{X_M} = \{x_3, x_4\}$. This is a 'yes'-instance of A-Majsat because, for every truth assignment to $\mathbf{X_A}$, the majority of truth assignments to $\mathbf{X_M}$ satisfy ϕ_{ex}.

From ϕ we construct a network $\mathcal{B}_\phi$ as described in Chapter 2, with nodes in the network for each propositional variable in the formula (each with a uniform probability distribution) and for each binary connective (with a probability distribution that corresponds to the truth table of this connective). Furthermore, a node C ('classifier') and a node S ('select') are added to $\mathcal{B}_\phi$, with arcs (S, C) and (V_ϕ, C), where V_ϕ is the node in $\mathcal{B}_\phi$ modelling the top connective of ϕ; the

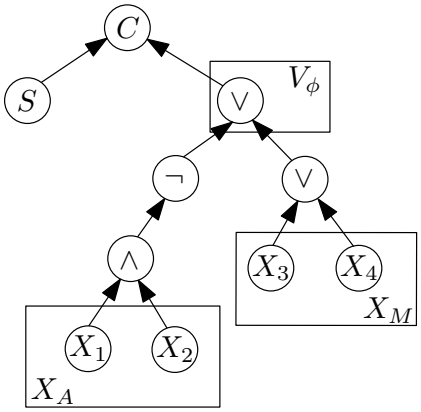


Figure 4.2 Network $\mathcal{B}_{\phi_{\text{ex}}}$ constructed from ϕ_{ex} in the co-NP$^{\text{PP}}$-hardness proof of Loose Monotonicity-D

---

select node S has values s_1, s_2 and s_3 with a uniform distribution and an ordering $s_1 < s_2 < s_3$, and the classifier node C has values c_1, c_2 and c_3 with the conditional probabilities as denoted in Table 4.3 and an ordering $c_1 < c_2 < c_3$. See Figure 4.2 for the resulting graphical structure of the network constructed from the example.

We observe that, given any joint value assignment $\mathbf{x_A}$ to $\mathbf{X_A}$, the following property holds: if $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \geq \frac{1}{2} + \frac{1}{2^n}$, then for all values c_i of C, we find that $F(c_i \,|\, s_1, \mathbf{x_A}) \geq F(c_i \,|\, s_2, \mathbf{x_A}) \geq F(c_i \,|\, s_3, \mathbf{x_A})$, where $F(C \,|\, S, \mathbf{x_A}) = F(C \,|\, S, V_\phi = \text{TRUE}) \cdot \Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) + F(C \,|\, S, V_\phi = \text{FALSE}) \cdot \Pr(V_\phi = \text{FALSE} \,|\, \mathbf{x_A})$. In contrast, if $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \leq \frac{1}{2}$ for some joint value assignment $\mathbf{x_A}$, then $F(c_1 \,|\, s_1, \mathbf{x_A}) > F(c_1 \,|\, s_2, \mathbf{x_A}) > F(c_1 \,|\, s_3, \mathbf{x_A})$ and $F(c_2 \,|\, s_1, \mathbf{x_A}) > F(c_2 \,|\, s_3, \mathbf{x_A}) > F(c_2 \,|\, s_2, \mathbf{x_A})$ from which we conclude that C is not loosely monotone in distribution in $\mathbf{X_A} \cup \{S\}$. The proof of Theorem 4.2 now shows that for the constructed network, the instance $(\mathcal{B}_\phi, C, \mathbf{X_A} \cup \{S\})$ is loosely monotone in distribution if and only if the corresponding A-Majsat-instance $(\phi, \mathbf{X_A}, \mathbf{X_M})$ is satisfiable.

Theorem 4.2. Loose Monotonicity-D *is* co-NP$^{\text{PP}}$*-complete.*

Proof. To prove membership in co-NP$^{\text{PP}}$, we show that there exists a certificate that can be used to *falsify* loose monotonicity in distribution in polynomial time,

	c_1	c_2	c_3
$S = s_1 \wedge V_\phi = \text{TRUE}$	1	0	0
$S = s_1 \wedge V_\phi = \text{FALSE}$	1	0	0
$S = s_2 \wedge V_\phi = \text{TRUE}$	0.5	0.25	0.25
$S = s_2 \wedge V_\phi = \text{FALSE}$	0.5	0.25	0.25
$S = s_3 \wedge V_\phi = \text{TRUE}$	$0.25 + \epsilon$	0.375	$0.375 - \epsilon$
$S = s_3 \wedge V_\phi = \text{FALSE}$	0.375	0.5	0.125

Table 4.3 Conditional probability table for C

given access to an oracle in PP. This certificate is a joint value assignment to $\mathbf{E}$ that satisfies the following constraints: $F(c \,|\, e_{i_1}, \mathbf{e}^-) < F(c \,|\, e_{i_3}, \mathbf{e}^-) < F(c \,|\, e_{i_2}, \mathbf{e}^-)$, where $c \in \Omega(C)$, $e_{i_1} < e_{i_2} < e_{i_3} \in \Omega(E_i)$, for a node $E_i \in \mathbf{E}$, and $\mathbf{e}^-$ is a joint value assignment to $\mathbf{E} \setminus \{E_i\}$. It is easy to see that this certificate falsifies loose monotonicity in distribution in polynomial time given access to an oracle deciding the PP-complete Inference-problem.

To prove co-$\mathsf{NP}^{\mathsf{PP}}$-hardness, we construct a transformation from A-Majsat. Let $(\phi, \mathbf{X_A}, \mathbf{X_M})$ be an instance of this problem, and let $\mathcal{B}_\phi$ be the network constructed from ϕ as described above. For a particular joint value assignment $\mathbf{x}$ to all n nodes in $\mathbf{X_A} \cup \mathbf{X_M}$, we have that $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x}) = 1$ if the joint value assignment $\mathbf{x}$ corresponds to a satisfying truth assignment to the variables in ϕ and $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x}) = 0$ if it corresponds to an unsatisfying truth assignment. Hence, for any joint value assignment $\mathbf{x_A}$ to the nodes in $\mathbf{X_A}$, $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \geq \frac{1}{2} + \frac{1}{2^n}$ if the majority of the truth assignments to the variables in $\mathbf{X_M}$ serve to satisfy ϕ. Consequently, if $(\phi, \mathbf{X_A}, \mathbf{X_M})$ is a satisfiable instance of A-Majsat, then $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \geq \frac{1}{2} + \frac{1}{2^n}$ for every joint value assignment $\mathbf{x_A}$. For all $\mathbf{x_A}$ and for all values $c_i \in \Omega(C)$ we then have that $F(c_i \,|\, s_1, \mathbf{x_A}) \geq F(c_i \,|\, s_2, \mathbf{x_A}) \geq F(c_i \,|\, s_3, \mathbf{x_A})$. Since all variables $X_j \in \mathbf{X_A}$ are binary, we further have for each variable $X_j \in \mathbf{X_A}$, for all joint value assignments $\mathbf{x}^-$ to $\mathbf{X_A} \cup \{S\} \setminus \{X_j\}$, all values $x_{j_1}, x_{j_2} \in \Omega(X_j)$ and all values $c_i \in \Omega(C)$, that $F(c_i \,|\, x_{j_1}, \mathbf{x}^-) \geq F(c_i \,|\, x_{j_2}, \mathbf{x}^-)$ or $F(c_i \,|\, x_{j_2}, \mathbf{x}^-) \geq F(c_i \,|\, x_{j_1}, \mathbf{x}^-)$. From these observations, we conclude that C is loosely monotone in distribution in $\mathbf{X_A} \cup \{S\}$.

If $(\mathcal{B}_\phi, C, \mathbf{X_A} \cup \{S\})$ is a loosely monotone instance of Loose Monotonicity-

D, then $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \geq \frac{1}{2} + \frac{1}{2^n}$ for every joint value assignment $\mathbf{x_A}$ to $\mathbf{X_A}$: if there would be a joint value assignment $\mathbf{x'_A}$ such that $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x'_A}) \leq \frac{1}{2}$, then C would not be monotone in distribution in S given that particular joint value assignment and C would not be loosely monotone in distribution in $\mathbf{X_A} \cup \{S\}$. Furthermore, if $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \geq \frac{1}{2} + \frac{1}{2^n}$, then the majority of the joint value assignments to $\mathbf{X_M}$ correspond to a satisfying truth assignment of $\mathbf{X_A} \cup \mathbf{X_M}$ to ϕ. We conclude that, if C is loosely monotone in distribution in $\mathbf{X_A} \cup \{S\}$, then $(\phi, \mathbf{X_A}, \mathbf{X_M})$ is a satisfiable instance of A-Majsat. Thus, if we can decide whether $(\mathcal{B}_\phi, C, \mathbf{X_A} \cup \{S\})$ is loosely monotone in distribution, we are able to decide $(\phi, \mathbf{X_A}, \mathbf{X_M})$. Therefore, Loose Monotonicity-D is co-$\mathsf{NP}^{\mathsf{PP}}$-hard. □

Note that the Loose Monotonicity-D-problem remains co-$\mathsf{NP}^{\mathsf{PP}}$-complete even when all nodes in the network at hand have indegree at most two.

4.3 Weak Monotonicity in Mode

In Van der Gaag et al. (2004), it was shown that the Strong Monotonicity-M-problem is co-$\mathsf{NP}^{\mathsf{PP}}$-complete[1], using a reduction from the $\mathsf{NP}^{\mathsf{PP}}$-complete cond-MAP-problem to not-Strong Monotonicity-M. We will show that the proof construction can be adjusted to prove $\mathsf{NP}^{\mathsf{PP}}$-hardness of not-Weak Monotonicity-M (and thus, by definition, co-$\mathsf{NP}^{\mathsf{PP}}$-hardness of Weak Monotonicity-M). Membership in co-$\mathsf{NP}^{\mathsf{PP}}$ can be proven using a similar argument as in Section 4.2. We do not know whether Weak Monotonicity-D is co-$\mathsf{NP}^{\mathsf{PP}}$-complete.

The cond-MAP-problem is defined in Van der Gaag et al. (2004) as follows.

cond-MAP
Instance: Let $\mathcal{B} = (\mathbf{G}, \Gamma)$ be a probabilistic network, and let Pr be its joint probability distribution. Let $W \in \mathbf{V}$ be a binary node in $\mathbf{G}$ with a value

[1] While the article claims (Van der Gaag et al., 2004, pp. 571) that a similar complexity result directly follows for the Strong Monotonicity-D-problem, this claim is not proven in the article and the proof is not easily reconstructable.

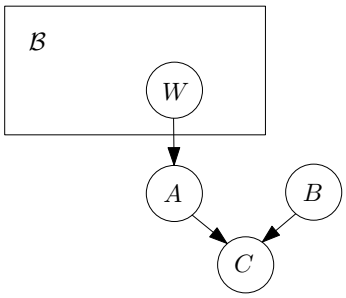


Figure 4.4 Network $\mathcal{B}'$ constructed from COND-MAP instance $(\mathcal{B}, W, w, \mathbf{V_{MAP}}, q)$

$w \in \Omega(W)$, let $\mathbf{V_{MAP}} \subset \mathbf{V} \setminus \{W\}$ denote a subset of nodes in $\mathbf{V}$, and let $0 \leq q \leq 1$ be a rational number.
Question: Is there a joint value assignment $\mathbf{v}$ to the nodes in $\mathbf{V_{MAP}}$ such that $\Pr(w \,|\, \mathbf{v}) > q$?

We first rephrase the reduction from COND-MAP to NOT-STRONG MONOTONICITY-M as given in Van der Gaag et al. (2004). Let $(\mathcal{B}, W, w, \mathbf{V_{MAP}}, q)$ be an instance of COND-MAP. From $\mathcal{B}$, we construct the network $\mathcal{B}'$ by adding three binary nodes A, B, and C and three arcs (W, A), (A, C), and (B, C) (see Figure 4.4). We define:

$$\Pr(a \,|\, W) = \begin{cases} 1 & \text{if } W = w \\ \frac{\frac{1}{2} - q}{1 - q} & \text{otherwise} \end{cases}$$

and

$$\Pr(c \,|\, A, B) = \begin{cases} 1 & \text{if } A = a \text{ and } B = \bar{b} \\ 0 & \text{otherwise} \end{cases}$$

Now, let $\mathbf{E} = \mathbf{V_{MAP}} \cup \{B\}$, and let $\mathbf{E}^+ = \mathbf{E}$ and therefore $\mathbf{E}^- = \varnothing$. In the thus constructed network, there is a value assignment $\mathbf{v}$ to $\mathbf{V_{MAP}}$ with $\Pr(w \,|\, \mathbf{v}) > q$ if and only if C is not isotone in mode in $\mathbf{E}^+$, i.e., if and only if there are joint value assignments $\mathbf{v_1}$ and $\mathbf{v_2}$ to $\mathbf{E}$ with $\mathbf{v_1} \prec \mathbf{v_2}$, but $\top(C \,|\, \mathbf{v_1}) > \top(C \,|\, \mathbf{v_2})$. We refer to Van der Gaag et al. (2004) for the formal proof. The intuition behind the proof is as follows. Assume that there exists a value assignment $\mathbf{v}$ to

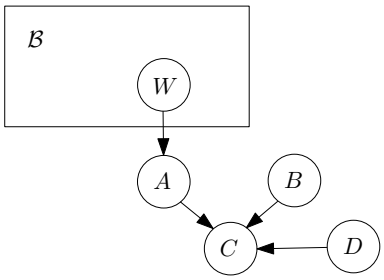


Figure 4.5 Network $\mathcal{B}'$ constructed from COND-MAP instance $(\mathcal{B}, W, w, \mathbf{V_{MAP}}, q)$

$\mathbf{V_{MAP}}$ such that $\Pr(w \,|\, \mathbf{v}) > q$. Let $\mathbf{v_1}$ and $\mathbf{v_2}$ be the joint value assignments to $\mathbf{E} = \mathbf{V_{MAP}} \cup \{B\}$ obtained by assigning $\mathbf{v}$ to $\mathbf{V_{MAP}}$ and $\bar{b}$, respectively b, to B; note that $\mathbf{v_1} \prec \mathbf{v_2}$. We find that

$$\Pr(a \,|\, \mathbf{v}) = \Pr(a \,|\, w) \cdot \Pr(w \,|\, \mathbf{v}) + \Pr(a \,|\, \bar{w}) \cdot \Pr(\bar{w} \,|\, \mathbf{v}) > q + (1 - q) \cdot \frac{\frac{1}{2} - q}{1 - q} = \frac{1}{2}$$

Furthermore, we find that $\Pr(c \mid \mathbf{v_1}) = \Pr(a \mid \mathbf{v}) > \frac{1}{2}$ since we assigned $\bar{b}$ to B in $\mathbf{v_1}$. We further find that $\Pr(c \mid \mathbf{v_2}) = 0$, hence, isotonicity in the mode of C in $\mathbf{E}$ is violated. Similarly it can be shown that, if there are joint value assignments $\mathbf{v_1}$ and $\mathbf{v_2}$ such that isotonicity is violated, then $\Pr(w \mid \mathbf{v}) > q$ must hold. Assume we have $\Pr(w \mid \mathbf{v}) = p$ for some value $p \in [0, 1]$. Since $\Pr(a \,|\, \mathbf{v}) = \Pr(a \,|\, w) \cdot \Pr(w \,|\, \mathbf{v}) + \Pr(a \,|\, \bar{w}) \cdot \Pr(\bar{w} \,|\, \mathbf{v})$ and $\Pr(a \,|\, \mathbf{v}) > \frac{1}{2}$, we have that $p + (1 - p) \cdot \frac{\frac{1}{2} - q}{1 - q} > \frac{1}{2}$. We conclude that $p > q$ and thus $\Pr(w \mid \mathbf{v}) > q$. Thus, NOT-STRONG MONOTONICITY-M is $\mathsf{NP}^{\mathsf{PP}}$-hard and STRONG MONOTONICITY-M is $\mathsf{co\text{-}NP}^{\mathsf{PP}}$-hard.

To prove hardness of WEAK MONOTONICITY-M, we change the construction of $\mathcal{B}'$ by adding an extra binary node D, with a uniform prior probability distribution, and an extra arc (D, C) (see Figure 4.5). Furthermore, we redefine the conditional probability distributions for node C as follows.

$$\Pr(c \,|\, A, B, D) = \begin{cases} 1 & \text{if } A = a,\ B = \bar{b} \text{ and } D = d \\ 1 & \text{if } A = a,\ B = b \text{ and } D = \bar{d} \\ 0 & \text{otherwise} \end{cases}$$

Theorem 4.3. WEAK MONOTONICITY-M *is* $\mathsf{co\text{-}NP}^{\mathsf{PP}}$*-complete*

Proof. To prove membership in co-$\mathsf{NP}^{\mathsf{PP}}$ of Weak Monotonicity-M, we show that there exists a certificate that can be used to *falsify* weak monotonicity in mode in polynomial time, given access to an oracle in PP. This certificate is a joint value assignment to $\mathbf{E}$ such that $\top(C \,|\, e_{i_1}, e_{j_1}, \mathbf{e}^-) > \top(C \,|\, e_{i_2}, e_{j_1}, \mathbf{e}^-)$ and yet $\top(C \mid e_{i_2}, e_{j_2}, \mathbf{e}^-) > \top(C \mid e_{i_1}, e_{j_2}, \mathbf{e}^-)$, where $\mathbf{e}^- \in \Omega(\mathbf{E} \setminus \{E_i, E_j\}), e_{i_1} < e_{i_2}, e_{i_1}, e_{i_2} \in \Omega(E_i)$ and $e_{j_1}, e_{j_2} \in \Omega(E_j)$, for two nodes $E_i, E_j \in \mathbf{E}$. It is easy to see that this certificate falsifies Weak Monotonicity-M in polynomial time given access to a PP-oracle.

To prove co-$\mathsf{NP}^{\mathsf{PP}}$-hardness, we reduce not-Weak Monotonicity-M from the $\mathsf{NP}^{\mathsf{PP}}$-complete problem cond-MAP. From the construction described above it follows that C is not weakly monotone in mode in $\mathbf{E} \cup \{D\} = \mathbf{V_{MAP}} \cup \{B, D\}$ if and only if there is a joint value assignment $\mathbf{v}$ to $\mathbf{V_{MAP}}$ with $\Pr(w \,|\, \mathbf{v}) > q$, since in that case we would have that $\top(C \mid \mathbf{v_1}, d) > \top(C \mid \mathbf{v_2}, d)$, but $\top(C \mid \mathbf{v_1}, \bar{d}) < \top(C \mid \mathbf{v_2}, \bar{d})$. Since the construction ensures that $\mathbf{v_1}$ and $\mathbf{v_2}$ differ in variable B only, we have that weak monotonicity is violated in B. From the cond-MAP instance $(\mathcal{B}, W, w, \mathbf{V_{MAP}}, q)$ we can construct a Weak Monotonicity-M instance $(\mathcal{B}', D, \mathbf{V_{MAP}} \cup \{W\})$ in polynomial time; hence we can reduce cond-MAP to not-Weak Monotonicity-M in polynomial time and thus not-Weak Monotonicity-M is $\mathsf{NP}^{\mathsf{PP}}$-hard and by definition Weak Monotonicity-M is co-$\mathsf{NP}^{\mathsf{PP}}$-hard. The construction obviously can be made in polynomial time, hence, Weak Monotonicity-M is co-$\mathsf{NP}^{\mathsf{PP}}$-complete. □

4.4 Tunable Monotonicity in Distribution

In the construction phase of a network, we will also be interested in the question whether a given set of parameters can be *tuned* in order to meet specific monotonicity constraints. In this section we prove that this problem is $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete for the loose monotonicity variant. We will again use the proof technique by Park and Darwiche to reduce EA-Majsat, the canonical $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete problem, to Tunable Loose Monotonicity-D. We do not know whether Tunable Loose Monotonicity-M is $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete.

We first give the definition of EA-Majsat:

EA-Majsat
Instance: Let ϕ be a Boolean formula with n variables $x_i, i = 1, \ldots, n, n \geq 1$. Let $1 \leq k < l \leq n$, let $\mathbf{X_E}$, $\mathbf{X_A}$, and $\mathbf{X_M}$ be the sets of variables x_1 to x_k, x_{k+1} to x_l, and x_{l+1} to x_n, respectively.
Question: Is there a truth assignment to $\mathbf{X_E}$ such that for every possible truth assignment to $\mathbf{X_A}$, the majority of the truth assignments to $\mathbf{X_M}$ satisfy ϕ?

From an EA-Majsat-instance $(\phi, \mathbf{X_E}, \mathbf{X_A}, \mathbf{X_M})$ we construct a probabilistic network $\mathcal{B}_\phi$ as in Section 4.2, but with designated node sets $\mathbf{X_E}$, $\mathbf{X_A}$, and $\mathbf{X_M}$. Again, the select node S has values s_1, s_2 and s_3 with a uniform distribution, and the classifier node C has conditional probabilities as in Table 4.3. All nodes X_i initially have a uniform probability distribution, with parameters $O_i = \Pr(X_i = \text{TRUE})$ for each node $X_i \in \mathbf{X_E}$. We take the instance $(\phi_{\text{ex}} = \neg((x_1 \vee x_2) \wedge (x_3 \vee x_4)) \wedge (x_5 \vee x_6), \mathbf{X_E} = \{x_1, x_2\}, \mathbf{X_A} = \{x_3, x_4\}, \mathbf{X_M} = \{x_5, x_6\})$ as an example; the graphical structure of the network $\mathcal{B}_{\phi_{\text{ex}}}$ constructed for ϕ_{ex} is shown in Figure 4.6. This EA-Majsat-instance is satisfiable: take $x_1 = x_2 = \text{FALSE}$, then for every truth assignment to $\{x_3, x_4\}$, the majority of the truth assignments to $\{x_5, x_6\}$ satisfy ϕ_{ex}. Correspondingly, in $\mathcal{B}_{\phi_{\text{ex}}}$, the classifier node C is loosely monotone in $\mathbf{X_A} \cup \{S\}$ if the parameters O_1, O_2 are assigned the value 0. In that case, for every joint value assignment $\mathbf{x}$ to $\{X_3, X_4\}$, $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x}) \geq \frac{1}{2} + \frac{1}{2^n}$, and thus we have that for all values c_i of C, $F(c_i \,|\, s_1, \mathbf{x}) \geq F(c_i \,|\, s_2, \mathbf{x}) \geq F(c_i \,|\, s_3, \mathbf{x})$ by a similar argument as in Section 4.2.

Theorem 4.4. Tunable Loose Monotonicity-D *is* $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$*-complete.*

Proof. To prove membership, it suffices to show that we can choose, non-deterministically, a combination of parameter values $\mathbf{o}$ for the set of tunable parameters $\mathbf{O}$ and use an oracle for Loose Monotonicity-D to verify that this combination of values indeed makes C loosely monotone in $\mathbf{E}$. Since Loose Monotonicity-D is $\mathsf{co\text{-}NP}^{\mathsf{PP}}$-complete, this proves[2] membership of $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$.

To prove hardness, we show that every EA-Majsat-instance $(\phi, \mathbf{X_E}, \mathbf{X_A}, \mathbf{X_M})$ can be reduced to a corresponding instance $(\mathcal{B}_\phi, \mathbf{O}, C, \mathbf{X_A} \cup \{S\})$ of Tunable

[2]Recall from Chapter 2 that $\mathsf{NP}^{\mathsf{co-C}} = \mathsf{NP}^{\mathsf{C}}$ since we can use both 'yes' and 'no' answers from the oracle.

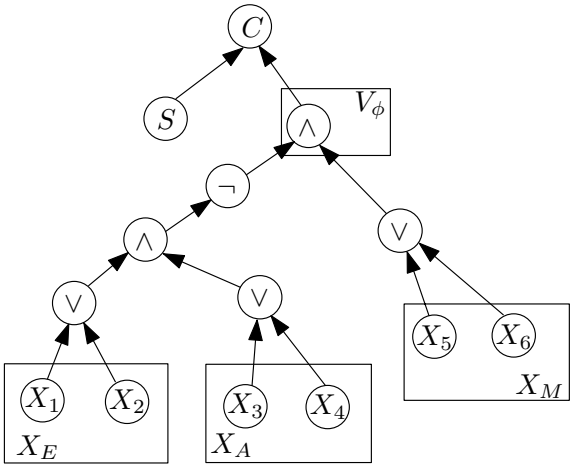


Figure 4.6 Network $\mathcal{B}_{\phi_{\text{ex}}}$ constructed from ϕ_{ex} in the $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-hardness proof of Tunable Loose Monotonicity-D

Loose Monotonicity-D in polynomial time. Let $\mathcal{B}_\phi$ be the probabilistic network constructed from ϕ as shown above. Assume there exists a combination $\mathbf{o_m}$ of values for the parameters in $\mathbf{O}$ such that C is loosely monotone in distribution in the evidence variables $\mathbf{X_A} \cup \{S\}$. As shown in Chapter 3, we may assume that all parameter values in $\mathbf{o_m}$ are either 0 or 1. Since C is loosely monotone in $\mathbf{X_A} \cup \{S\}$, in particular it holds that $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \geq \frac{1}{2} + \frac{1}{2^n}$ for every joint value assignment $\mathbf{x_A}$ to the nodes in $\mathbf{X_A}$ as shown in Section 4.2; if $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x_A}) \leq \frac{1}{2}$ for any such joint value assignment $\mathbf{x_A}$, then C would no longer be loosely monotone in $\mathbf{X_A} \cup \{S\}$. But then, there exists a solution to the EA-Majsat-instance $(\phi, \mathbf{X_E}, \mathbf{X_A}, \mathbf{X_M})$. On the other hand, if $(\phi, \mathbf{X_E}, \mathbf{X_A}, \mathbf{X_M})$ is a satisfying instance of EA-Majsat, then there exists a truth assignment to $\mathbf{X_E}$ such that, for all truth assignments to $\mathbf{X_A}$, the majority of the truth assignments to $\mathbf{X_M}$ satisfy ϕ. If we assign values corresponding to the truth values in $\mathbf{X_E}$ to the parameters in $\mathbf{O}$, C will be loosely monotone in $\mathbf{X_A} \cup \{S\}$ by a similar argument as in the proof of Theorem 4.2. The reduction can obviously be done in polynomial time, hence, Tunable Loose Monotonicity-D is $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete. □

We do not know whether Tunable Weak Monotonicity-D and Tunable Strong Monotonicity-D are $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-hard. While Loose Monotonicity-

D was proven co-$\mathsf{NP}^{\mathsf{PP}}$-complete by a reduction from a suitable satisfiability variant, hardness of Weak Monotonicity-M and Strong Monotonicity-M was proven by a reduction from cond-MAP. This makes these proofs less 'portable' to problems in a complexity class higher in the counting hierarchy CH. We also do not know whether Tunable Weak Monotonicity-M and Tunable Strong Monotonicity-M are $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete.

4.5 Conclusion

In this chapter, several variants of the Monotonicity-problem in probabilistic networks were introduced. We argued that a less strict notion of monotonicity than the definition given in Van der Gaag et al. (2004) (denoted as Strong Monotonicity here) is sometimes appropriate. We have shown that determining whether a relation is *weakly* monotone in mode or *loosely* monotone in distribution is equally hard as determining whether it is *strongly* monotone in mode. We have proven that the problem of tuning a network such that it is monotone in distribution is $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete for the loose monotonicity variant. While these hardness results all suggest intractability of the problems, the $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-completeness result for Tunable Loose Monotonicity-D is interesting both from a theoretical point of view—we do not know of any other problem with a practical application that is $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete—and in practice, since it suggests that Tunable Loose Monotonicity-D remains NP-hard even with strong restrictions on the topology of the network and the number of parameters involved (see also Chapter 3).

Chapter 5

Finding the kth MPE and kth Partial MAP

An important problem in probabilistic networks is finding the most likely joint value assignment to a set of nodes $\mathbf{M}$ (known as the MAP variables), given full or partial evidence for the other nodes in the network. When the set of evidence nodes $\mathbf{E}$ is equal to the entire complement of $\mathbf{M}$ in the network (i.e., $\mathbf{E} = \mathbf{V} \setminus \mathbf{M}$), the problem is known as the Most Probable Explanation or MPE-problem . Most Probable Explanation is known to have various NP-complete decision variants (Shimony, 1994; Bodlaender et al., 2002) and is hard to approximate as well (Abdelbar and Hedetniemi, 1998). Finding the most likely joint value assignment, given evidence for a proper *subset* of the complement set (i.e., $\mathbf{V}$ is partitioned into the MAP variables $\mathbf{M}$, the evidence variables $\mathbf{E}$ and a set of intermediate variables $\mathbf{I} \neq \varnothing$) is known as the Partial Maxi-

mum A-Posteriori Probability or Partial MAP-problem. This problem is even harder: Park and Darwiche (2004) proved $\mathsf{NP}^{\mathsf{PP}}$-completeness of the decision variant of this problem.

In practical applications, one often wants to find a number of different joint value assignments with a high likelihood, rather than just the most likely one (Santos, 1991; Charniak and Shimony, 1994). For example, in medical applications, one wants to suggest alternative (but also likely) explanations to a set of observations. One might like to prescribe medication that treats a number of plausible (combinations of) diseases, rather than just the most probable combination. It may also be useful to examine the second-best explanation to gain insight in *how good* the best explanation is, relative to other solutions, or how sensitive it is to changes in the parameters of the network (Chan and Darwiche, 2006). While algorithms exist that can find the k-th most probable joint value assignment fast once the best explanation is known (Charniak and Shimony, 1994), it has been shown that in general calculating or even approximating the k-th best explanation is NP-hard (Abdelbar and Hedetniemi, 1998). The exact complexity of this problem has, however, not been established until now.

The complexity of finding the k-th most probable joint value assignment given partial evidence has, to our best knowledge, not yet been investigated. Yet, in many applications it is unlikely that full evidence of the complement of the nodes of interest in the network is available. For example, in the *Oesophagus Network* discussed in Chapter 1, a number of nodes are intermediate and non-observable. The *ALARM network* (Beinlich et al., 1989) has sixteen observable and thirteen intermediate nodes. Therefore, the problem of finding k-th best assignments given *partial* evidence, may even be more relevant in practical applications than the corresponding problem where full evidence is available.

In this chapter, we extend the problem of finding the most probable joint value assignment to the problem of finding the k-th most probable joint value assignment for arbitrary values of k, with either full or partial evidence. We will prove that (variants of) these problems are complete for the complexity classes $\mathsf{FP}^{\mathsf{PP}}$ and $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$, respectively, suggesting that these problems are much harder than the (already intractable) restricted cases where $k = 1$, and also much harder than the PP-complete Inference-problem.

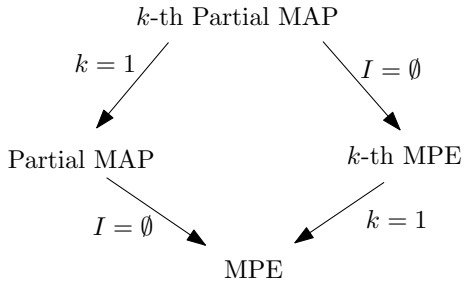


Figure 5.1 Relationship between Most Probable Explanation, Partial MAP, Kth MPE, and Kth Partial MAP.

We recall that decision variants of Most Probable Explanation and Partial MAP are NP-complete and $\mathsf{NP}^{\mathsf{PP}}$-complete, respectively. We now further show that the problem of actually *finding* the most probable joint value assignment given full or partial evidence (i.e., the functional variants, rather than the decision variants, of these problems), is $\mathsf{FP}^{\mathsf{NP}}$-complete, respectively $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete. Figure 5.1 shows the relation between the Most Probable Explanation variants discussed above; the relations hold both with functional problems as with the decision problems. In the remainder, we will use (f) and (d) when appropriate to discriminate between functional and decision variants of problems.

There are problems known to be $\mathsf{FP}^{\mathsf{NP}}$-complete or $\mathsf{FP}^{\mathsf{PP}}$-complete, like finding the optimal (Krentel, 1988) or k-th optimal (Toda, 1994) tour in a TSP-problem. However, finding the most probable or k-th most probable joint value assignment given partial evidence is (to the best of our knowledge) the first problem with a practical application that is now shown to be $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$-, respectively $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete. This makes the problem interesting from a more theoretical viewpoint as well.

This chapter is organised as follows. First, in Section 5.1, we will discuss the complexity classes P^{PP} and P^{NP} and their functional counterparts $\mathsf{FP}^{\mathsf{PP}}$ and $\mathsf{FP}^{\mathsf{NP}}$. We will discuss the complexity of finding k-th joint value assignments with full, respectively partial, evidence in Sections 5.2 and 5.3. In Section 5.4 we conclude this chapter.

5.1 Complexity classes described by metric Turing Machines

In complexity theory, we are often interested in *decision problems*, i.e., problems for which the answer is *yes* or *no*. Well-known complexity classes like P and NP are defined for decision problems and are formalised using Turing Machines. In this chapter, we emphasize *function problems*, i.e., problems for which the answer is a function of the input. For example, the problem of determining whether a solution to a 3Sat instance exists, is in NP; the problem of actually *finding* such a solution is in the corresponding function class FNP. Function classes are defined using Turing Transducers, i.e., machines that not only halt in an accepting state on a satisfying input on its input tape, but also return a result on an output tape.

In Chapter 2, we have introduced *oracles* and complexity classes like $\mathsf{NP}^{\mathsf{PP}}$ and $\mathsf{co\text{-}NP}^{\mathsf{PP}}$, i.e., classes of problems for which certificates of membership (respectively non-membership) can be verified in polynomial time, given access to an oracle for problems in PP. In probabilistic networks, problems in these classes typically combine guessing solutions (or counterexamples) and calculating probabilities. We have seen examples of complete problems for $\mathsf{NP}^{\mathsf{PP}}$ in Chapter 3 (the Parameter Tuning-problems) and for $\mathsf{co\text{-}NP}^{\mathsf{PP}}$ in Chapter 4 (the Monotonicity-problems). Here we introduce the classes P^{NP} and P^{PP} and their functional variants $\mathsf{FP}^{\mathsf{NP}}$ and $\mathsf{FP}^{\mathsf{PP}}$, as studied by e.g. Papadimitriou (1984), Krentel (1988), and Toda (1994).

The complexity class P^{NP} is defined as the class of languages, decidable in polynomial time by a deterministic Turing Machine with access to an NP oracle. P^{NP}, or Δ_2^p using an alternative notation, is at the second level of the Polynomial Hierarchy (Stockmeyer, 1977). Complete problems for this class (and its functional counterpart $\mathsf{FP}^{\mathsf{NP}}$) have been discussed by Papadimitriou (1984) and Krentel (1988). Similarly, the class P^{PP} is defined as the class of languages decidable in polynomial time by a deterministic Turing Machine with access to a PP oracle. P^{PP} is contained in the Counting Hierarchy (CH). Toda (1994) discussed complete problems for P^{PP} and $\mathsf{FP}^{\mathsf{PP}}$.

The canonical complete problems for $\mathsf{FP}^{\mathsf{NP}}$ and P^{NP}, respectively $\mathsf{FP}^{\mathsf{PP}}$ and P^{PP}, are LexSat and MidSat (or, equivalently, KthSat), respectively (Krentel,

1988; Toda, 1994). For the functional classes $\mathsf{FP}^{\mathsf{NP}}$ and $\mathsf{FP}^{\mathsf{PP}}$, LexSat and MidSat (KthSat) ask for the lexicographically first, respectively middle (or k-th), satisfying assignment $x_1x_2\ldots x_n \in \{0,1\}^n$ to a Boolean formula ϕ. Correspondingly, for the classes P^{NP} and P^{PP}, the decision variants of these problems ask whether the least significant bit in such an assignment equals 1. Throughout this chapter, we denote x_1 to be the most significant element in the order: the assignment $0111\ldots1$ is ordered *before* $1000\ldots0$.

One way to prove *membership* in P^{NP}or P^{PP} is showing that a problem instance is accepted by a deterministic Turing Machine in polynomial time, given access to an oracle[1] for a particular problem in NP, respectively PP. Another way was introduced by Krentel (1988), namely by using *metric Turing Machines*. Similarly, membership in $\mathsf{FP}^{\mathsf{NP}}$ or $\mathsf{FP}^{\mathsf{PP}}$ can be proven using *metric Turing Transducers*.

Definition 5.1 (Metric Turing Machine). *A metric Turing Machine (metric TM for short) $\widehat{\mathcal{M}}$ is a polynomial-time bounded non-deterministic Turing Machine such that every computation path halts with a binary number on an output tape. $Out_{\widehat{\mathcal{M}}}(x)$ then denotes the set of outputs of $\widehat{\mathcal{M}}$ on input x, $Opt_{\widehat{\mathcal{M}}}(x)$ defines the smallest number in $Out_{\widehat{\mathcal{M}}}(x)$, and $KthValue_{\widehat{\mathcal{M}}}(x,k)$ is defined to be the k-th smallest number in $Out_{\widehat{\mathcal{M}}}(x)$. Metric Turing Transducers (metric TTs for short), denoted as $\widehat{\mathcal{T}}$, are defined likewise using Turing Transducers with an additional output tape.*

A function f is *polynomial-time one-Turing reducible* to a function g ($f \leq^{\mathsf{FP}}_{\text{1-T}} g$ for short) if there exist polynomial-time computable functions T_1 and T_2 such that for every x, $f(x) = T_1(x, g(T_2(x)))$ (Toda, 1994, p.5). A function f is in $\mathsf{FP}^{\mathsf{NP}}$ if and only if there exists a metric TT $\widehat{\mathcal{T}}$ such that $f \leq^{\mathsf{FP}}_{\text{1-T}} \text{Opt}_{\widehat{\mathcal{T}}}$. Correspondingly, a set L is in P^{NP} if and only if a metric TM $\widehat{\mathcal{M}}$ can be constructed, such that $\text{Opt}_{\widehat{\mathcal{M}}}(x)$ is odd if and only if $x \in L$. Similar observations hold for $\mathsf{FP}^{\mathsf{PP}}$ and P^{PP}, and the $\text{KthValue}_{\widehat{\mathcal{M}}}$ and $\text{KthValue}_{\widehat{\mathcal{T}}}$ functions (Krentel, 1988; Toda, 1994). $\mathsf{FP}^{\mathsf{NP}}$- or $\mathsf{FP}^{\mathsf{PP}}$-hardness of a function problem can be proven by reducing it from a known $\mathsf{FP}^{\mathsf{NP}}$-, respectively $\mathsf{FP}^{\mathsf{PP}}$-hard problem using a polynomial-time one-Turing reduction; P^{NP}- or P^{PP}-hardness of a decision problem can be proven by reducing it from a known P^{NP}-, respectively P^{PP}-hard problem using polynomial-time many-one reductions.

[1] Note that the machine may use both positive and negative answers of the oracle in its computation. NP is presumed to be a strict subset of P^{NP} and PP a strict subset of P^{PP}.

In the remainder, we will construct metric TTs for the functional variants of the MPE- and Partial MAP-problems to prove membership in $\mathsf{FP}^{\mathsf{NP}}$, $\mathsf{FP}^{\mathsf{PP}}$, $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$, and $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$. In the hardness proofs in this chapter, we will sometimes use the fact that $\#\mathsf{P}$ is polynomial-time Turing equivalent to PP (Simon, 1977), that is, these classes are equally powerful when used as an oracle. Informally, this implies that a class $\mathcal{C}$ with oracle access to $\#\mathsf{P}$ is equal to the class $\mathcal{C}$ with oracle access to PP. For example, the class P^{PP} is equal to the class $\mathsf{P}^{\#\mathsf{P}}$; however, the former notation is more common. Lastly, we assume that all joint value assignments can be uniquely ordered: $\mathbf{v_1} \prec \mathbf{v_2}$ if $\Pr(\mathbf{v_1}) < \Pr(\mathbf{v_2})$; if $\Pr(\mathbf{v_1}) = \Pr(\mathbf{v_2}) = p$ for two joint value assignments $\mathbf{v_1}$ and $\mathbf{v_2}$, they are ordered lexicographically by their respective binary representation. In general, $\mathbf{v_1} \prec \mathbf{v_2}$ if and only if $\text{bin}(\Pr(\mathbf{v_1}), \mathbf{v_1}) \prec \text{bin}(\Pr(\mathbf{v_2}), \mathbf{v_2})$.

5.2 Kth MPE

In this section we prove $\mathsf{FP}^{\mathsf{PP}}$-completeness for the functional variant of the Kth MPE-problem. More specifically, we show that Kth MPE (f) can be computed by a metric TT in polynomial time (proving membership of $\mathsf{FP}^{\mathsf{PP}}$), and we prove hardness of the problem by a reduction from KthSat. We formally define the functional version of the Kth MPE-problem as follows.

Kth MPE (f)
Instance: A probabilistic network $\mathcal{B} = (\mathbf{G}, \Gamma)$, where $\mathbf{V}$ is partitioned into a set of evidence nodes $\mathbf{E}$ with a joint value assignment $\mathbf{e}$ and a set of MAP variables $\mathbf{M}$, a natural number k.
Output: The k-th most probable joint value assignment $\mathbf{v_k}$ to the nodes in $\mathbf{M}$ given evidence $\mathbf{e}$; if no such assignment exists, the output will be the empty set.

Note that the MPE (f)-problem is a specific case of this problem where $k = 1$. Further note that we can transform the functional version of Kth MPE into a decision variant by designating a node $V_d \in \mathbf{V} \setminus \mathbf{E}$ with v_d as one of its values, and asking whether $V_d = v_d$ in the solution $\mathbf{v_k}$. The formulation of this decision variant is quite similar to the formulation of the decision version of KthSat

where the problem is to determine whether a designated variable in the problem solution is set to TRUE.

The functional version of KTHSAT , the problem that we will use in the reduction, is defined as follows.

KTHSAT (f)
Instance: A Boolean formula $\phi(x_1, \ldots, x_n), n \geq 1$, a natural number k.
Output: The lexicographically k-th truth assignment $\mathbf{x_k} \in \{0,1\}^n$ that satisfies ϕ, or the empty set if such an assignment does not exist.

We define the LEXSAT (f)-problem to be the special case of this problem where $k = 1$. We will now use the formula $\phi_{\text{ex}} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$ for our running example. As in previous chapters, we construct a probabilistic network $\mathcal{B}_\phi$ from the Boolean formula ϕ of a KTHSAT-instance as illustrated for our example formula in Figure 5.2. For each variable x_i in the formula ϕ, we create a matching stochastic variable X_i in $\mathbf{V}$ for the network $\mathcal{B}_\phi$, with possible values TRUE and FALSE. These nodes are roots in the network $\mathcal{B}_\phi$ and constitute the *variable instantiation* part ($\mathbf{X}$) of the network. The prior probabilities $p_i = \Pr(X_i = \text{TRUE})$ for $i = 1, \ldots, n$ are chosen such that the prior probability of a particular value assignment $\mathbf{x}$ to $\mathbf{X}$ is higher than that of $\mathbf{x}'$ if and only if the corresponding truth assignment $\mathbf{x}_\phi$ to $X_1, \ldots, X_n$ is lexicographically ordered before $\mathbf{x}'_\phi$. More in particular, we choose prior probabilities $p_1, \ldots, p_i, \ldots, p_n$ such that $p_i = \frac{1}{2} - \frac{2^i - 1}{2^{n+1}}$. Given the observation that the variables X_i are mutually independent apriori, this choice assures that for every $i < n$, $p_i \cdot (1 - p_{i+1}) \cdot \ldots \cdot (1 - p_n) > (1 - p_i) \cdot p_{i+1} \cdot \ldots \cdot p_n$, thereby guaranteeing the desired property. In our example with four variables, the prior probabilities are $p_1 = \frac{15}{32}$, $p_2 = \frac{13}{32}$, $p_3 = \frac{9}{32}$, and $p_4 = \frac{1}{32}$. Note that we can formulate these probabilities using a number of bits which is polynomial in the number of variables of the KTHSAT-instance.

The remainder of the construction uses the technique discussed earlier in Section 2.4 and used in Chapters 3 and 4. For each logical operator in ϕ, we create an additional stochastic variable in the network, whose parents are the corresponding sub-formulas (or single sub-formula in case of a negation operator) and whose conditional probability table is equal to the truth table of that operator. We denote the stochastic variable that is associated with the top-level operator

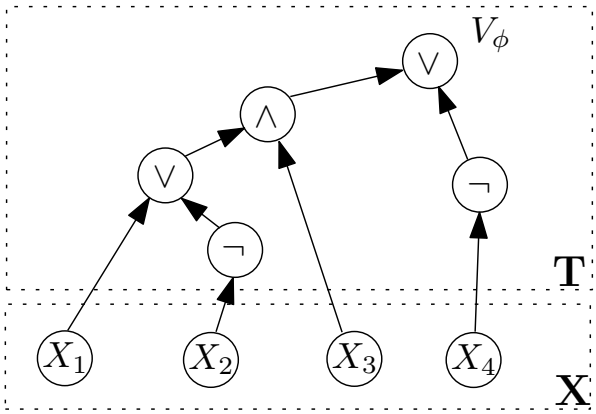


Figure 5.2 Example of k-th MPE construction for the formula $\phi_{\text{ex}} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$

---

in ϕ with V_ϕ. The thus constructed part of the network will be called the *truth-setting* part ($\mathbf{T}$) of the network. It is easy to see that, given a particular value assignment $\mathbf{x}$ to the nodes X_i in the network, $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{x}) = 1$ if and only if the corresponding truth assignment to the variables in ϕ satisfies ϕ.

Theorem 5.2. Kth MPE *is* $\mathsf{FP}^{\mathsf{PP}}$*-complete.*

Proof. To prove membership, we show that a metric TT can be constructed for the Kth MPE-problem. We first observe that $\Pr(\mathbf{M} \mid \mathbf{e}) = \frac{\Pr(\mathbf{M},\mathbf{e})}{\Pr(\mathbf{e})}$ and that $\Pr(\mathbf{e})$ can be regarded as a constant factor if we are interested in the *relative order* of the conditional probabilities of joint value assignments to $\mathbf{v}$ given evidence $\mathbf{e}$. Now, let $\widehat{\mathcal{T}}$ be a metric non-deterministic TT that on input $(\mathcal{B}, \mathbf{e})$ calculates the joint probability distribution $\Pr(\mathbf{M}, \mathbf{e})$. Since $\mathbf{V} = \mathbf{M} \cup \mathbf{E}$ and $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(V_i \,|\, \pi(V_i))$, any computation path of $\widehat{\mathcal{T}}$ calculates $\Pr(\mathbf{m}, \mathbf{e})$ for a particular joint value assignment $\mathbf{m}$ to $\mathbf{V} \setminus \mathbf{E}$ by traversing a topological sort of $\mathbf{V}$ and non-deterministically choosing value assignments v_i (conform the evidence $\mathbf{e}$), at each step i, and multiplying the corresponding probabilities. This takes a computation time which is polynomial in the total number of variables in the Kth MPE instance. The output is, for each computation path, a binary representation of $1 - \Pr(\mathbf{m}, \mathbf{e})$ with sufficient (but polynomial) precision, combined with a binary representation of $\mathbf{v}$ itself. Then, clearly $\text{KthValue}_{\widehat{\mathcal{T}}}((\mathcal{B}, \mathbf{e}), k)$ returns an encoding of the k-th most probable explanation for $\mathbf{e}$. This proves

that Kth MPE is in $\mathsf{FP}^{\mathsf{PP}}$.

To prove hardness, we reduce KthSat to Kth MPE. Let (ϕ, k) be an instance of KthSat and let $\mathcal{B}_\phi$ be the network constructed from ϕ as described above. For any joint value assignment $\mathbf{x}$ to $\mathbf{X}$, $\Pr(\mathbf{X} = \mathbf{x} \,|\, V_\phi = \text{TRUE}) = \frac{\Pr(\mathbf{X}=\mathbf{x}, V_\phi=\text{TRUE})}{\Pr(V_\phi=\text{TRUE})}$. The prior probability $\Pr(V_\phi = \text{TRUE})$ can be regarded as constant (say α) in our instance. We further observe that $\Pr(\mathbf{X} = \mathbf{x}, V_\phi = \text{FALSE}) = 0$ if $\mathbf{x}$ is a satisfying assignment. For any satisfying assignment $\mathbf{x}$, therefore, we have that $\Pr(\mathbf{X} = \mathbf{x} \,|\, V_\phi = \text{TRUE}) = \alpha \cdot \Pr(\mathbf{X} = \mathbf{x})$.

With $\mathbf{E} = \{V_\phi\}$, we have that the set $\mathbf{M}$ contains both the nodes in the set $\mathbf{X}$ as well as the nodes that model logical operators. Note that the values of the nodes that model logical operators are fully determined by the values of their parents. Then, given evidence $V_\phi = \text{TRUE}$, the kth MPE corresponds to the lexicographically k-th satisfying value assignment to the variables in ϕ, and vice versa. Thus, given an algorithm for calculating the kth MPE, we can solve the KthSat-problem as well. Clearly, the above reduction is a polynomial-time one-Turing reduction from KthSat to Kth MPE. This proves $\mathsf{FP}^{\mathsf{PP}}$-hardness of Kth MPE. □

From the construction used above, we observe that the Kth MPE-problem remains $\mathsf{FP}^{\mathsf{PP}}$-complete even when all nodes of the probabilistic network have indegree at most two and the non-evidence nodes are all root nodes.

We now turn to the special case where $k = 1$, i.e., the Most Probable Explanation (f) problem. $\mathsf{FP}^{\mathsf{NP}}$-completeness of Most Probable Explanation (f) can be shown using only a slight modification of the construction.

Proposition 5.3. Most Probable Explanation *(f) is* $\mathsf{FP}^{\mathsf{NP}}$*-complete.*

Proof. To prove membership, we construct a similar metric TT as above. We have that $\text{Opt}_{\widehat{\mathcal{T}}}(\mathcal{B}, \mathbf{e})$ then returns the most probable explanation for evidence $\mathbf{e}$. This proves that Most Probable Explanation is in $\mathsf{FP}^{\mathsf{NP}}$. To prove hardness, we apply the same construction as above to reduce LexSat to Most Probable Explanation using a polynomial-time one-Turing reduction. This proves $\mathsf{FP}^{\mathsf{NP}}$-hardness of Most Probable Explanation. □

As a side remark, we observe that Most Probable Explanation (f) is in $\mathsf{FP}^{\mathsf{NP}}$, while Most Probable Explanation (d), the problem whether there exists an instantiation $\mathbf{v}$ such that $\Pr(\mathbf{v} \mid \mathbf{e} > q$ for given $\mathbf{e}$ and q, is in NP(Shimony, 1994). This relation between the decision variant and the functional variant of a problem is quite commonly found in optimisation problems: if the solution of a functional problem variant has polynomially bounded length, then there exists a polynomial-time Turing reduction from the functional variant to the decision variant of that problem, and hence if the decision variant is in NP, then the functional variant of the problem is in $\mathsf{FP}^{\mathsf{NP}}$ (Papadimitriou, 1994). A typical example is the Travelling Salesman Problem, where the decision problem is known to be NP-complete and the functional problem is $\mathsf{FP}^{\mathsf{NP}}$-complete (Krentel, 1988).

It is common for decision variants of optimisation problems to refer to a threshold value for the optimal solution. In the case of Most Probable Explanation, e.g., the problem would be whether there exists a joint value assignment to $\mathbf{V}$ with a posterior probability given $\mathbf{e}$ greater than a threshold value q. One can argue whether a formulation of decision problems in terms of a threshold value gives a true indication of the hardness of the original problem. For example, one can give a certificate that corresponds to a joint value assignment whose probability is greater than q; that value assignment, however, does not need to be the most probable one, which was the original question. If we would reformulate the decision problem of an optimisation problem to refer to a particular *property* of the optimal solution (e.g., for Most Probable Explanation: is it the case that in the most probable explanation, variable V_i has value v_i?), then this decision problem can be proven to be P^{NP}-complete rather than NP-complete, and the proof of membership (constructed using a metric TM) would actually involve the most probable solution: take a similar construction as above to reduce the decision variant of KthSat (given a formula ϕ, is the lexicographically k-th satisfying truth assignment odd) to Kth MPE and ask whether X_n has the value true in the k-th most probable explanation. Furthermore, such a formulation of a decision problem would discriminate between the complexity of problems in which *any* solution qualifies—like 3Sat—and problems in which we are interested particularly in the *optimal* solution. Thus, to obtain more informative complexity proofs, a decision variant of an optimisation problem should be constructed using a property of the optimal solution, rather than a threshold value for a solution.

5.3 Kth Partial MAP

While the decision variant of the Most Probable Explanation-problem is complete for the class NP, i.e., solvable in polynomial time by a non-deterministic TM, the decision variant of the Partial MAP-problem is complete for $\mathsf{NP}^{\mathsf{PP}}$, i.e., solvable by a non-deterministic TM with access to an oracle for problems in PP (Park and Darwiche, 2004). In the previous section we have proven that the functional variant of the Kth MPE-problem is complete for $\mathsf{FP}^{\mathsf{PP}}$, thus solvable in polynomial time by a nondeterministic metric Turing Transducer. Intuitively, this suggests that the Kth Partial MAP-problem is complete for $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$, the class of function problems solvable in polynomial time by a metric TT with access to a PP-complete oracle. To the best of our knowledge, no complete problems have been discussed in the literature for this complexity class. We will now show that the Kth Partial MAP-problem indeed is complete for the class $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$, by a reduction from a suitable Satisfiability variant. We introduce the KthNumSat-problem, defined as follows.

KthNumSat
Instance: A Boolean formula $\phi(x_1, \ldots, x_m, \ldots, x_n), m \leq n, n \geq 1$, natural numbers k, l.
Output: The lexicographically k-th assignment $\mathbf{x_k}$ to $x_1, \ldots, x_m$ for which exactly l assignments $\mathbf{x_l}$ to $x_{m+1}, \ldots, x_n$ satisfy ϕ, or the empty set if such an assignment does not exist.

We will prove in the appendix that KthNumSat is $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete. Similarly, we will prove that LexNumSat, the special case where $k = 1$, is $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete. To prove hardness of Kth Partial MAP (f), we will use a version of this problem with *bounds* on the probability distribution over the MAP nodes $\mathbf{M}$.

Bounded Kth Partial MAP (f)
Instance: A probabilistic network $\mathcal{B} = (\mathbf{G}, \Gamma)$, where $\mathbf{V}$ is partitioned into a set of evidence nodes $\mathbf{E}$ with a joint value assignment $\mathbf{e}$, a set of intermediate nodes $\mathbf{I}$, and a set of MAP nodes $\mathbf{M}$; a natural number k, and rational numbers $0 \leq a \leq b \leq 1$.

Output: A tuple $(\mathbf{v_k}, p_k)$, where $\mathbf{v_k}$ is the k-th most probable assignment to $\mathbf{M}$ given evidence $\mathbf{e}$, where we consider only joint value assignments with a joint probability $p_k = \Pr(\mathbf{v_k}, \mathbf{e})$ in the interval $[a, b]$; the empty set, if such an assignment does not exist.

The Kth Partial MAP(f)-problem, i.e. the original problem *without* boundary constraints, is a special case of the bounded problem with $a = 0$ and $b = 1$. Now we can use binary search techniques to find a solution to the *bounded* problem variant in polynomial time, using an algorithm for the *unbounded* problem variant; we include the probability $\Pr(\mathbf{v_k}, \mathbf{e})$ in the output of the unbounded problem to be able to calculate a solution to the bounded problem as follows. Assume that the bounds are a and b and that we are interested in the k-th Partial MAP within the interval $[a, b]$. We first compute $\Pr(\mathbf{e})$ by considering an empty set of MAP variables (Park and Darwiche, 2004). Let p_i denote the probability of the i-th Partial MAP. With binary search techniques and using only a polynomial number of Kth Partial MAP queries, we can calculate natural numbers k' and k'' such that and $p_{k'} < a \leq p_{k'+1}$ and $p_{k''} \leq b < p_{k''+1}$. Another single Kth Partial MAP query is needed to find the $k' + k$-th Partial MAP. We test whether $k' + k \leq k''$ and output 'yes' if this is the case, and 'no' otherwise. We can thus transform a bounded problem variant into a problem without bounds in polynomial time, and vice versa, i.e., Bounded Kth Partial MAP is Turing equivalent to Kth Partial MAP. However, using the bounded problem formulation makes our hardness proof easier.

We will prove $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-completeness of Bounded Kth Partial MAP (f) by a reduction from KthNumSat. We will again use the formula $\phi_{\text{ex}} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$ as a running example (see Figure 5.3). We want to find the lexicographically second assignment to $\{x_1, x_2\}$ for which exactly three truth assignments to $\{x_3, x_4\}$ satisfy ϕ_{ex}, that is, $k = 2$ and $l = 3$. The reader can verify that this is the case for $x_1 = \text{TRUE}, x_2 = \text{FALSE}$.

As in the previous section, we construct a probabilistic network $\mathcal{B}_\phi$ from a given KthNumSat instance $(\phi(x_1, \ldots, x_m, \ldots, x_n), k, l)$. Again, we create a stochastic variable X_i for each variable x_i in ϕ, but now with a uniform probability distribution, i.e., $\Pr(X_i = \text{TRUE}) = \frac{1}{2}$. We denote the nodes $X_1, \ldots, X_m$ as the variable instantiation part ($\mathbf{X}$). These nodes are the MAP variables $\mathbf{M}$ in our Bounded Kth Partial MAP-construction. For each logical operator in ϕ, we create additional nodes in the network as in the previous section, with V_ϕ as the

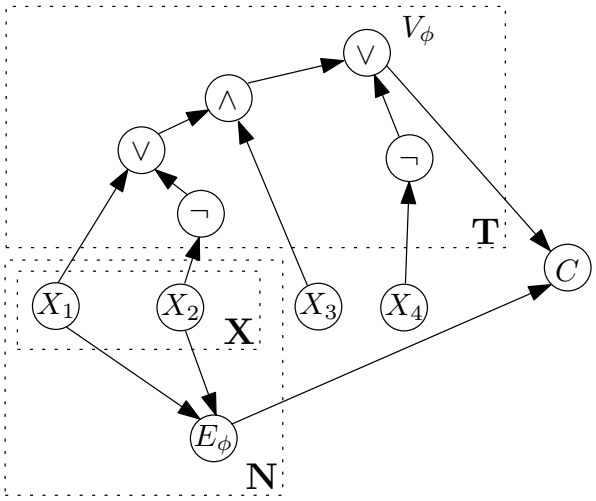


Figure 5.3 Example of k-th Partial MAP construction for the formula $\phi_{\text{ex}} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$, with MAP variables X_1 and X_2

---

node associated with the top level operator in ϕ. Observe that, for any value assignment $\mathbf{v_k}$ to the MAP variables $\{X_1, \ldots, X_m\}$, $\Pr(V_\phi = \text{TRUE} \,|\, \mathbf{v_k}) = \frac{s}{2^{n-m}}$, where s is the number of truth value assignments to the variables $\{x_{m+1}, \ldots, x_n\}$ that jointly with $\mathbf{v_k}$ satisfy ϕ.

Furthermore, we construct an *enumeration* part (**N**) of the network. This sub-network consists of the MAP variables $X_1, \ldots, X_m$ and additional nodes $E_{p,q}$, each with possible values TRUE and FALSE. Without loss of generality, we assume that the number of MAP variables is a power of two; we can use additional dummy nodes if needed. **N** has the form of a $\log m$-deep binary tree, with the MAP variables $X_1, \ldots, X_m$ as roots, a single leaf variable E_ϕ, and a structure as follows. Let $\pi(E_{p,1}) = (X_{2p-1}, X_{2p}), p = 1, \ldots, \frac{m}{2}$, and $\pi(E_{p,q}) = (E_{2p-1,q-1}, E_{2p,q-1})$ denote the sequence of parents for $E_{p,1}$ and for $E_{p,q}, q = 2, \ldots, \log m$, respectively. Then the conditional probability table for $E_{p,q}$ is defined as follows:

$$\begin{aligned}
\Pr(E_{p,q} = \text{TRUE} \,|\, \pi(E_{p,q}) = (\text{TRUE}, \text{TRUE})) &= 0 \\
\Pr(E_{p,q} = \text{TRUE} \,|\, \pi(E_{p,q}) = (\text{TRUE}, \text{FALSE})) &= \tfrac{1}{2^{p+n-m+1}} \\
\Pr(E_{p,q} = \text{TRUE} \,|\, \pi(E_{p,q}) = (\text{FALSE}, \text{TRUE})) &= \tfrac{2}{2^{p+n-m+1}} \\
\Pr(E_{p,q} = \text{TRUE} \,|\, \pi(E_{p,q}) = (\text{FALSE}, \text{FALSE})) &= \tfrac{3}{2^{p+n-m+1}}
\end{aligned}$$

The node at the lowest level of this tree will be denoted as E_ϕ. If the number of levels of this tree (including $\mathbf{X}$) is odd, then we denote the node at the lowest level as E'_ϕ, and we add an additional binary node E_ϕ, with an arc (E'_ϕ, E_ϕ) and a CPT that acts as an inverter, i.e., $\Pr(E_\phi = \text{TRUE} \,|\, E'_\phi = \text{FALSE}) = 1$ and $\Pr(E_\phi = \text{FALSE} \,|\, E'_\phi = \text{TRUE}) = 1$. In the example network, there are only two MAP variables ($m = 2$) so $E_\phi = E_{1,1}$ with probabilities $\Pr(E_\phi = \text{TRUE} \,|\, X_1, X_2)$ equal to 0, $\frac{1}{16}$, $\frac{2}{16}$, and $\frac{3}{16}$ for the value assignments $\{\text{TRUE}, \text{TRUE}\}$, $\{\text{TRUE}, \text{FALSE}\}$, $\{\text{FALSE}, \text{TRUE}\}$ and $\{\text{FALSE}, \text{FALSE}\}$, to X_1, X_2, respectively. Note that the above construction ensures that, for any assignment $\mathbf{v_k}$ to $\mathbf{M}$, the probability $\Pr(E_\phi = \text{TRUE} \,|\, \mathbf{v_k})$ is always smaller than $\frac{1}{2^{n-m}}$: this probability is at most $\frac{3}{2^{n-m+2}}$ if $\mathbf{N}$ consists of a singleton variable, and will be even smaller when $\mathbf{N}$ grows.

Observe that the number of levels in the tree of $E_{p,q}$ nodes equals $\log m$: with each additional level, the number of nodes doubles, and so does the maximum value of p in the CPTs for the nodes $E_{p,q}$. Thus, the value of the denominator of each conditional probability in the CPT of a node $E_{p,q}$ grows with a factor $2^{\frac{p}{2}}$ with respect to the denominator of the probabilities in the CPT of $E_{p/2,q}$. We will now show that if an assignment $\mathbf{v_k}$ to the MAP variables is lexicographically ordered before $\mathbf{v'_k}$, then $\Pr(E_\phi = \text{TRUE} \mid \mathbf{v'_k}) < \Pr(E_\phi = \text{TRUE} \mid \mathbf{v_k})$. This is obviously the case in the most basic case with two variables X_1, X_2: here $\Pr(E_\phi = \text{TRUE} \mid \mathbf{x})$ equals 0, $\frac{1}{4}$, $\frac{2}{4}$, and $\frac{3}{4}$, for joint value assignment $\mathbf{x} = (\text{TRUE}, \text{TRUE})$, $(\text{TRUE}, \text{FALSE})$, $(\text{FALSE}, \text{TRUE})$, and $(\text{FALSE}, \text{FALSE})$, respectively. Assume that we have m variables $X_1, \ldots, X_m$ with $\Pr(E_\phi = \text{TRUE} \,|\, \mathbf{x_j}) = \frac{i+aj}{2^k}$, where j is the decimal equivalent of the binary value of $\mathbf{x_j}$, $a \in \{-1, 1\}$, and i and k are particular values (in the basal case, $i = 3$, $a = -1$, and $k = 2$), we will show that a construction of $2m$ variables $X_1, \ldots, X_{2m}$ will yield a probability distribution $\Pr(E_\phi = \text{TRUE} \,|\, \mathbf{x_j}) = \frac{i'-aj}{2^{k'}}$ for particular values i' and k'; that is, if the construction with m variables ensured that lexicographical higher ordered values of $\mathbf{x}$ induced a higher probability $\Pr(E_\phi = \text{TRUE} \mid \mathbf{x})$, then the construction with $2m$ variables preserves that property, but with the lexicographic order reversed.

For a joint value assignment $\mathbf{x_r}$ to the variables in $\mathbf{X} = X_1, \ldots, X_m$ (for tree level r), let $\mathbf{x_{r(t)}}$ denote the assignment to variable X_t. We have that $\Pr(E_\phi = \text{TRUE} \,|\, \mathbf{x_{r+1}}) = \sum_{\mathbf{X_r}} \Pr(E_\phi = \text{TRUE} \,|\, \mathbf{X_r}) \cdot \Pr(\mathbf{X_{r(1)}} \,|\, \mathbf{X_{r+1(1)}}, \mathbf{X_{r+1(2)}}) \cdot \Pr(\mathbf{X_{r(2)}} \,|\, \mathbf{X_{r+1(3)}}, \mathbf{X_{r+1(4)}}) \cdot \ldots \cdot \Pr(\mathbf{X_{r(p)}} \,|\, \mathbf{X_{r+1(2p-1)}}, \mathbf{X_{r+1(2p)}})$. Furthermore, we have from the induction hypothesis that $\Pr(E_\phi = \text{TRUE} \,|\, \mathbf{X_r}) = \frac{i+aj}{2^k}$. From the CPT for $E_{p,q}$ we find that the denominator doubles with every factor of the summation

since p increases with every variable; thus, an arbitrary row in the summation has the following structure: $\Pr(E_\phi = \text{TRUE} | \mathbf{x_{r+1}}) = \frac{i+aj}{2^k} \cdot \frac{A}{2^{k+1}} \cdot \frac{C}{2^{k+2}} \cdot \ldots \cdot \frac{Z}{2^{k+p}}$. Now assume we have two distinct joint value assignments $\mathbf{x}$ and $\mathbf{x}'$; by definition, they have a common part in front (that may be empty) and at some point they start to differ. Assume that they differ only in the last position, then only the first and the last factor in the summation are relevant, so we have $\Pr(E_\phi = \text{TRUE} | \mathbf{x_{r+1}}) = \frac{i+aj}{2^k} \cdot \alpha \cdot \frac{Z}{2^{k+p}}$ for a particular constant value α. Observe that we sum over all p joint value instantiations to $\mathbf{X_r}$; of course, in $\frac{1}{2}$ of the cases the last position of an instantiation will be TRUE and in $\frac{1}{2}$ of the cases it will be false. If the last position is TRUE, then the summation can be simplified to $\alpha \cdot \frac{p \cdot aj + 4p}{2^{2k+p+1}}$. On the other hand, if the last position is FALSE, then the summation simplifies to $\alpha \cdot \frac{p \cdot aj + 4p}{2^{2k+p+1}} + \frac{p}{2^{k+p+1}}$. Using a similar argument, if we only look at the assignment at position $m-1$, we find that the summation simplifies to $\alpha \cdot \frac{p \cdot aj + 4p}{2^{2k+p+1}}$ (for TRUE), respectively $\alpha \cdot \frac{p \cdot aj + 4p}{2^{2k+p+1}} + \frac{2p}{2^{k+p+1}}$ (for FALSE), and in general, for position $m - l (l = 0, \ldots, m)$, $\alpha \cdot \frac{p \cdot aj + 4p}{2^{2k+p+1}}$ for TRUE and $\alpha \cdot \frac{p \cdot aj + 4p}{2^{2k+p+1}} + \frac{2^l \cdot p}{2^{k+p+1}}$ for FALSE. We conclude that the desired property of the probability distribution is guaranteed.

To conclude the construction, we add an additional node C with parents V_ϕ and E_ϕ, with the following conditional probability table:

$$\Pr(C = \text{TRUE} | V_\phi, E_\phi) = \begin{cases} 1 & \text{if } V_\phi = \text{TRUE} \wedge E_\phi = \text{TRUE} \\ \frac{1}{2} & \text{if } V_\phi = \text{TRUE} \wedge E_\phi = \text{FALSE} \\ \frac{1}{2} & \text{if } V_\phi = \text{FALSE} \wedge E_\phi = \text{TRUE} \\ 0 & \text{if } V_\phi = \text{FALSE} \wedge E_\phi = \text{FALSE} \end{cases}$$

Since $\Pr(E_\phi = \text{TRUE} | \mathbf{v_k}) < \frac{1}{2^{n-m}}$ for any joint value assignment $\mathbf{v_k}$ to the MAP variables, this CPT assures that the probability $\Pr(C = \text{TRUE} | \mathbf{v_k})$ is within the interval $[\frac{s}{2^{n-m+1}}, \frac{s+1}{2^{n-m+1}}]$, where s denotes the number of value assignments to the variables $\{x_{m+1}, \ldots, x_n\}$ that make ϕ true.

Theorem 5.4. Bounded Kth Partial MAP *(f) is* $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$*-complete.*

Proof. The $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$ membership proof is very similar to the $\mathsf{FP}^{\mathsf{PP}}$ membership proof of the Kth MPE (f)-problem, but now we are allowed to use an oracle for Exact Inference (which is #P-complete, see Section 2.4) to compute probabilities $\Pr(\mathbf{v})$. Note that for the Bounded Kth Partial MAP (f) problem,

we actually *need* the power of this oracle: in contrast to Kth MPE (f) where we compute the probability of a joint value assignment to *all* variables, which takes polynomial time, we must now sum over all joint value assignments to $\mathbf{I}$ to compute $\Pr(\mathbf{v_k}, \mathbf{e})$. If $\Pr(\mathbf{v_k}, \mathbf{e})$ is within the interval $[a, b]$, we output the tuple $(\mathbf{v_k}, 1 - \Pr(\mathbf{v_k}, \mathbf{e}))$; if not, we output $\varnothing$. Clearly, $\text{KthValue}_{\widehat{\mathcal{T}}}$ returns the k-th probable value assignment to the MAP variables, and this proves that Bounded Kth Partial MAP (f) is in $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$.

To prove hardness, we construct a probabilistic network $\mathbf{B}_\phi$ from a given instance $\phi(x_1, \ldots, x_m, \ldots, x_n)$, similar to the previous section. The conditional probabilities in the thus constructed network ensure that the probability of a value assignment $\mathbf{v_k}$ to the nodes $\{X_1, \ldots, X_m\}$ such that l truth value assignments to the variables $\{x_{m+1}, \ldots, x_n\}$ satisfy ϕ, is in the interval $[\frac{l}{2^{n-m+1}}, \frac{l+1}{2^{n-m+1}}]$. Moreover, if both $\mathbf{v_k}$ and $\mathbf{v_k}'$ are such that l truth value assignments to the variables $\{x_{m+1}, \ldots, x_n\}$ satisfy ϕ, then $\Pr(C = T \mid \mathbf{v_k}) > \Pr(C = T \mid \mathbf{v'_k})$ if the truth value that corresponds with $\mathbf{v_k}$ is lexicographically ordered before $\mathbf{v'_k}$. Thus, with evidence $C =$ true and ranges $[\frac{s}{2^{n-m+1}}, \frac{s+1}{2^{n-m+1}}]$, the k-th Partial MAP corresponds to the lexicographical k-th truth assignment to the variables $x_1 \ldots x_m$ for which exactly s truth assignments to $x_{m+1} \ldots x_n$ satisfy ϕ. Clearly, the above reduction is a polynomial-time one-Turing reduction from KthNum-Sat to Kth Partial MAP. This proves $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-hardness of Bounded Kth Partial MAP (f). □

Observe again, that the problem remains $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete when the MAP variables have no incoming arcs, when all nodes have indegree at most two, and all nodes are binary. $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$-completeness of Bounded Partial MAP (f), the special case of Bounded Kth Partial MAP (f) where $k = 1$, now follows as a corollary, the proof of which is very similar as in the previous section and will be left to the reader.

Corollary 5.5. Bounded Partial MAP *(f) is* $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$*-complete.*

5.4 Conclusion

In this chapter, we addressed the computational complexity of finding the k-th MPE or k-th partial MAP. We have shown that the Kth MPE-problem is $\mathsf{FP}^{\mathsf{PP}}$-

complete, making it considerably harder than both Most Probable Explanation (which is $\mathsf{FP}^{\mathsf{NP}}$-complete) and Inference (which is PP-complete), given usual assumptions in complexity theory. The computational power of P^{PP} and $\mathsf{FP}^{\mathsf{PP}}$ (and thus the intractability of Kth MPE) is illustrated by Toda's theorem (1991) which states that P^{PP} includes the entire Polynomial Hierarchy (PH). Yet, finding the k-th MPE is arguably easier than finding the most probable explanation given only partial evidence (the Partial MAP-problem) which is $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete. When inference can be done in polynomial time (such as in polytrees) we can find the k-th MPE in polynomial time (Sy, 1992; Srinivas and Nayak, 1996), while the Partial MAP-problem remains NP-complete.

Finding the k-th Partial MAP is considerably harder than finding the k-th MPE, again under usual assumptions in complexity theory. We have shown that finding the k-th Partial MAP is $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete in general. Park and Darwiche (2004) show that the Partial MAP-problem remains NP-complete on polytrees, using a reduction from 3Sat[2]. Their proof can be easily modified to reduce Kth Partial MAP on polytrees from the $\mathsf{FP}^{\mathsf{PP}}$-complete problem Kth3Sat (Toda, 1994), hence finding the k-th Partial MAP on polytrees remains $\mathsf{FP}^{\mathsf{PP}}$-complete; a similar observation holds for the functional variant of Partial MAP which remains $\mathsf{FP}^{\mathsf{NP}}$-complete on polytrees.

For small or fixed k, the Kth MPE- and Kth Partial MAP-problems may be somewhat easier than for arbitrary k. On the other hand, Kth MPE is already NP-complete in the special case that $k = 1$, ruling out a possible *fixed-parameter tractable* algorithm which would have a running time exponential only in k.

[2]Technically, Park and Darwiche reduce Partial MAP from Maxsat to preserve approximation results.

Part II

Abstraction

Chapter 6

Inference in Interval-based Networks

While probabilistic networks are based on intuitive and qualitative notions of causality and probabilistic influence in uncertain knowledge, eliciting the required numerical probabilistic information from experts can be a difficult task (Druzdzel and van der Gaag, 2000). Furthermore, many problems related to fully quantified probabilistic networks have a high computational complexity, as has been shown in the previous chapters. To overcome these problems, qualitative probabilistic networks, or QPNs, have been proposed by Wellman (1990) as a qualitative abstraction of probabilistic networks.

In QPNs, the conditional probabilities for a variable given its parents in the network are summarised into a sign per parent, which denotes the qualitative influence between these variables. In contrast to quantitative networks, where inference is intractable (as shown in Chapter 2), standard QPNs have polynomial-

time inference algorithms, in which the qualitative influence of an observed value upon other variables can be calculated by message-passing between neighbouring nodes in the network (Druzdzel and Henrion, 1993*a*). Applications of QPNs include their use (as an intermediate step) in the construction of probabilistic networks (Renooij and van der Gaag, 2002), in approximation algorithms for probabilistic networks (van der Gaag et al., 2004), and for inference when the exact probability distribution is unknown or irrelevant (Wellman, 1990).

Unfortunately, reasoning in a qualitative abstraction fails to give a conclusive result when influences with contrasting signs are combined. Renooij and van der Gaag (1999) introduced *Enhanced* QPNs in order to allow for more flexibility in expressing strengths of influences (e.g., *weakly positive* versus *strongly positive*) and partially resolve conflicts upon combining influences of different strength. Also, *mixed* networks (Renooij and van der Gaag, 2002) have been proposed to facilitate stepwise quantification by allowing both qualitative and quantitative influences to be modelled in a network. Although polynomial-time algorithms are known for inference in standard qualitative networks, the computational complexity of inference in enhanced networks has not been determined yet.

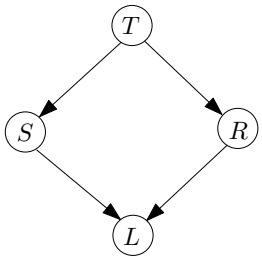
In this chapter we recall the definition of QPNs in Section 6.1, and discuss enhanced and interval-based QPNs in Section 6.2. The basic inference problem in these QPNs is discussed, and in Section 6.3, we show that a more generalised inference problem om interval graphs, in which we allow problem instances that have no underlying probabilistic network, is NP-hard. In Section 6.4 we introduce a framework to relate various enhancements, such as enhanced, *richly* enhanced, and *kappa*-enhanced operators to the interval-based model, and we show that the generalised inference problem remains NP-hard if we use discretely subdivided—rather than continuous—intervals. While we did not succeed in establishing NP-hardness for inference in interval-based or other (less general) variants of enhanced networks, such as the networks with enhanced and rich enhanced operators suggested by Renooij and van der Gaag (1999), we argue why a polynomial inference algorithm for these problems is unlikely to exist. This chapter is concluded in Section 6.5.

6.1 Background: Qualitative Probabilistic Networks

A qualitative probabilistic network $\mathcal{Q} = (\mathbf{G}, \Delta)$ is characterised by associating a set Δ of qualitative influences and synergies (Wellman, 1990) with a directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$. A QPN can be seen as an *abstraction* of a family of quantitative probabilistic networks, where the modelled joint probability distribution respects the restrictions imposed by Δ. The influences and synergies in Δ are denoted by signs. For example, a *positive* influence of a node A on its successor B, denoted with $S^+(A, B)$, expresses that higher values for A make higher values for B more likely, regardless of influences of other nodes on B. For binary variables A and B this can be expressed as $\Pr(b \,|\, a\mathbf{x}) - \Pr(b \,|\, \bar{a}\mathbf{x}) \geq 0$ for any joint value assignment $\mathbf{x}$ to the other parents of B in $\mathbf{G}$. For variables with more values, the cumulative conditional probability distribution $F(A \,|\, B)$ is used, where $S^+(A, B)$ expresses that, for all values b_i of B, $F(a_i \,|\, b_i) \leq F(a_j \,|\, b_i)$ if $a_i > a_j$. *Negative* influences, denoted by S^-, and *zero* influences, denoted by S^0, are defined analogously. In the remainder of this chapter we assume, without loss of generality, that all variables are binary.

If an influence is not positive, negative or zero, it is *ambiguous*, denoted by $S^?$. This may be the case when the influence is non-monotone, e.g., $\Pr(b \mid a\mathbf{x_1}) - \Pr(b \,|\, \bar{a}\mathbf{x_1}) > 0$, and $\Pr(b \,|\, a\mathbf{x_2}) - \Pr(b \,|\, \bar{a}\mathbf{x_2}) < 0$ for two different joint value assignments $\mathbf{x_1}$ and $\mathbf{x_2}$ to the other parents of B. In addition, the sign '?' may arise during inference in cases where the actual influence is unknown, i.e., cannot be determined. If this happens, the sign represents our lack of knowledge about that particular influence in the network, rather than the actual influence, and it is therefore desirable to generate as few of these signs as possible. Note that we assume a certain ordering on the values of the variables, in particular that $a > \bar{a}$. In Chapter 7, we will discuss situations where a variable does not have a predefined natural ordering.

Influences can be direct (influences along arcs), indirect (influences along trails) or *induced* (inter-causal influences arising from *product synergies*). In the latter case, the value of one node influences the probabilities of the values of another (not directly connected) node, given a particular value for a third node (Druzdzel and Henrion, 1993*c*). Furthermore, the notion of *additive synergy* is used to capture the joint effect of two nodes on a third, rather than the effect of each node separately. Both product and additive synergies are particularly



$$\begin{array}{llll} \Pr(t) = & 0.65 & & \\ \Pr(s\,|\,t) = & 0.10 & \Pr(r\,|\,t) = & 0.85 \\ \Pr(s\,|\,\overline{t}) = & 0.01 & \Pr(r\,|\,\overline{t}) = & 0.35 \\ \Pr(l\,|\,sr) = & 0.70 & \Pr(l\,|\,s\overline{r}) = & 0.15 \\ \Pr(l\,|\,\overline{s}r) = & 0.75 & \Pr(l\,|\,\overline{s}\overline{r}) = & 0.17 \end{array}$$

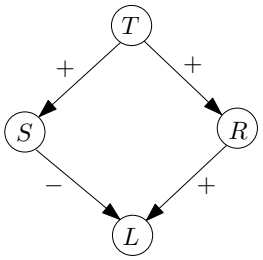


Figure 6.1 The *Radiotherapy* network and its qualitative abstraction

useful when a QPN is used as an intermediate step in the construction of a quantified probabilistic network. They can model constraints on the probability distribution, without the need to specify the exact probabilities. Since product and additive synergies are not used in our hardness proofs, we will not discuss these synergies here; the interested reader can refer to Druzdzel and Henrion (1993*c*) or Wellman (1990).

Example 6.1 (from Renooij (2001)). *We consider a fragment of the Radiotherapy network which models the effect of radiotherapy on life expectancy (Figure 6.1). All variables are binary. Node L models a life-expectancy of at least six weeks, T models the administration of radiotherapy, R models a reduction of the tumour, and S models the development of scar tissue. If a patient receives therapy, the tumour is likely to be reduced. On the other hand, scar tissue may develop. The associated conditional probabilities in the probabilistic network are summarised by '+' signs in the corresponding QPN. The tumour reduction increases the life expectancy of the patient; in contrast, the development of scar tissue decreases the life expectancy. These effects are summarised by the '+' and '−' signs, respectively.*

Various properties hold for qualitative influences, namely *symmetry*, *transitivity*, *composition*, and *associativity* properties, as shown by Wellman (1990) and Renooij and van der Gaag (1999). We define a *trail* t from A to B in a directed graph as a simple path from A to B in the underlying undirected graph, i.e., a list of arcs connecting A to B, regardless of the direction of the arcs. We denote the set of nodes on the trail as $\mathbf{V}(t)$, and the its set of arcs as $\mathbf{A}(t)$. In the remainder, we assume that a trail is *sinkless*, i.e., for a trail t, there are no nodes V_{i-1}, V_i, and $V_{i+1} \in \mathbf{V}(t)$ such that both $(V_{i-1}, V_i) \in \mathbf{A}(t)$ *and* $(V_{i+1}, V_i) \in \mathbf{A}(t)$. We define $\hat{S}^{\delta}(A, B, t)$ as the influence S^{δ}, with $\delta \in \{+, -, 0, ?\}$, from a node A

Symmetry:	$\hat{S}^{\delta}(A,B,t_i) \in \Delta^*$	$\Leftrightarrow$	$\hat{S}^{\delta}(B,A,t_i^{-1}) \in \Delta^*$
Transitivity:	$\hat{S}^{\delta}(A,B,t_i), \hat{S}^{\delta'}(B,C,t_j) \in \Delta^*$	$\Rightarrow$	$\hat{S}^{\delta \otimes \delta'}(A,C,t_i \circ t_j) \in \Delta^*$
Composition:	$\hat{S}^{\delta}(A,B,t_i), \hat{S}^{\delta'}(A,B,t_j) \in \Delta^*$	$\Rightarrow$	$\hat{S}^{\delta \oplus \delta'}(A,B,t_i \parallel t_j) \in \Delta^*$
$\oplus$-Associativity:	$\hat{S}^{(\delta \oplus \delta') \oplus \delta''} \in \Delta^*$	$\Leftrightarrow$	$\hat{S}^{\delta \oplus (\delta' \oplus \delta'')} \in \Delta^*$
$\otimes$-Associativity:	$\hat{S}^{(\delta \otimes \delta') \otimes \delta''} \in \Delta^*$	$\Leftrightarrow$	$\hat{S}^{\delta \otimes (\delta' \otimes \delta'')} \in \Delta^*$

Figure 6.2 Properties of qualitative influences

on a node B along the trail t. Furthermore, we define Δ^* as the transitive closure of Δ, t^{-1} as the trail inverse of t, $t_i \circ t_j$ as the (sinkless) trail concatenation of t_i and t_j, and $t_i \parallel t_j$ as the parallel trail composition of t_i and t_j.

The *symmetry* property states that if $\hat{S}^{\delta}(A,B,t_i) \in \Delta^*$, then also $\hat{S}^{\delta}(B,A,t_i^{-1}) \in \Delta^*$. In the example network, $\Pr(s \mid t) - \Pr(s \mid \bar{t}) = 0.10 - 0.01 \geq 0$. Likewise, using Bayes' rule we can calculate that $\Pr(t \mid s) - \Pr(t \mid \overline{s}) = 0.95 - 0.63 \geq 0$, so the influence of S on T has the same sign as the influence of T on S. This property holds in general. The transitivity property defines the sign of the chained effect between two variables along a trail using the $\otimes$-operator; the composition property defines the combined effect of a variable on another variable along multiple trails using the $\oplus$-operator. The definition of these properties is shown in Figure 6.2, and the semantics of the $\otimes$- and $\oplus$-operators are defined in Figure 6.3. Lastly, we define $\hat{S}^{\delta}(A,B)$ to denote the combined effect of A on B over all trails between A and B.

In our example, we can infer from these properties that the positive effect of therapy on scar tissue also implies a positive effect in the opposite direction (symmetry); that the positive effects of therapy on tumour reduction, and of tumour reduction on life expectancy imply a positive effect of therapy on life expectancy along the trail $\{(T,R),(R,L)\}$ (transitivity); and that the opposite signs of the effects of therapy on life expectancy along the trails $\{(T,S),(S,L)\}$ and $\{(T,R),(R,L)\}$ imply an overall unknown effect (composition). The latter example illustrates the limited usefulness of QPNs when dealing with trade-offs in the network. The effect of therapy on life expectancy can be calculated to be positive, using the probabilities in the network, yet this effect is lost in the abstraction.

$\otimes$	$+$	$-$	0	$?$
$+$	$+$	$-$	0	$?$
$-$	$-$	$+$	0	$?$
0	0	0	0	0
$?$	$?$	$?$	0	$?$

$\oplus$	$+$	$-$	0	$?$
$+$	$+$	$?$	$+$	$?$
$-$	$?$	$-$	$-$	$?$
0	$+$	$-$	0	$?$
$?$	$?$	$?$	$?$	$?$

Figure 6.3 The $\otimes$- and $\oplus$-operators for combining signs

Using these properties of QPNs, an efficient, polynomial-time, inference algorithm was constructed by Druzdzel and Henrion (1993*a*), which was subsequently improved by van Kouwen et al. (2009). The algorithm propagates observed node values to neighbouring nodes, thus determining the effect of an observation for a node O on the other nodes in the network, i.e., it computes $\hat{S}^{\delta}(O, A)$ for all $A \in \mathbf{V}(\mathbf{G})$. The algorithm visits each node at most two times, and therefore halts after a polynomial number of steps, as shown in Druzdzel and Henrion (1993*a*) and confirmed by van Kouwen et al. (2009). Furthermore, while no formal proof has been given yet, it is common knowledge that the algorithm is deterministic.

6.2 Enhanced QPNs

While qualitative influences are useful to model probabilistic influences between nodes in a network, a lot of information is lost in the abstraction. For example, trade-offs cannot be modelled in a qualitative network. In the Radiotherapy network, the administration of radiotherapy increases life expectancy because of the tumour reduction, but decreases life expectancy due to the development of scar tissue. The positive effect of tumour reduction, however, is much larger than the negative effect of scar tissue as the conditional probability table in Figure 6.1 shows. This trade-off cannot be modelled in a QPN because no distinction can be made between strong and weak influences. Therefore, extensions to the QPN model have been suggested that preserve a larger amount of information in the abstraction than the traditional QPN model.

One of these extensions is the *enhanced* formalism proposed by Renooij and van der Gaag (1999). This formalism introduces a notion of relative strength.

$\otimes_e$	$++^j$	$+^j$	$+_?$	0	$-_?$	$-^j$	$--^j$	?
$++^i$	$++^{i+j}$	$+^j$	$+_?$	0	$-_?$	$-^j$	$--^{i+j}$	?
$+^i$	$+^i$	$+^{i+j}$	$+^i$	0	$-^i$	$-^{i+j}$	$-^i$	?
$+_?$	$+_?$	$+^j$	$+_?$	0	$-_?$	$-^j$	$-_?$	?
0	0	0	0	0	0	+	−	0
$-_?$	$-_?$	$-^j$	$-_?$	0	$+_?$	$+^j$	$+_?$	?
$-^i$	$-^i$	$-^{i+j}$	$-^i$	0	$+^i$	$+^{i+j}$	$+^i$	?
$--^i$	$--^{i+j}$	$-^j$	$-_?$	0	$+_?$	$+^j$	$++^{i+j}$	?
?	?	?	?	?	?	?	?	?

Figure 6.4 The enhanced $\otimes_e$-operator

For example, given a specific cut-off value $\alpha \geq 0$, a positive influence can be *strongly* positive, i.e., $\Pr(b\,|\,a\mathbf{x}) - \Pr(b\,|\,\overline{a}\mathbf{x}) \geq \alpha$ for all joint value assignments $\mathbf{x}$ to the other parents of B than A or *weakly* positive, i.e., $\alpha \geq \Pr(b\,|\,a\mathbf{x}) - \Pr(b\,|\,\overline{a}\mathbf{x}) \geq 0$ for all assignments $\mathbf{x}$. The basic '+' and '−' signs are redefined to model weak effects, and the signs '++' and '−−' are introduced to model strong effects. In addition, the signs '$+_?$' and '$-_?$' are used to denote positive or negative influences of unknown strength. The uncertainty about the strength of a sign may arise when combining effects during inference. For example, combining a strong positive with a weak negative effect leads to a positive effect of unknown strength. Furthermore, the signs are augmented with multiplication indices to handle complex dependencies on α as a result of transitive and compositional combinations. Multiplication indices capture the fact that indirect influences (established using transitivity, or chaining of influences) are weaker than direct ones. For example, for a single isolated trail $\{(A, B), (B, C)\}$ with strongly positive influences, that is $\Pr(b\,|\,a) - \Pr(b\,|\,\overline{a}) \geq \alpha$ and $\Pr(c\,|\,b) - \Pr(c\,|\,\overline{b}) \geq \alpha$, we find that $\Pr(c\,|\,a) - \Pr(c\,|\,\overline{a}) \geq \alpha^2$. Using this notion of strength, trade-offs are then modelled by compositions of weak and strong opposite signs. The $\oplus_e$- and $\otimes_e$-operators associated with the transitivity and composition properties in so-called *enhanced* QPNs are shown in Figures 6.4 and 6.5.

Another extension of the QPN model is the *interval-based network* (Renooij and van der Gaag, 2002), where each arc has an associated numerical influence interval rather than a qualitative sign. Such an influence is denoted as $F^{[p,q]}(A, B)$, meaning that $\Pr(b\,|\,a\mathbf{x}) - \Pr(b\,|\,\overline{a}\mathbf{x}) \in [p, q]$, for every joint value assignment $\mathbf{x}$ to the parents of B other than A. Note that, given this definition,

$\oplus_e$	$++^j$	$+^j$	$+_?$	0	$-_?$	$-^j$	$--^j$	$?$
$++^i$	$++^m$	$++^i$	$++^i$	$++^i$	$?$	$a)$	$?$	$?$
$+^i$	$++^j$	$+_?$	$+_?$	$+^i$	$?$	$?$	$d)$	$?$
$+_?$	$++^j$	$+_?$	$+_?$	$+_?$	$?$	$?$	$?$	$?$
0	$++^j$	$+^j$	$+_?$	0	$-_?$	$-^j$	$--^j$	$?$
$-_?$	$?$	$?$	$?$	$-_?$	$-_?$	$-_?$	$--^j$	$?$
$-^i$	$b)$	$?$	$?$	$-^i$	$-_?$	$-_?$	$--^j$	$?$
$--^i$	$?$	$c)$	$?$	$--^i$	$--^i$	$--^i$	$--^m$	$?$
$?$	$?$	$?$	$?$	$?$	$?$	$?$	$?$	$?$

where $m = min(i, j)$,

a) $+_?$, if $i \leq j$; ?, otherwise

b) $+_?$, if $j \leq i$; ?, otherwise

c) $-_?$, if $i \leq j$; ?, otherwise

d) $-_?$, if $j \leq i$; ?, otherwise

Figure 6.5 The enhanced $\oplus_e$-operator

$S^+(A, B) \iff F^{[0,1]}(A, B)$; similar observations hold for S^-, S^0 and $S^?$. We will call the intervals $[-1, 0]$, $[0, 1]$, $[0, 0]$ and $[-1, 1]$ the *unit intervals*, being special cases that correspond to signs in the traditional qualitative networks. The $\otimes_i$- and $\oplus_i$-operators associated with the transitivity and composition properties in interval-based networks are defined in Figure 6.6.

Note that the symmetry and associativity properties of qualitative networks no longer hold in these enhancements. For example, although a positive influence from a node A to B in the direction of the arc also is a positive influence in the opposite direction, the *strength* of this influence cannot be determined and must be specified explicitly. Also, the outcome of the combination of multiple signs may depend on the evaluation order of the operators. For example, $++^1 \oplus_e -^1 \oplus_e +^1$ can evaluate as $(++^1 \oplus_e -^1) \oplus_e +^1 = +_?$ or as $++^1 \oplus_e (-^1 \oplus_e +^1) =?$, depending on the evaluation order of the operators. For enhanced QPNs or interval-based networks, a similar sign-propagation algorithm as in van Kouwen et al. (2009) can be constructed. In contrast to the algorithm for inference in standard QPNs, this algorithm, however, may not run in polynomial time, and the outcome may depend on the evaluation order of the operators (Renooij and van der Gaag, 2008; Renooij and van der Gaag, 2002).

$\otimes_i$	$[r, s]$
$[p, q]$	$[\min X, \max X]$, where $X = \{p \cdot r, p \cdot s, q \cdot r, q \cdot s\}$

$\oplus_i$	$[r, s]$
$[p, q]$	$[p + r, q + s] \cap [-1, 1]$

Figure 6.6 The $\otimes_i$- and $\oplus_i$-operators for interval multiplication and addition

6.3 Complexity of the problem

An interval-based QPN $\mathcal{Q} = (\mathbf{G}, \Delta)$ is not 'merely' a directed acyclic graph and a set of intervals associated with each arc: $\mathcal{Q}$ must have a *model*, i.e., a probabilistic network that satisfies the constraints imposed by the intervals. Inference in these interval-based QPNs appears to be infeasible and indeed no polynomial-time inference algorithm is known. If we remove the restriction that $\mathcal{Q}$ must have a *model*, and allow the instance to be *any* directed acyclic interval graph, we have a more general inference problem on interval-based *pseudo*-QPNs, which allows us to study the complexity of the inference mechanism in an unrestricted manner. In the remainder, we will analyse the complexity of inference in this problem variant. We state the decision variant of this problem, denoted as Pseudo Interval Inference, as follows.

Pseudo Interval Inference
Instance: Let $\mathcal{Q} = (\mathbf{G}, \Delta)$ denote an interval-based pseudo-QPN, in which $A \in \mathbf{V}(\mathbf{G})$ is an observed node and $B \in \mathbf{V}(\mathbf{G}) \setminus \{A\}$ is a non-observed node.
Question: Is there an evaluation order of the influences in Δ such that the influence of A on B, computed using the $\otimes_i$- and $\oplus_i$-operators and given that evaluation order, is a strict subinterval of $[-1, 1]$?

Note that in this problem definition $\mathcal{Q}$ may or may not have an actual model. We will prove NP-hardness of Pseudo Interval Inference by a polynomial-time reduction from 3Sat. A 3Sat instance is a logical formula in conjunctive normal form, that is, a conjunction of clauses, where each clause is a disjunction of literals (variables or their negation). In a 3Sat instance, each clause has exactly three literals. The associated decision problem is whether there exists an assignment to the variables that makes the formula true. This problem is known to be NP-complete (Garey and Johnson, 1979).

Given any 3Sat instance (U, C) consisting of ternary clauses C on Boolean variables U, we construct an interval-based qualitative network $\mathcal{Q}_{(U,C)}$, with designated nodes I and Y. We prove that, upon instantiation of I to $[1, 1]$ (i.e., observing that $I =$ TRUE), an ordering of the influences in $\mathcal{Q}$ exists which results in an influence on Y that is a strict subinterval of $[-1, 1]$, if and only if the corresponding 3Sat instance is satisfiable. To improve readability, in the remainder of this chapter the $\oplus$- and $\otimes$-operators, when used without index, denote operators on intervals as defined in Figure 6.6. In the network, the influence of a node A on a node B along the arc (A, B) is given as an interval; when the interval equals $[1, 1]$ (i.e., $\Pr(b \mid a) = 1$ and $\Pr(b \mid \overline{a}) = 0$) then the interval is omitted in the figures for readability.

As a running example, we construct a network for the following 3Sat instance, introduced in Cooper (1990).

Example 6.2. 3Sat_{ex} *= (U,C), with:*
$U = \{u_1, u_2, u_3, u_4\}$*, and*
$C = \{(u_1 \vee u_2 \vee u_3)$*,* $(\overline{u_1} \vee \overline{u_2} \vee u_3)$*,* $(u_2 \vee \overline{u_3} \vee u_4)\}$*.*

This instance is satisfiable, for example with the truth assignment $u_1 =$ TRUE*,* $u_2 =$ FALSE*,* $u_3 =$ TRUE*, and* $u_4 =$ TRUE*.*

6.3.1 Construction

For each variable u_i in the 3Sat instance, the network contains a fragment Vg_i, called a *variable gadget*, as shown in Figure 6.7. After the instantiation of node I with $[1, 1]$, the influence at node V_4 equals the $\oplus$-combination of $[\frac{1}{2}, 1]$, $[-\frac{1}{2}, \frac{1}{2}]$, and $[-1, -\frac{1}{2}]$, which is either $[-1, \frac{1}{2}]$, $[-\frac{1}{2}, 1]$ or $[-1, 1]$, depending on the order of evaluation. We use the non-associativity of the $\oplus$-operator in this fragment as a non-deterministic choice of assignment of truth values to the variables u_i of the 3Sat instance. As we will see later, an evaluation order that leads to $[-1, 1]$ can be safely dismissed (it will act as a 'falsum' in all the clauses, making both u_i and $\overline{u_i}$ false), so we will concentrate on the influence $[-\frac{1}{2}, 1]$ at node V_4 (which will be our TRUE assignment to the variable u_i) and $[-1, \frac{1}{2}]$ (FALSE assignment) as the two possible choices. We now construct nodes u_i for our 3Sat instance, each with a variable gadget Vg_i as input via an arc with the influence $[1, 1]$. Each such node u_i can thus have either $[-1, \frac{1}{2}]$ or $[-\frac{1}{2}, 1]$ for its

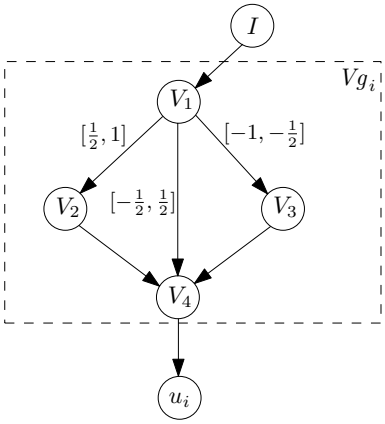


Figure 6.7 *Variable gadget* Vg_i connected to I and u_i

influence, non-deterministically.

Observe that the network fragment is not a fragment of a 'real' interval-based QPN: the chosen structure and intervals are such that no probabilistic network exists that satisfies these constraints.

For each clause C_j in the 3SAT instance, we add a clause fragment Cl_j and connect the variable gadget Vg_i of u_i to Cl_j if u_i occurs in C_j. Figure 6.8 shows the construction of the combined fragments for our example 3SAT instance; Figure 6.9 depicts a single clause fragment. The influence associated with the arc from u_i to the node w_i in the fragment Cl_j is $F^{[p,q]}(u_i, w_i)$, where $[p, q]$ equals $[-1, 0]$ if $\overline{u_i}$ is in C_j, and $[0, 1]$ if u_i is in C_j. Note that $\otimes$-multiplication with $[-1, 0]$ will transform an influence of $[-1, \frac{1}{2}]$ to $[-\frac{1}{2}, 1]$ and vice versa, and that a similar operation with $[0, 1]$ will not change them. We can therefore regard an influence $F^{[-1,0]}$ as modelling negation of the truth assignment for that node u_i. Note that the interval $[-1, 1]$ representing TRUE nor FALSE, will stay the same in both cases.

The clause fragment is shown in Figure 6.9. The influences of the three parent nodes u_i for a clause C_j (each of which has an influence of either $[-1, \frac{1}{2}]$, $[-\frac{1}{2}, 1]$, or $[-1, 1]$) are $\otimes$-multiplied with the arc influence $F_{i,j}$, and then $\oplus$-combined with the instantiation node (with a value of $[1, 1]$), forming literal nodes w_i.

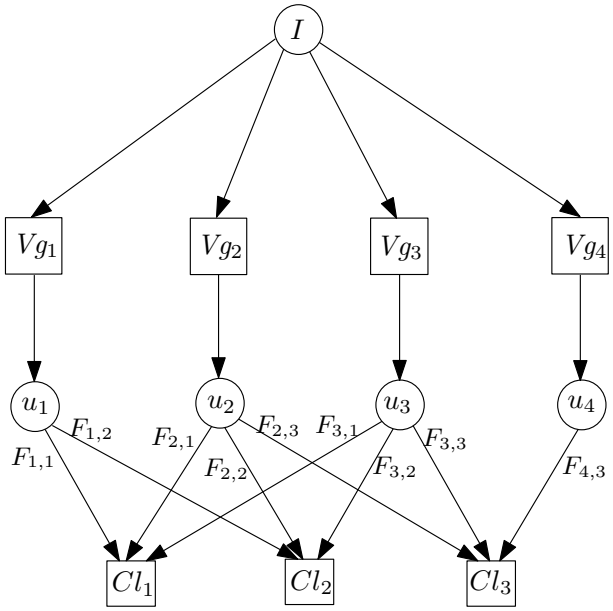


Figure 6.8 The *literal-clause construction*

Note that for an assignment FALSE to a positive literal u_i in C, or an assignment TRUE to a negative literal $\overline{u_i}$, the influence is $[-1, \frac{1}{2}] \oplus [1, 1] = [0, 1]$. For a TRUE assignment to a positive literal or a FALSE assignment to a negative literal, the influence is $[-\frac{1}{2}, 1] \oplus [1, 1] = [\frac{1}{2}, 1]$. Since the $[-1, 1]$ outcome of the variable gadget does not change by multiplication with the $F_{i,j}$ influence, the influence on the literal w_i would become $[-1, 1] \oplus [1, 1] = [0, 1]$, which is the same influence as that for a FALSE assignment to a positive literal or a TRUE assignment to a negative literal. In case of this assignment to the variable u_i, both the literals u_i and $\overline{u_i}$ will contribute the value FALSE to any clause, and thus fail to satisfy it. If such an assignment can satisfy the 3SAT instance, the instance will also be satisfied with the assignment that gives u_i an arbitrary TRUE or FALSE value.

In the clause fragment, the influences associated with the nodes w_i are subsequently multiplied by $[\frac{1}{2}, 1]$ and added using $\oplus$-addition[1] to give the clause intermediate result at node O_j (Figure 6.9). At this point, O_j has an influence of $[\frac{k}{4}, 1]$, where k equals the number of literals which are true in this clause.

[1]Note that the evaluation order is irrelevant at this point.

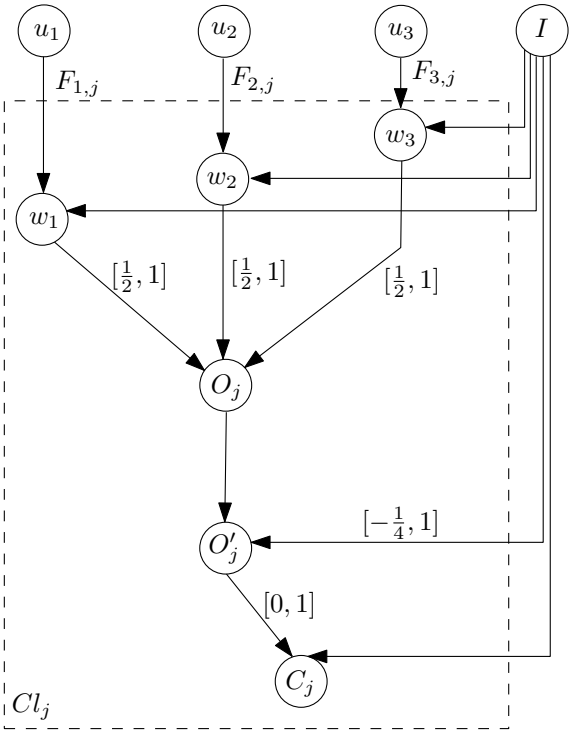


Figure 6.9 The *clause fragment*

The consecutive addition of $[-\frac{1}{4}, 1]$, multiplication by $[0, 1]$ and addition of $[1, 1]$ mimics the *logical or*-operator. Adding $[-\frac{1}{4}, 1]$ to $[\frac{k}{4}, 1]$ will lead to a positive interval if $k \geq 1$ and to the interval $[-\frac{1}{4}, 1]$ if $k = 0$. As a result of the multiplication by $[0, 1]$, the positive interval will be 'relaxed' to $[0, 1]$ for any $k \geq 1$ but remains $[-\frac{1}{4}, 1]$ if $k = 0$. Finally, adding $[1, 1]$ results in an influence for the result node C_j of $[\frac{3}{4}, 1]$ if no literal in the clause was true, and of $[1, 1]$ if one or more literals are true.

We then combine the separate clause variables C_j into a variable D_{n-1}, by adding arcs from each clause to D_{n-1} using intermediate variables D_1 to D_{n-2}. The use of these intermediate variables allows us later to use this construction also in more restricted cases. The interval associated with these edges is $[\frac{1}{2}, 1]$, leading to an interval of $[1, 1]$ in D_{n-1} if and only if all clause variables C_j have an interval of $[1, 1]$ (see Figure 6.10). If one or more clause result nodes have a

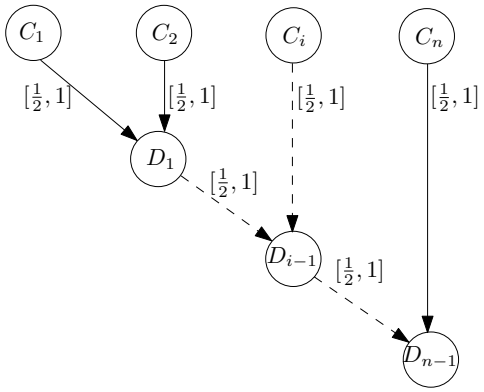


Figure 6.10 Connecting the clauses

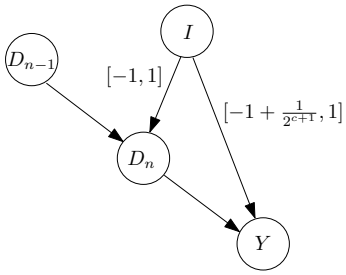


Figure 6.11 Constructing the output node Y

value of $[\frac{3}{4}, 1]$ (i.e., the clause is not satisfied by the variable instantiation), the interval influence in D_{n-1} has a value between $[\frac{3}{4}, 1]$ and $[\frac{2^{c+1}-1}{2^{c+1}}, 1]$ (where c is the number of clauses). Finally, we construct the output node Y by consecutively adding the influence in D_{n-1} to $[-1, 1]$ to obtain D_n, and finally adding this influence to $[-1 + \frac{1}{2^{c+1}}, 1]$ to obtain Y (Figure 6.11). This would result in a strict subinterval of $[-1, 1]$ (namely, $[-1 + \frac{1}{2^{c+1}}, 1]$) if and only if the influence in D_{n-1} is equal to $[1, 1]$, i.e, if all clauses are satisfied. If one or more clauses are not satisfied, then the influence in D_n will be at most $[\frac{2^{c+1}-1}{2^{c+1}}, 1]$ and the consecutive additions will ensure an influence in Y of $[-1, 1]$.

6.3.2 NP-hardness proof

Using the construct presented in the previous section, NP-hardness of Pseudo Interval Inference can be proven as follows.

Theorem 6.3. Pseudo Interval Inference *is* NP*-hard.*

Proof. To prove hardness, we construct a transformation from 3Sat. Let (U, C) be an instance of this problem, and let $\mathcal{Q}_{(U,C)}$ be the interval-based pseudo-QPN constructed from this instance, as described in the previous section. When the node I in $\mathcal{Q}_{(U,C)}$ is instantiated with $[1, 1]$, then I has an influence on Y which is a strict subset of $[1, 1]$ if and only if all nodes C_j have a value of $[1, 1]$, i.e., there exists an ordering of the operators in the variable-gadgets such that at least one literal in C_j is true for every clause C_j. We conclude that (U, C) has a solution with at least one true literal in each clause if and only if the Pseudo Interval Inference-problem has a solution for network $\mathcal{Q}_{(U,C)}$, instantiation $I = [1, 1]$ and output node Y. Since $\mathcal{Q}_{(U,C)}$ can be computed from (U, C) in polynomial time, we have a polynomial-time transformation from 3Sat to Pseudo Interval Inference, which proves NP-hardness of Pseudo Interval Inference. □

6.3.3 On the possible membership of NP

Although Pseudo Interval Inference has been shown to be NP-hard, membership in NP (and, as a consequence, NP-completeness) is still open. To prove membership of NP, one has to prove that if an arbitrary instance is solvable, then there exists a certificate that can be used to verify this claim in polynomial time. A trivial certificate could be a formula, using the $\oplus$- and $\otimes$-operators, influences, and parentheses, describing how the influence on a particular node is calculated from the instantiated node and the characteristics of the network. Unfortunately, such a certificate can become exponentially large, and verifying a claim using this certificate would take time exponential in the size of the network. While the properties of the traditional (i.e., non-enhanced) $\otimes$- and $\oplus$-operators ensure that a node sign can change at most two times upon inference, this no longer holds for the interval operators. From the definition of the $\oplus_i$-operator in Figure 6.6, we can see that a node interval can change with every

update. Thus, the number of interval changes can be as large as the (possibly exponential) number of trails between the instantiation node and the target node.

6.4 Operator variants

In Section 6.2, we discussed several variations of the standard QPN framework, of which the interval-based QPNs were the most general enhancement. We will show that the other enhancements can be related to the interval-based model in a straightforward way using so-called *relaxation schemes*.

If we take a closer look at the $\oplus_e$- and $\otimes_e$-operators defined in Renooij and van der Gaag (1999) (see Figures 6.4 and 6.5) and compare them with the interval operators $\oplus_i$ and $\otimes_i$, we can see that the interval results are sometimes 'relaxed'. We see that symbols representing influences correspond to intervals, but after the application of any operators to these intervals, the resulting interval is sometimes enlarged to an interval that corresponds to one of the available symbols. For example, in the interval model we have $[\alpha, 1] \oplus_i [-1, 1] = [\alpha - 1, 1]$, but, while $[\alpha, 1]$ corresponds to '++' in the enhanced model and $[-1, 1]$ corresponds to '?', combining '++' and '?' using the $\oplus_e$-operator results in a '?'-sign, which corresponds to the interval $[-1, 1]$. When we use enhanced QPNs (rather than interval-based networks), the lower limit $\alpha - 1$ is relaxed to -1 because the actually resulting interval $[\alpha - 1, 1]$ does not correspond to any symbol. To connect the (enhanced) qualitative models to interval-based network models, we now introduce *relaxation schemes* that map the result of every operation within the interval-based framework to the minimally enlarged interval that can be represented by one of the available symbols. Then, both the standard and the enhanced QPN framework can be regarded as special cases of interval-based networks with a relaxation scheme.

Definition 6.4. *(Relaxation scheme) A mapping R is a relaxation scheme if R maps an interval $[a, b]$ to an interval $[c, d]$, denoted as $R([a, b]) = [c, d]$, where $[a, b] \subseteq [c, d]$.*

In standard QPNs, the relaxation scheme (which we will denote as R_I or the *unit scheme*) is defined as:

$$R_I([a,b]) = \begin{cases} [0,1] & \text{if } a \geq 0 \wedge b > 0 \\ [-1,0] & \text{if } a < 0 \wedge b \leq 0 \\ [0,0] & \text{if } a = b = 0 \\ [-1,1] & \text{otherwise.} \end{cases}$$

Similarly, the $\oplus_e$- and $\otimes_e$-operators are embedded in the interval-based framework by the following relaxation schemes, in which m equals $\min(i,j)$ and α is the cut-off value used in the definition of the $\oplus_e$- and $\otimes_e$-operators.

$$R_{\otimes_e}([a,b]) = \begin{cases} [-1,1] & \text{if } a < 0 \wedge b > 0 \\ [a,b] & \text{otherwise.} \end{cases}$$

$$R_{\oplus_e}([a,b]) = \begin{cases} [\alpha^m, 1] & \text{if } a = \alpha^i + \alpha^j \leq b \\ [-1, -\alpha^m] & \text{if } b = -(\alpha^i + \alpha^j) \geq a \\ [0,1] & \text{if } a \leq b = \alpha^i + \alpha^j \\ [-1,0] & \text{if } a = -(\alpha^i + \alpha^j) \leq b \\ [0,1] & \text{if } a = (\alpha^i - \alpha^j) \text{ and } b \geq 0 \text{ and } i < j \\ [-1,0] & \text{if } a = -(\alpha^i - \alpha^j) \text{ and } b \leq 0 \text{ and } i < j \\ [-1,1] & \text{if } a < 0 \text{ and } b > 0 \\ [a,b] & \text{otherwise.} \end{cases}$$

The notion of a relaxation scheme allows us to relate various operators in a uniform way. Apart from the traditional (non-enhanced) operators and the $\oplus_e$- and $\otimes_e$-operators discussed above, similar relaxation schemes can be constructed for, e.g., the *richly* enhanced $\oplus_r$-operator defined in Renooij and van der Gaag (2008) and the *kappa*-enhanced operators defined in Renooij et al. (2003).

We now discuss whether the NP-hardness proof of Pseudo Interval Inference can be extended to other problem variants. In order to represent every possible 3Sat instance, the operators defined by a relaxation scheme must allow the generation of variable gadgets, and retain enough information to discriminate between the cases where zero or more literals in each clause are true. Furthermore, the relaxation scheme must allow the representation of true and false. For a relaxation scheme that effectively divides the interval $[-1,1]$ in discrete blocks with size of a multiple of $\frac{1}{4}$ (such as $R_{\frac{1}{4}}([a,b]) = [\frac{\lfloor 4a \rfloor}{4}, \frac{\lceil 4b \rceil}{4}]$), the construction for the proof is essentially the same as for the general case discussed in section 6.2. This relaxation scheme does not have any effect on the intervals we used in the variable gadget and the clause fragment of $\mathcal{Q}_{(U,C)}$. The network

constructed in the NP-hardness proof for the general case used only intervals $[a, b]$ for which $R_{\frac{1}{4}}([a, b]) = [a, b]$. Furthermore, when connecting the clauses, the possible influences in D_{n-1} are relaxed to $[0, 1]$, $[\frac{1}{4}, 1]$, $[\frac{1}{2}, 1]$, $[\frac{3}{4}, 1]$, and $[1, 1]$, so we can construct Y by consecutively adding the interval in D_{n-1} to $[-1, 1]$ and $[-\frac{3}{4}, 1]$. The following result follows as a corollary.

Corollary 6.5. *The* Pseudo Interval Inference*-problem remains* NP*-hard when relaxation scheme* $R_{\frac{1}{4}}$ *is applied.*

Note that this result shows that we *do not need* the full power of the interval model to prove NP-hardness, i.e., the problem remains NP-hard even if we use discretely subdivided intervals.

The non-associativity of the $\oplus_e$-operator defined in Renooij and van der Gaag (1999) suggests NP-hardness of the inference problem in enhanced QPNs[2] as well. Furthermore, in Renooij and van der Gaag (2008) it was shown that the sign-propagation algorithm for enhanced and rich enhanced models may take exponential time, since the multiplication indices involved with enhanced node signs may need to be updated with every visit. The same holds for the strength factors in kappa-enhanced networks (Renooij et al., 2003).

Although $\oplus_e$ is not associative, it cannot produce results that can be regarded as opposites. For example, the expression $(+{+}^1 \oplus_e +^1 \oplus_e -^1)$ can lead to a positive influence of unknown strength ('$+_?$') when evaluated as $((+{+}^1 \oplus_e +^1) \oplus_e -^1)$ or an unknown influence ('?') when evaluated as $(+{+}^1 \oplus_e (+^1 \oplus_e -^1))$, but never to a negative influence. A transformation from a 3Sat variant might not succeed because of this reason. However, it might be possible to construct a transformation from Pseudo Interval Inference with relaxation scheme $R_{\frac{1}{4}}$, which we leave as a topic of further research.

Lastly, we do not know whether Interval Inference is NP-hard, for a similar reason as mentioned above: it is to the best of our knowledge not possible to make a construction that produces results that are opposites, given the constraint that a probabilistic network must exist that respects the constraints imposed by the graph structure and the intervals. As a side note we remark

[2]That is, the problem of determining whether there exists an evaluation order of the influences in Δ such that the influence of an observation for a node O on another node A, computed using the $\otimes_e$- and $\oplus_e$-operators and given that evaluation order, is not equal to '?'.

that it is likely that determining *whether* a given interval graph constitutes a 'real' interval-based QPN is intractable and is even unlikely to be in NP[3].

6.5 Conclusion

In this chapter, we addressed the computational complexity of inference in interval-based networks and enhanced qualitative probabilistic networks. As a first step, we embedded both standard and enhanced QPNs in the interval-based model using relaxation schemes, and we showed that inference in this general model is NP-hard and remains NP-hard for relaxation scheme $R_{\frac{1}{4}}$, when the restriction that the network constitutes an underlying probabilistic network is removed. In general, inference in interval-based pseudo-networks thus is intractable, like inference in probabilistic networks. Whether inference in interval-based networks (i.e., networks that do have a model) is NP-hard remains open. Propagation algorithms for interval-based networks may sometimes need exponential time, for example when a network has many trails between observation node and output node.

There are two factors that may contribute to the hardness of Pseudo Interval Inference. Firstly, we observe that reasoning in enhanced and interval-based networks is (at least with the current definition of the $\otimes$- and $\oplus$-operators) under-defined: the outcome of the inference process depends on choices during evaluation. Secondly, node intervals can change with every update, in contrast to the standard QPN model where node signs can only change twice, as discussed in Section 6.1. Thus, when there are exponentially many trails between two nodes, an observation may lead to an exponential number of node updates. A similar observation can be made for enhanced, rich enhanced, and kappa-enhanced operators.

Despite these considerations, enhanced and interval-based qualitative models are useful as an intermediate step in the construction of a probabilistic network

[3]For example, take a graph with n nodes $X_1, \ldots, X_n$ such that X_n is a sink and there are arcs from all other nodes to X_n (and no other arcs). We can describe this interval graph using $\mathcal{O}(n)$ bits, yet a description of a corresponding probabilistic network may need $\mathcal{O}(c^n)$ bits for a particular constant c. Thus, in general we cannot give a description of a network as a certificate for the existence of such a network.

(Renooij and van der Gaag, 2002). Nevertheless, it is highly desirable that inference algorithms always give the same result, independent of non-deterministic steps in the algorithm. An *anytime* algorithm, that calculates more specific results when given more time, may balance running time and preciseness of the results of inference according to the needs of the application, and might further facilitate the use of qualitative models to design, validate, analyse, and simulate probabilistic networks.

Chapter 7

Local Monotonicity

In the previous chapter, we discussed qualitative probabilistic networks (QPNs). In these qualitative networks, ambiguities can occur as a result of trade-offs in the original quantitative network which are lost in the abstraction. We have discussed *enhanced* and *interval* QPNs that introduced a notion of strength to prevent information loss due to trade-offs. Loss of information can also occur, however, as a result of the presence of non-monotone influences in the network (Druzdzel and Henrion, 1993*b*). Upon inference in QPNs, non-monotone influences between variables are propagated to all *d-connected* nodes, thus potentially masking all knowledge stored in the network. Various (partial) solutions to this problem have been proposed for networks with binary variables, for example by using context-specific sign-propagation (Renooij et al., 2002) or by introducing situational influences (Bolt et al., 2005). In this chapter, we investigate the possibility of reordering the values of non-binary variables in order to make the

network *locally monotone*.

We start this chapter with a problem motivation in Section 7.1. In Section 7.2, we introduce some notations and definitions. We discuss an algorithm for deciding whether a network can be made locally monotone by variable reordering in Section 7.3, and prove NP-hardness of the optimisation problem in Section 7.4. In Section 7.5, we show that the optimisation problem is hard to approximate as well, and we suggest a branch-and-bound strategy as an exact algorithm in Section 7.6. Some final remarks are made in Section 7.7.

7.1 Motivation

Our approach is illustrated in the probabilistic network in Figure 7.1. This example network models some fictitious knowledge with respect to the presence of sickle cell disease *(SCD)*, gastric ulcer *(GU)* and lung cancer *(LC)*. Apart from these three classification variables, the network models four observable variables, namely ethnic group *(EG)*, blood type *(BT)*, gender *(G)*, and smoking habits *(SH)*. The associated conditional probability tables are shown in Table 7.2. The QPN resulting after qualitative abstraction is shown in Figure 7.3. We assume that all variables are ordered as they appear in the CPTs, with lower ordered values appearing before higher values. Note that four out of eight relations in the network are monotone: G has a negative effect on SH and LC, and SH has positive effects on GU and LC. This information is lost, however, upon the observation that a patient has a gastric ulcer: the non-monotone influence (denoted by) '?' on the active path between G and EG will eventually spread throughout the network and even cancel out the known negative effects (directly and indirectly through SH) of G on LC. While an unknown influence of GU on SH may be correct if the trails between GU and SH model trade-offs in the network that lead to conflicting influences, we will see in the remainder that this is not the case. The information loss is not due to the qualitative abstraction, but purely to the non-monotonicity in the network.

To investigate whether this loss of information can be avoided, we note that while SH for example has a natural ordering (with *non-smoker* $<$ *light-smoker* $<$ *heavy-smoker*), EG, BT, and G lack such an ordering; the chosen ordering is arbitrary. If we would have chosen *Asian* $<$ *White* $<$ *Native American* $<$

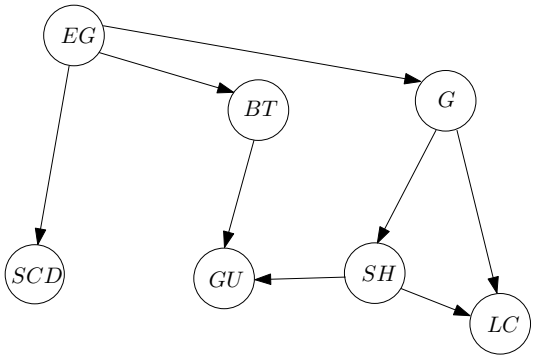


Figure 7.1 A small example network

Hispanic < *African American* as the ordering for EG, then the influence of EG on SCD would be monotone: higher values for EG would make sickle cell disease more likely. In fact, with this ordering for EG and with $O < B < AB < A$ as the ordering for BT, all variables have an ordering such that all influences in the network are monotone and no information is lost upon inference (see Figure 7.4).

The above example demonstrates that it is important to choose variable orderings in such a way that all influences between neighbouring variables are monotone. If this is the case, we define the network to be *locally monotone*. If local monotonicity is not possible, then we should ensure that the number of arcs which induce monotone influences is maximal.

7.2 Monotonicity and variable orderings

The definition of influence sign from the previous chapter assumed an implicit *ordering* on the values of the variables involved. Such an ordering is often natural, e.g., $\bar{a} < a$ and *never* < *sometimes* < *always*. Sometimes, however, the ordering is arbitrary, like an ordering of the values of the variable *Ethnic Group*. Nevertheless, an ordering is required to determine the effect of an influence between two variables and to determine whether the network is locally monotone. Thus, for nodes for which no natural ordering exists, we want to order the val-

Ethnic Group	
Asian	0.04
African Am.	0.13
Native Am.	0.01
Hispanic	0.16
White	0.66

Smoking Habits			
Gender	*Non*	*Light*	*Heavy*
Male	0.63	0.20	0.17
Female	0.71	0.17	0.12

Gender		
Ethnic Group	*Male*	*Female*
Asian	0.51	0.49
African Am.	0.49	0.51
Native Am.	0.50	0.50
Hispanic	0.49	0.51
White	0.50	0.50

Blood Type				
Ethnic Group	*A*	*O*	*AB*	*B*
Asian	0.30	0.43	0.03	0.24
African Am.	0.56	0.32	0.03	0.09
Native Am.	0.44	0.39	0.04	0.13
Hispanic	0.47	0.35	0.04	0.14
White	0.40	0.41	0.04	0.15

Sickle Cell Disease		
Ethnic Group	*No*	*Yes*
Asian	0.995	0.005
African Am.	0.92	0.08
Native Am.	0.99	0.01
Hispanic	0.98	0.02
White	0.99	0.01

Gastric Ulcer						
	Smoking Habits					
Blood	Non		Light		Heavy	
Type	*No*	*Yes*	*No*	*Yes*	*No*	*Yes*
A	0.99	0.01	0.985	0.015	0.97	0.03
O	0.96	0.04	0.94	0.06	0.91	0.09
AB	0.99	0.01	0.985	0.015	0.97	0.03
B	0.99	0.01	0.985	0.015	0.97	0.03

Lung Cancer				
	Gender			
Smoking	Male		Female	
Habits	*No*	*Yes*	*No*	*Yes*
Non	0.98	0.02	0.99	0.01
Light	0.88	0.12	0.92	0.08
Heavy	0.72	0.28	0.83	0.17

Table 7.2 Conditional probability tables for the example network

ues in a way that maximises the number of monotone influences. To facilitate further discussion, we first define an ordering as used in this chapter.

Definition 7.1 (ordering). *For any node X, $\mathbf{E}(X)$ is defined as the set of all permutations of $\Omega(X)$. An* ordering *of X is an element from $\mathbf{E}(X)$. We will often use η and θ to denote orderings, and we will write $E(X) = \eta$ if X has been assigned the ordering η. We use the superscript T to denote a reverse ordering: if $\eta = (x_1 < x_2 < \ldots < x_n)$, then $\eta^T = (x_n < \ldots < x_2 < x_1)$.*

Note that a qualitative influence of X on another node is positive given a particular ordering η for X if and only if it is negative given η^T, and vice versa. Consequently, η and η^T can be considered equivalent when only monotonicity is considered. In the remainder, when $\eta, \theta \in \mathbf{E}(X)$ are assumed distinct, we also assume that $\eta \neq \theta^T$. We use the shorthand notation $x_1 <_\eta x_2$ to denote that $x_1 < x_2$ under ordering η.

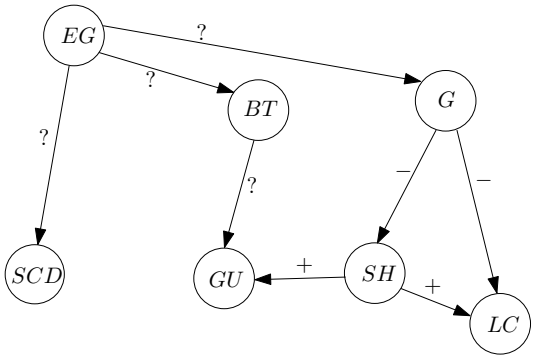


Figure 7.3 Qualitative influences of the network in Figure 7.1

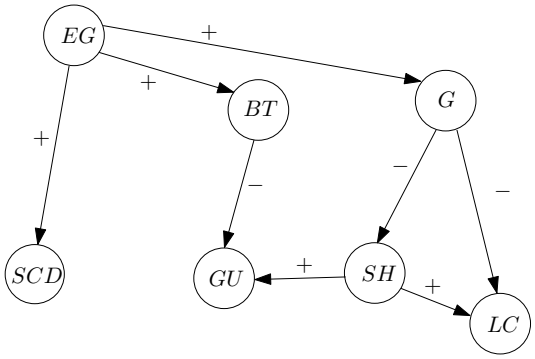


Figure 7.4 The resulting QPN with reordered variables EG and BT

A variable with k values has $k!$ possible orderings. If the number of values per variable is unbounded, therefore, there are exponentially many orderings to consider. In practice, however, the number of values a variable can take is often small. For example, in the Alarm network (Beinlich et al., 1989), the number of values is at most four, and in the Oesophageal network (van der Gaag et al., 2002) it is at most six. In the remainder, we shall assume that k is small and can be regarded a fixed constant.

7.2.1 Monotonicity functions and matrices

We now give the definition of a *monotonicity function* $M_{X,Y} : (\mathbf{E}(X), \mathbf{E}(Y)) \rightarrow \{\text{TRUE}, \text{FALSE}\}$, which determines whether a particular combination of orderings η and θ for two nodes X and Y makes the influence along the arc (X, Y) monotone. From the definition of qualitative influence, we have that when a node Y has more than one parent, say $\pi(Y) = \{X\} \cup \mathbf{Z}$, the influence of X on Y is monotone for a combination of orderings $\eta \in \mathbf{E}(X)$ and $\theta \in \mathbf{E}(Y)$, if the direction of the effect of X on Y is the same (i.e., either non-decreasing or non-increasing) for all values of $\mathbf{Z}$. The monotonicity function for (X, Y) for a *given* joint value assignment $\mathbf{z} \in \Omega(\mathbf{Z})$ will be called a *partial monotonicity function*, written $M_{X,Y|\mathbf{z}}$, to emphasise that the function is *conditional* on that joint value assignment $\mathbf{z}$.

Definition 7.2 ((partial) monotonicity function). *Let X and Y be nodes connected by an arc (X, Y), where Y has additional parents $\mathbf{Z}$. Let $\eta \in \mathbf{E}(X)$ and $\theta \in \mathbf{E}(Y)$. If, with η and θ, the influence between X and Y equals $S^+(X, Y)$ or $S^0(X, Y)$, then the influence is isotone for orderings η and θ, denoted by $M^+_{X,Y}(\eta, \theta)$; likewise, it is antitone for η and θ, denoted by $M^-_{X,Y}(\eta, \theta)$, if it equals $S^-(X, Y)$ or $S^0(X, Y)$. The monotonicity function $M_{X,Y}$ is defined by $M_{X,Y}(\eta, \theta) = M^+_{X,Y}(\eta, \theta) \vee M^-_{X,Y}(\eta, \theta)$. The partial monotonicity function $M_{X,Y|\mathbf{z}}(\eta, \theta)$ evaluates to* TRUE *if $M_{X,Y}(\eta, \theta) =$* TRUE *given the joint value assignment $\mathbf{z}$ of $\mathbf{Z}$.*

Observe that $M_{X,Y}(\eta, \theta) = M_{X,Y}(\eta^T, \theta) = M_{X,Y}(\eta, \theta^T) = M_{X,Y}(\eta^T, \theta^T)$ for all $\eta \in \mathbf{E}(X)$ and $\theta \in \mathbf{E}(Y)$, since $M_{X,Y} = M^+_{X,Y} \vee M^-_{X,Y}$, $M^+_{X,Y}(\eta, \theta) \leftrightarrow M^-_{X,Y}(\eta^T, \theta)$, and $M^+_{X,Y}(\eta, \theta) \leftrightarrow M^-_{X,Y}(\eta, \theta^T)$. Partial monotonicity functions can be combined for multiple joint value assignments of $\mathbf{Z}$. Informally, the combined partial monotonicity function for joint value assignments $\mathbf{z_1}$ and $\mathbf{z_2}$ evaluates to TRUE for a particular combination of orderings, if the individual partial monotonicity functions are either all isotone or all antitone for that combination.

Definition 7.3 (combining partial monotonicity functions). *Consider the arc (X, Y) as defined before, with $\mathbf{Z} = \pi(Y) \setminus \{X\}$. Then, for $\delta \in \{+, -\}$ and arbitrary joint value assignments $\mathbf{z_1}, \mathbf{z_2}$ to $\mathbf{Z}$, we have that $M^\delta_{X,Y|\{\mathbf{z_1},\mathbf{z_2}\}}(\eta, \theta) = M^\delta_{X,Y|\mathbf{z_1}}(\eta, \theta) \wedge M^\delta_{X,Y|\mathbf{z_2}}(\eta, \theta)$.*

Observe that $M^{\delta}_{X,Y}(\eta,\theta) = \bigwedge_{\mathbf{z}\in\mathbf{Z}} M^{\delta}_{X,Y|\mathbf{z}}(\eta,\theta)$.

With every monotonicity function $M_{X,Y}$, a binary $|\,\mathbf{E}(X)\,| \times |\,\mathbf{E}(Y)\,|$ matrix $\mathbf{M_{X,Y}}$ is associated, called the *monotonicity matrix* of $M_{X,Y}$. In these matrix $\mathbf{M_{X,Y}}$, a cell $(\eta,\theta), \eta \in \mathbf{E}(X), \theta \in \mathbf{E}(Y)$, equals 1 if $M_{X,Y}(\eta,\theta) =$ TRUE and 0 otherwise. Similarly, a *partial monotonicity matrix* $\mathbf{M_{X,Y|z}}$ is associated with a partial monotonicity function. We will often illustrate these matrices using a grid, where shaded areas denote monotone combinations of orderings in $\mathbf{E}(X)$ and $\mathbf{E}(Y)$. We will use monotonicity matrices in our algorithm for determining whether a network can be made locally monotone.

7.2.2 Properties of monotonicity matrices

The definition of monotonicity implies that some matrices over $\mathbf{E}(X)$ and $\mathbf{E}(Y)$ cannot constitute a monotonicity matrix for (X,Y), thus restricting the number of orderings per variable that can lead to monotone influences in the network. More in particular, we have that two columns of a monotonicity matrix are either equal or disjoint, as we will see below. This property can be used to speed up the construction of a monotonicity matrix. We observe that if the influence between X and Y is monotone for two distinct orderings $\eta, \eta' \in \mathbf{E}(X)$ of X and an ordering $\theta \in \mathbf{E}(Y)$ of Y, then there are at least two equal columns in the conditional probability table, i.e., there are values x_i and $x_j, x_i \neq x_j$, of node X such that $\Pr(y_k \,|\, x_i, \mathbf{z}) = \Pr(y_k \,|\, x_j, \mathbf{z})$ for all $y_k \in \Omega(Y)$ and all joint value assignments $\mathbf{z}$ to parents of Y other than X. But then, $M_{X,Y}(\eta,\theta') = M_{X,Y}(\eta',\theta')$ for *all* orderings $\theta' \in \mathbf{E}(Y)$, i.e., the columns in the monotonicity matrix are equal.

Lemma 7.4. *Let $X, Y, \mathbf{Z}$ be such that $\pi(Y) = \{X\} \cup \mathbf{Z}$. Let $\eta, \eta' \in \mathbf{E}(X)$ be distinct orderings for X and let $\theta \in \mathbf{E}(Y)$. Then, if $M_{X,Y}(\eta,\theta) \wedge M_{X,Y}(\eta',\theta) =$ TRUE, then there exist $x_i, x_j \in \Omega(X)$, with $x_i \neq x_j$, such that $\Pr(y_k \,|\, x_i, \mathbf{z}) = \Pr(y_k \,|\, x_j, \mathbf{z})$ for all $y_k \in \Omega(Y), \mathbf{z} \in \Omega(\mathbf{Z})$.*

Proof. Without loss of generality, we assume that $M_{X,Y}(\eta,\theta)$ and $M_{X,Y}(\eta',\theta)$ both indicate isotone influences of X on Y. Since η and η' are distinct, there exist i and j such that for $x_i, x_j \in \Omega(X)$ $x_i <_{\eta} x_j$ and $x_j <_{\eta'} x_i$. For the cumulative distribution function $F(Y \,|\, X, \mathbf{Z})$ for each joint value assignment $\mathbf{z}$

to $\mathbf{Z}$, it then follows from $M_{X,Y}(\eta, \theta) = \text{TRUE}$ and the assumption of isotonicity that $F(Y \mid x_i, \mathbf{z}) \geq F(Y \mid x_j, \mathbf{z})$; from $M_{X,Y}(\eta', \theta) = \text{TRUE}$ and the isotonicity it also follows that $F(Y \mid x_j, \mathbf{z}) \geq F(Y \mid x_i, \mathbf{z})$. But then $F(Y \mid x_i) = F(Y \mid x_j)$ and $\Pr(y_k \mid x_i) = \Pr(y_k \mid x_j)$ for all $y_k \in \Omega(Y)$. □

From the lemma, we obtain the following property of monotonicity matrices that we will use in our algorithm:

Corollary 7.5. *Let* $\eta, \eta' \in \mathbf{E}(X)$ *and* θ, $\theta' \in \mathbf{E}(Y)$. *Then,* $\exists\theta[M_{X,Y}(\eta, \theta) \wedge M_{X,Y}(\eta', \theta) = \text{TRUE}] \rightarrow \forall\theta'[M_{X,Y}(\eta, \theta') = M_{X,Y}(\eta', \theta')]$.

7.3 Local Monotonicity

A probabilistic network $\mathcal{B}$ is *locally monotone* if all signs in its corresponding QPN $\mathcal{Q}$ are either positive, negative, or zero. $\mathcal{B}$ is *maximally locally monotone* if the number of non-monotone signs in $\mathcal{Q}$ cannot be (further) reduced by reordering the values of one or more variables. In this section, we discuss the problem of determining whether a network can be made locally monotone. Maximising the number of positive, negative, and zero signs in a network will be discussed in Section 7.4. We start with formalising the local monotonicity problem.

Local Monotonicity
Instance: A probabilistic network $\mathcal{B} = (\mathbf{G}, \Gamma)$.
Question: Is there an ordering $\eta_X \in \mathbf{E}(X)$ for each $X \in \mathbf{V}$ such that by applying these orderings in the network, $\mathcal{B}$ is locally monotone?

In the remainder of this section, we present an algorithm for Local Monotonicity, with a running time which is exponential only in the treewidth of the graph, assuming that the number of values per variable is fixed. We show how monotonicity matrices can be computed, and how these matrices actually induce a instance of the Constraint Satisfaction-problem. We present an algorithm solving such instances, using dynamic programming, and illustrate the algorithm on an example network.

7.3.1 Constructing monotonicity matrices

Let $X, Y, \mathbf{Z}$ be such that $\pi(Y) = \{X\} \cup \mathbf{Z}$. Computing a partial monotonicity matrix in a straightforward manner for the arc (X, Y), given a joint value assignment $\mathbf{z}$ to $\mathbf{Z}$, takes $\mathcal{O}(k^2 \cdot (k!)^2)$ time, since there are $k!$ orderings for both ends of the arc and computing the sign of the influence between X and Y (conditional on $\mathbf{z}$) takes a time which is quadratic in k. If all other nodes in the network have an arc towards Y, then computing the (unconditional) monotonicity matrix for (X, Y) takes $\mathcal{O}(k^{2n} \cdot (k!)^2)$ time. Note that this is (for small k) still polynomial in the size of the network, since the CPT of Y has k^n entries.

The running time can be reduced to $\mathcal{O}(k^n \cdot k!)$ in the best case by exploiting the following observation. If the influence between X and Y is monotone for two distinct orderings $\eta, \eta' \in \mathbf{E}(X)$ and an ordering $\theta \in \mathbf{E}(Y)$, then it follows from Corollary 7.5, that two columns in a monotonicity matrix are either equal or disjoint. It suffices to observe that there exists a $\theta \in \mathbf{E}(Y)$ such that $M_{X,Y}(\eta, \theta) = M_{X,Y}(\eta', \theta) = \text{TRUE}$ to conclude that we can copy the entire column of the monotonicity matrix without actually computing it. Using this observation, we can often reduce the complexity of calculating the monotonicity matrix: in the best case, all columns are equal and we conclude this at the first cell of every column.

7.3.2 Allowed sets and arc constraints

The monotonicity matrices induce constraints on the orderings that can be chosen for each variable such that the network is locally monotone. We discriminate between *universal* and *non-universal* constraints: if the constraints induced by the monotonicity matrix $\mathbf{M}_{X,Y}$ are universal, then there exist subsets $\mathbf{E}^+(X) \subseteq \mathbf{E}(X)$ and $\mathbf{E}^+(Y) \subseteq \mathbf{E}(Y)$ such that $M_{X,Y}(\eta, \theta)$ evaluates to TRUE if and only if $\eta \in \mathbf{E}^+(X)$ and $\theta \in \mathbf{E}^+(Y)$. If all arcs (Y, X) and (X, Z), with $Y \in \pi(X)$ and $Z \in \sigma(X)$, entering or leaving X induce only universal constraints, then an ordering $E(X) \in \{\eta \mid (\bigcap_{Y \in \pi(X)} M_{Y,X}(\mathbf{E}(Y), \eta)) \cap (\bigcap_{Z \in \sigma(X)} M_{X,Z}(\eta, \mathbf{E}(Z))) = \text{TRUE}\}$ can never violate local monotonicity of the network.

Of course, not all constraints are universal. Using $\pi_f(X)$ and $\sigma_f(X)$ to denote the parents and children of X, respectively, such that $(Y \in \pi_f(X), X)$ and

$(X, Z \in \sigma_f(X))$, respectively, are arcs with universal constraints, we will denote the set $\{\eta \mid (\bigcap_{Y \in \pi_f(X)} M_{Y,X}(\mathbf{E}(Y), \eta)) \cap (\bigcap_{Z \in \sigma_f(X)} M_{X,Z}(\eta, \mathbf{E}(Z))) = \text{TRUE}\}$ by $\mathcal{E}$, and call it the *allowed set*. Note that, if all arcs inducing *non*-universal constraints for X would be removed, the allowed set consists of all orderings of $\Omega(X)$ that can be validly chosen, in other words, there exists a network $\mathbf{G}' = (\mathbf{V}, \mathbf{A}')$ where $\mathbf{A}'$ is the (possibly empty) set of arcs inducing universal constraints, such that the allowed set $\mathcal{E}_X$ of X is the set of orderings that can be chosen for X without leading to a violation of local monotonicity of $\mathbf{G}'$.

For each arc (X, Y) inducing *non*-universal constraints, the set of tuples $(\eta \in \mathbf{E}(X), \theta \in \mathbf{E}(Y))$ with $M_{X,Y}(\eta, \theta) = \text{TRUE}$, denoted as $\mathcal{A}_{X,Y}$, will be called the *arc constraints* for (X, Y). The combination of allowed sets and arc constraints fully defines the orderings that do not violate local monotonicity. If the allowed set is empty for any variable in $\mathbf{V}$ or if there is an arc in $\mathbf{A}$ whose arc constraints cannot be satisfied, then there is no set of orderings for which $\mathcal{B}$ is locally monotone.

7.3.3 Constructing a constraint satisfaction problem instance

Using the allowed sets and arc constraints, we can formulate a local monotonicity problem on a network $\mathcal{B}$ as a *constraint satisfaction problem* (CSP) $\mathcal{I}$, and formulate a dynamic programming algorithm on $\mathcal{I}$. We will show that the constructed CSP has a solution if and only if $\mathcal{B}$ is locally monotone. A CSP $\mathcal{I}$ is defined as a 3-tuple $\langle \mathbf{V}, \mathbf{D}, \mathbf{C} \rangle$, where $\mathbf{V}$ denotes a set of variables, $\mathbf{D}$ denotes a domain of values, and $\mathbf{C}$ denotes a set of constraints. Each constraint is a tuple $\langle t, R \rangle$, where t denotes a tuple of variables and R denotes a relation over these variables composed of tuples of values. A solution to $\mathcal{I}$ is a function f from $\mathbf{V}$ to $\mathbf{D}$ such that for each constraint $\langle t, R \rangle$, with $t = \langle V_1, \ldots, V_m \rangle$, $\langle f(V_1), \ldots, f(V_m) \rangle \in R$. The *primal graph* $\mathbf{G}$ of $\mathcal{I}$ is an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $(V_1, V_2) \in \mathbf{E}$ if and only if there is a constraint $\langle t, R \rangle \in \mathbf{C}$ with $V_1, V_2 \in t$.

We now construct a CSP $\mathcal{I}$ for a probabilistic network $\mathcal{B}$. A trivial definition of the variable set $\mathbf{V}$ of $\mathcal{I}$ consists of a variable per node in the network, with as domain the allowed set of that node. We can further reduce the variable set, however. Variables that are not endpoints of any arc with arc constraints can be assigned any ordering from their allowed set and thus can be eliminated from

our CSP. So, we define $\mathbf{V}$ as $\{X \mid \exists_{Y \in \pi(X) \cup \sigma(X)} \mathcal{A}_{Y,X} \neq \varnothing\}$. The domain $\mathbf{D}$ of these variables includes, for each variable X, the allowed set $\mathcal{E}_X$. The constraint set $\mathbf{C}$ consists of all tuples $\langle t, R \rangle$ where t are the endpoints of an arc (X, Y) and R is a union of the constraints in $\mathcal{A}_{X,Y}$. If the thus constructed CSP $\mathcal{I}$ has a solution, then $\mathcal{B}$ is locally monotone, and vice versa. Observe that $\mathcal{I}$ has no solution if the allowed set $\mathcal{E}_X$ is empty for any variable X.

To solve $\mathcal{I}$, we construct a tree-decomposition T_C of the underlying graph induced by the variables in $\mathbf{V}$ and the constraints in $\mathbf{C}$. Assuming that T_C is a nice tree-decomposition (see Section 2.2), we will present a dynamic programming algorithm on T_C that decides Local Monotonicity in $\mathcal{O}(m \cdot (k!)^w \cdot \|\mathcal{B}\|)$, where w is the treewidth of the primal graph of $\mathcal{I}$, m is the number of arcs in $\mathcal{B}$, and k is the maximal number of values per node in $\mathbf{V}$.

7.3.4 Deciding Local Monotonicity

In our algorithm, we visit the bags in the tree-decomposition T_C from the leaves to the root. For each bag i, we construct a set $\mathcal{S}_i$ of possible combinations of orderings for the variables in that bag, according to the following rules:

- leaf nodes: Suppose i is a leaf node. By definition, $\mathbf{X_i}$ has exactly one variable X. We set $\mathcal{S}_i$ equal to the allowed set $\mathcal{E}_X$ for that variable.
- insert nodes: Suppose i is an insert node. Then $\mathbf{X_i}$ inserts one variable X to the set of variables in its child j. We set $\mathcal{S}_i$ equal to the subset of the Cartesian product of $\mathcal{S}_j$ and $\mathcal{E}_X$ that satisfies the arc constraints $\mathcal{A}_{X,Y}$ and $\mathcal{A}_{Y,X}$ for all variables $Y \in \mathbf{X_j}$.
- forget nodes: Suppose i is a forget node. Then $\mathbf{X_i}$ is equal to $\mathbf{X_j}$ minus one variable X. Let $\mathcal{S}_i$ be such that $\mathcal{S}_j$ is the Cartesian product of $\mathcal{S}_i$ and $\mathcal{E}_X$, i.e., we 'forget' the elements in $\mathcal{E}_X$.
- join nodes: Suppose i is a join node. Then i has two children j and k, with $\mathbf{X_i} = \mathbf{X_j} = \mathbf{X_k}$. We set $\mathcal{S}_i$ to the intersection of $\mathcal{S}_j$ and $\mathcal{S}_k$.

If, during the iteration over the bags, the set $\mathcal{S}$ for some bag $\mathbf{X_i}$ becomes empty, then the network is not locally monotone; otherwise, a satisfying ordering for

all variables exists. Such an ordering can then be found by a slightly adapted constructive algorithm which we will not further discuss here.

Lemma 7.6. *The algorithm presented above correctly decides* Local Monotonicity.

Proof. Assume that, for a network $\mathcal{B}$, an ordering can be chosen for each variable X such that $\mathcal{B}$ is locally monotone. Naturally, such an ordering must be contained in the allowed set of the variable, otherwise monotonicity is violated. In the dynamic program, we iterate over all bags i in the tree-decomposition to determine a set $\mathcal{S}_i$ of possible combinations of orderings for the variables in i. Note that in the analysis only the INSERT NODES are relevant: if i is a LEAF NODE, then it contains only one variable X and $\mathcal{S}_i = \mathcal{E}_X$; in FORGET and JOIN NODES, no additional constraints are imposed on $\mathcal{S}_i$. If i is an INSERT NODE, it is easy to see that $\mathcal{S}_i$ consists of all possible combinations of orderings of the variables in i. If an ordering exists that makes the sub-network induced by the variables in i locally monotone, the algorithm will put it in $\mathcal{S}_i$; if no such ordering exists, $\mathcal{S}_i$ will be empty. □

For a complexity analysis, again only the work at the INSERT NODES is relevant. Observe that there are $(k!)^{|\mathbf{X_j}|}$ elements in $\mathcal{S}_j$. There are at most $k!$ possible orderings in the allowed set of the introduced variable $\mathcal{E}_X$, and there are at most $|\ \mathbf{X_j}\ |$ arcs (and thus sets of arc constraints) from Y to variables in $\mathbf{X_j}$. Due to the symmetry of the monotonicity matrices, we can construct the possible combinations in X_i by testing all $(k!) \cdot (k!)^{|\mathbf{X_j}|} = (k!)^{|\mathbf{X_j}|+1} = (k!)^{|\mathbf{X_i}|}$ elements of the Cartesian product against the arc constraints in $\mathbf{X_j}$, which can be done in $\mathcal{O}(m \cdot (k!)^w)$. Calculating the monotonicity schemes and the allowed sets for all variables can be done in $\mathcal{O}((k!)^2 \cdot n \cdot \|\mathcal{B}\|)$ and calculating the arc constraints in $\mathcal{O}((k!)^2 \cdot m)$, hence the running time of the algorithm is $\mathcal{O}(m \cdot (k!)^w \cdot \|\mathcal{B}\|)$, which is exponential only in w under the assumption that k is a constant.

7.3.5 An example network

In Figure 7.5 a small example network $\mathbf{G}_{\text{ex}}$ with monotonicity matrices is given to illustrate the algorithm. For simplicity, we assume that in every variable $X \in \mathbf{G}_{\text{ex}}$ only the orderings $x_1, \ldots, x_4$ are relevant. Combinations of orderings

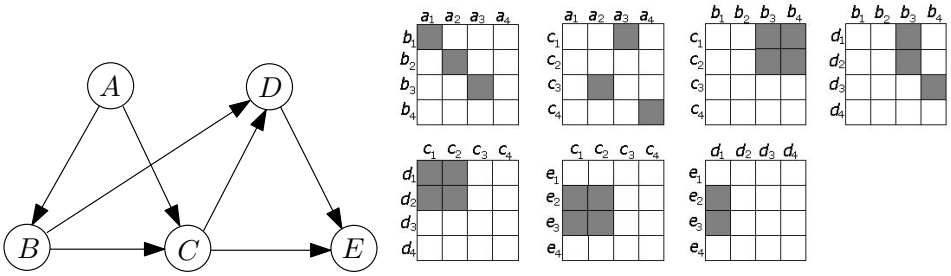
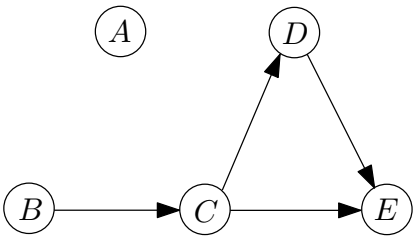


Figure 7.5 Example graph $\mathbf{G}_{\text{ex}}$ with monotonicity matrices



$$\begin{aligned}\mathcal{E}_A &= \{a_1, a_2, a_3, a_4\}\\ \mathcal{E}_B &= \{b_3, b_4\}\\ \mathcal{E}_C &= \{c_1, c_2\}\\ \mathcal{E}_D &= \{d_1\}\\ \mathcal{E}_E &= \{e_2, e_3\}\end{aligned}$$

Figure 7.6 Arcs with universal constraints and allowed sets in $\mathbf{G}_{\text{ex}}$

that lead to a monotone influence are shaded in the monotonicity matrices. For example, (A, B) is monotone for ordering $E(A) = a_1$ and $E(B) = b_1$, but not for $E(A) = a_1$ and $E(B) = b_2$.

To construct a tree-decomposition T_C of this example network, we first construct the allowed sets of all variables. In the initial situation, the allowed set for each variable X is the set $\mathbf{E}(X)$. In the next step, we calculate for each node the allowed set, by considering arcs that induce universal constraints and intersecting the original allowed sets of both vertices covered by these arcs with the appropriate rows (for arcs entering X) respectively columns (for arcs leaving X). This step is illustrated in Figure 7.6 where the arcs with universal constraints and the allowed sets induced by the monotonicity matrices of these arcs are shown.

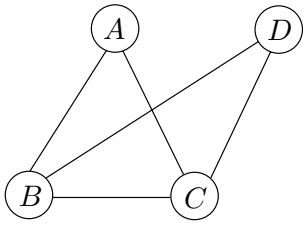


Figure 7.7 The underlying graph $\mathbf{G_U}$ of $\mathbf{G}_{\text{ex}}$

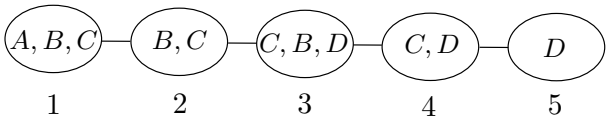


Figure 7.8 A tree-decomposition of $\mathbf{G_U}$

Furthermore, there are arcs which induce arc constraints. In $\mathbf{G}_{\text{ex}}$ these arcs are (A, B), (A, C) and (B, D) and the corresponding arc constraints are as follows:

- $\mathcal{A}_{A,B} = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$
- $\mathcal{A}_{A,C} = \{(a_2, c_3), (a_3, c_1), (a_4, c_4)\}$
- $\mathcal{A}_{B,D} = \{(b_3, d_1), (b_3, d_2), (b_4, d_3)\}$

The variables that are endpoints of arcs with arc constraints are A, B, C and D and they induce the underlying graph in Figure 7.7. From this graph, we construct a nice tree-decomposition as in Figure 7.8. We calculate, for each bag (1 to 5), the possible combinations of orderings of the variables in that bag, starting with the leaf node.

i	node type	possible combinations	comments
5	leaf	(d_1)	allowed set of D
4	insert	(d_1, c_1), (d_1, c_2)	universal constraint
3	insert	$(d_1, c_1, b_3), (d_1, c_2, b_3)$	b_1, b_2, b_4 are impossible
2	forget	$(c_1, b_3), (c_2, b_3)$	just forget d_1
1	insert	(a_3, c_1, b_3)	c_2 is impossible, solution found!

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

7.4 Max-Local Monotonicity

In this section, we formalise the problem of optimising the number of monotone arcs, and show that deciding whether there exists an ordering for a network $\mathcal{B}$ such that at least q arcs in the underlying QPN $\mathcal{Q}$ have '+', '−', or '0' signs is NP-complete, i.e., infeasible in general if $\mathsf{P} \neq \mathsf{NP}$. Furthermore, we prove that this problem is hard to approximate as well. We first give a formal definition of the decision version of this problem.

Max-Local Monotonicity
Instance: Probabilistic network $\mathcal{B} = (\mathbf{G}, \Gamma)$, where m denotes the number of arcs in $\mathbf{A}(\mathbf{G})$, and let q be a positive integer such that $q \leq m$. Let $\mathbf{Q}$ denote the QPN induced by $\mathcal{B}$.
Question: Is there an ordering $E(X)$ for each $X \in \mathbf{V}$ such that the number of monotone signs in $\mathcal{Q}$ is at least q?

We assume that the ordering of all variables is arbitrary, i.e., can be chosen at will. Note that a network where a natural ordering is given for some nodes can be translated into an equivalent network without such nodes, in which the number of monotone arcs will be optimal if and only if that particular ordering is chosen. For example, if the ordering of a node C with values $\{low, mid, high\}$ and degree d is defined to be $low < mid < high$, we can enforce this condition by adding $d+1$ dummy nodes D_i with $\Omega(D_i) = \{\text{TRUE}, \text{FALSE}\}$ and arcs from C to these nodes that are monotone only if C has the ordering $low < mid < high$. For example $\Pr(D = \text{TRUE} \,|\, C = low) = 0.2$, $\Pr(D = \text{TRUE} \,|\, C = mid) = 0.4$, $\Pr(D = \text{TRUE} \mid C = high) = 0.6$. 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.

We now reduce Max-Local Monotonicity from the Graph 3-Colourability problem, defined as follows in Garey and Johnson (1979).

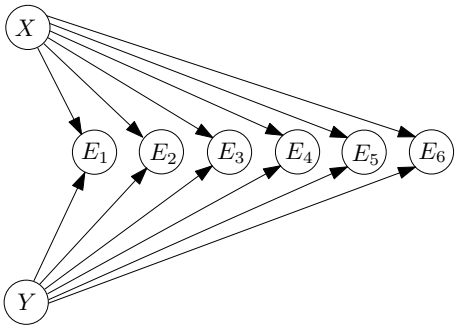


Figure 7.9 Construction with 6 extra nodes

Graph 3-Colourability
Instance: An undirected graph $G = (\mathbf{V}, \mathbf{E})$.
Question: Is there 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 coloured with three colours such that no adjacent nodes have the same colour?

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be an instance of Graph 3-Colourability. From the undirected graph $\mathbf{G}$, we construct the directed graph $\mathbf{G}' = (\mathbf{V}', \mathbf{A})$ as follows (see Figure 7.9) to obtain the Max-Local Monotonicity-instance $(\mathcal{B} = (\mathbf{G}', \Gamma), q)$.

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

Note that we replace every edge in the original graph with an instance of the subnetwork in Figure 7.9. We label the orderings for the nodes in $\mathbf{V}'$ as follows.

- $\eta_1 = x_2 < x_1 < x_3$.

E_i	E_1	E_2	E_3	E_4	E_5	E_6
$E(X)$	$\{\eta_1, \eta_2\}$	$\{\eta_1, \eta_2\}$	$\{\eta_1, \eta_3\}$	$\{\eta_1, \eta_3\}$	$\{\eta_2, \eta_3\}$	$\{\eta_2, \eta_3\}$
$E(E_i)$	$\{\eta_2, \eta_3\}$	$\{\eta_1, \eta_3\}$	$\{\eta_1, \eta_2\}$	$\{\eta_2, \eta_3\}$	$\{\eta_1, \eta_2\}$	$\{\eta_1, \eta_3\}$

Table 7.10 Combinations of $E(X)$ and $E(E_i)$ that lead to monotone influences. For example, the arc (X, E_1) is monotone if X has ordering η_1 or η_2, and E_1 has ordering η_2 or η_3

- $\eta_2 = x_1 < x_3 < x_2$.
- $\eta_3 = x_1 < x_2 < x_3$.

Note that $\mathbf{E}(X) = \{\eta_1, \eta_2, \eta_3\}$ for all nodes X.

Now, for all nodes $E_i, i = 1, \ldots, 6$, we construct a conditional probability table such that $M_{X,E_i}(E(X), E(E_i)) =$ TRUE if and only if the orderings for $E(X)$ and $E(E_i)$ are as in Table 7.10, and $M_{Y,E_i}(E(Y), E(E_i)) =$ TRUE if and only if $E(Y) = E(E_i)$. The conditional probability table for node E_1 is given in Table 7.11; the other tables can be generated likewise.

We now claim that, in the constructed sub-network, there is a maximum of eight arcs out of twelve monotone if $E(X) = E(Y)$, and nine arcs monotone if $E(X) \neq E(Y)$. We assume, without loss of generality, that $E(Y) = \eta_1$. If we choose $E(E_i) = \eta_1$ for all E_i, then all six outgoing arcs from Y to E_i induce monotone influences. Now there are two cases:

- $E(X) = \eta_1$. There are *two* monotone influences: (X, E_2) and (X, E_3). Monotonicity is violated in the other children.
- $E(X) = \eta_2$ or η_3. There are *three* monotone influences: either (X, E_2), (X, E_5) and (X, E_6); or (X, E_3), (X, E_5) and (X, E_6).

Note that there is no ordering for X that will lead to *more* than three monotone influences, and thus we can have at maximum nine monotone influences for each subnetwork. We will use this construct to prove NP-completeness.

$\mathbf{Pr}(E_1 \mid x_1, Y)$			
	e_1	e_2	e_3
y_1	0.42	0.30	0.28
y_2	0.28	0.44	0.28
y_3	0.28	0.30	0.42

$\mathbf{Pr}(E_1 \mid x_2, Y)$			
	e_1	e_2	e_3
y_1	0.44	0.28	0.28
y_2	0.28	0.44	0.28
y_3	0.28	0.28	0.44

$\mathbf{Pr}(E_1 \mid x_3, Y)$			
	e_1	e_2	e_3
y_1	0.44	0.28	0.28
y_2	0.28	0.44	0.28
y_3	0.28	0.28	0.44

Table 7.11 Conditional probability table for node E_1 with arcs (X, E_1) and (Y, E_1)

Theorem 7.7. Max-Local Monotonicity *is* NP*-complete.*

Proof. Membership of NP is trivial. Using a certificate that consists of orderings for all vertices, we can easily test in polynomial time whether at least q arcs are monotone. To prove NP-hardness, we construct a transformation from the Graph 3-Colourability 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 that is constructed from this instance, as described above. If $\mathbf{G}$ would be 3-colourable, then we could asign a colour $c_i, i = 1, \ldots, 3$ to every node such that no adjacent nodes have the same colour. But then, we can assign each variable the ordering η_i such that, for each sub-network in $\mathbf{G}'$, there are exactly 9 arcs monotone, and thus $9 \cdot \mid \mathbf{E} \mid$ arcs in $\mathbf{G}'$ are monotone. On the other hand, if at least $9 \cdot \mid \mathbf{E} \mid$ arcs in $\mathbf{G}'$ are monotone, then all nodes X and Y that were adjacent in $\mathbf{G}$, have different orderings and thus could be assigned a different colour: just pick colour c_i for ordering η_i. 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-Colourability to the Max-Local Monotonicity problem, which proves NP-hardness of the latter. □

Note that the given construction implies that the problem remains NP-complete if the maximal number of values per node is limited to three.

7.5 Non-Approximability

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

7.5.1 Approximation preserving reductions and APX-hardness

We will give some background on approximation problems and reductions, and refer the interested reader to textbooks like Ausiello et al. (1998) or overview articles like Crescenzi (1997). Here we are concerned with NP-optimisation (NPO) problems; without loss of generality we assume maximisation (rather than minimisation) problems. An NP-optimisation problem is characterised by the set of its instances (recognisable in polynomial time) and a measure m on the solution space. If y is a (feasible) solution for instance x, then $m(x, y)$ 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 $R(x, y) = \frac{\mathbf{opt}(x)}{m(x,y)}$.

While the best solution for an NPO-problem can be hard to find, sometimes an algorithm can be constructed with *performance guarantee* r which will find a solution within a factor r of the optimum. 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; we call $A(x)$ an r-approximate solution.

A relevant distinction exists between NPO-problems that can be approximated with *some* ratio (class APX) and NPO-problems that can in polynomial time be approximated with *every* ratio (class PTAS). We have that P $\subseteq$ PTAS $\subseteq$ APX $\subseteq$ NPO, and these inclusions are strict unless P = NP(Ausiello et al., 1998).

Definition 7.8 (from (Ausiello et al., 1998).). *For a problem* P *, an algorithm A is a polynomial-time approximation scheme (PTAS) for* P *if, for any*

instance x of P *and any rational value $r > 1$, A when applied to input (x, r) returns an r-approximate solution of x in time polynomial in $\mid x \mid$.* PTAS *is the class of* NPO*-problems that admit a polynomial-time approximation scheme.* APX *is the class of all* NPO*-problems* P *such that, for some $r > 1$, there exists a polynomial-time r-approximate algorithm for* P *.*

If a problem P is APX-hard, then it is unlikely that P has a PTAS, since this would imply P = NP. We use an *approximation-preserving reduction* to prove APX-hardness of Max-Local Monotonicity. In particular, we will use an *L-reduction* to reduce Max-Local Monotonicity from the known APX-hard problem Max-3-Colourability, defined as follows.

Max-3-Colourability
Instance: Undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$; let F be the set of all functions $f : \mathbf{V} \to \{1, 2, 3\}$.
Question: What is the maximum number of bichrome edges (i.e., edges where the endpoints have a different colour) for any $f \in F$?

An L-reduction from a problem P to a problem Q is a reduction that preserves membership in PTAS, and is defined as follows.

Definition 7.9 (L-reduction). *Let A and B be algorithms that compute solutions to* P *and* Q *, respectively, and let f and g be functions such that $A(x) = g(B(f(x)))$ for all instances x of* P *. The tuple (f, g) will be called an L-reduction if there exist α, β such that:*

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

Furthermore, if an L-reduction from P to Q exists, and P is APX-hard, then also Q is APX-hard. We will show that the reduction from (the decision variants of) Max-3-Colourability to Max-Local Monotonicity is actually a L-reduction when we consider the functional versions of these problems.

Theorem 7.10. Max-Local Monotonicity *is* APX*-hard.*

Proof. We show that MAX-3-COLOURABILITY L-reduces to MAX-LOCAL MONOTONICITY; since MAX-3-COLOURABILITY is APX-hard, APX-hardness of MAX-LOCAL MONOTONICITY follows. Firstly, we show that there exists a constant α such that $\mathbf{opt}_{\text{M3C}}(x) \geq \mathbf{opt}_{\text{MLM}}(f(x))/\alpha$. We observe that, for any graph $\mathbf{G}$, $\mathbf{opt}_{\text{M3C}}(\mathbf{G}) \geq \frac{2}{3}m$, since an algorithm that colours the nodes greedily, giving each node the colour that occurs least often among its (already coloured) neighbours, makes monochrome at most $\frac{1}{3}$ of the edges that become completely coloured in this step. Consequently, the resulting colouring has at least $\frac{2}{3}m$ bichrome edges.

Let $f(\mathbf{G})$ denote the probabilistic network constructed in Section 7.4. 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 ordering; our function g colours each node in $\mathbf{G}$ with the ordering of the corresponding node in $f(\mathbf{G})$. Thus, for $\alpha = 13\frac{1}{2}$, $\mathbf{opt}_{\text{M3C}}(\mathbf{G}) \geq \mathbf{opt}_{\text{MLM}}(f(\mathbf{G}))/\alpha$,

Furthermore, it is clear that, for any graph $\mathbf{G}$, the absolute error in the computed colouring is equal to the absolute error in the computed MAX-LOCAL MONOTONICITY-instance, i.e., $\mathbf{opt}_{\text{M3C}}(\mathbf{G}) - m(\mathbf{G}, g(I)) = \mathbf{opt}_{\text{MLM}}(\mathbf{G}) - m(f(\mathbf{G}), I))$. We conclude that MAX-3-COLOURABILITY L-reduces to MAX-LOCAL MONOTONICITY, hence, MAX-LOCAL MONOTONICITY is APX-hard. □

7.6 A branch-and-bound algorithm

A trivial algorithm for MAX-LOCAL MONOTONICITY (try all possible combination of orderings in the network, count the number of monotone edges for this combination, and return the maximum) takes $\mathcal{O}((k!)^n)$ time. 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, i.e., MAX-LOCAL MONOTONICITY may be in APX. 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 Wolsey and Nemhauser (1988)). In such a strategy, the set of possible solutions is partitioned (the branch step), and upper (or lower, for minimisation 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).

7.6.1 Initial heuristic - a lower bound

A lower bound on the number of monotone influences can be calculated in polynomial time (for a fixed number of values per node k). To compute a lower bound heuristic, we consider only arcs which induce universal constraints. For a network $\mathbf{G}$ we construct $\mathbf{G'} = (\mathbf{V}, \mathbf{A'})$ where $\mathbf{A'}$ is the (possibly empty) set of arcs with universal constraints, and the allowed set of all $X \in \mathbf{V}$ is the set of orderings 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 $\mathcal{E}_X$ to $\mathbf{E}(X)$ for all $X \in \mathbf{V'}$ 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 $\mathcal{E}$ 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 $\mathbf{G}$.

7.6.2 Branching and bounding

Using this lower bound, we consecutively branch on the possible orderings 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 ordering of the variable we branch on (i.e., the allowed set is a singleton, corresponding with the branch value) and calculate how many arcs with universal constraints remain monotone in the network. This value is added to the number of arcs with non-universal constraints; 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 arcs with non-universal constraints 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 trade off between the tightness of the bounds - and thus the number and depth of the branches - and the time needed to calculate such bounds.

This branch-and-bound algorithm has been implemented and tested with the ALARM-network as benchmark. It turned out that the algorithm can find optimal orderings, provided that the number of values per node is small, the number of arc constraints is limited, and the number of variables without a natural ordering is small. On the ALARM-network (which fits these criteria), an optimal ordering was found within a matter of seconds; however, our heuristic lower bound may be too loose (and the domains too large) for larger networks to perform well, and indeed in larger networks the algorithm performs badly (J. Nederlof, personal communication). For networks where some nodes have a large range of possible values, this approach will be infeasible, even when tighter bounds are used. Other methods must be used in such cases to calculate or approximate an optimal solution.

7.6.3 An example

We will use the graph in Figure 7.12 as a example to sketch our branch-and-bound algorithm. We assume that, for every variable, only four orderings are relevant; again we will denote a particular ordering with indexed lower case variables, e.g., $\mathbf{E_C} = \{c_1, c_2, c_3, c_4\}$. On the right part of Figure 7.12, the monotonicity matrices for the arcs in the graph are shown. For example, (A, B) is monotone if $E(A) = a_1$ and $E(B) = b_2$.

We start with the heuristic lower bound as proposed in Section 7.6.1. The arcs with universal constraints 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 $E(C) = c_1$, $E(C) = c_2$, $E(C) = c_3$, and $E(C) = c_4$, terminating branches with

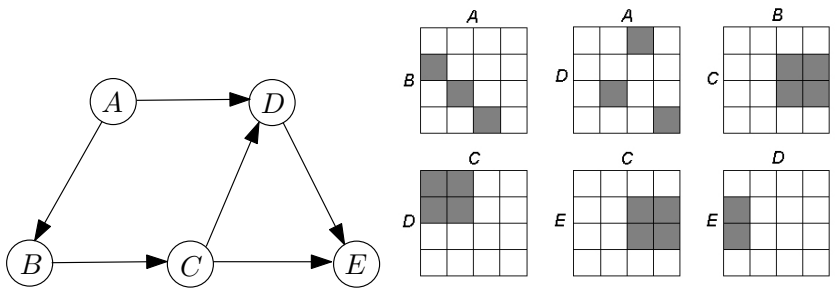


Figure 7.12 The graph and monotonicity matrices used in our example

an upper bound lower than three. Eventually, the algorithm will find the optimal solutions $\{E(A) = a_3, E(B) = b_4, E(C) = c_3, E(D) = d_1, E(E) = e_2 \vee e_3\}$.

7.7 Conclusion

In this chapter, we studied the problem of optimising the number of monotone arcs in a network, by reordering the values of variables in the network, and thus minimising the number of '?'s in the corresponding QPN. We showed that this is an NP-hard problem, and hard to approximate as well, even when the number of values per variable is at most three. The infeasibility of this problem may not be surprising, given the co-$\mathsf{NP}^{\mathsf{PP}}$-hardness of determining whether a network is *globally* monotone, as discussed in Chapter 4. We proposed a branch-and-bound approach to calculate optimal orderings and discussed preliminary findings of an implementation of this algorithm.

Part III

Inference

Chapter 8

Efficient Algorithms and Treewidth

In Section 2.4 we discussed the EXACT INFERENCE-problem of calculating the probability $\Pr(x_i \mid \mathbf{e})$ for a particular value x_i of X_i given evidence $\mathbf{e}$, as well as its decision variants INFERENCE and POSITIVE INFERENCE. We have seen that any probability can be calculated from a network, using Bayes' rule (reviewed in Chapter 2) and the fact that $\Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr(X_i \mid \pi(X_i))$ and $\Pr(X) = \sum_{y_i \in \Omega(Y)} \Pr(X \wedge y_i)$. However, the number of terms in the summation can grow exponentially large. As we have seen, more efficient inference algorithms have been developed, like the *Junction Tree* (Lauritzen and Spiegelhalter, 1988) and *Bucket Elimination* (Dechter, 1999) algorithms, but these algorithms also have a running time that is worst-case exponential in the size of the network. In Chapter 2 we reviewed the proofs that EXACT INFERENCE, INFERENCE and POSITIVE INFERENCE are #P-complete, PP-complete, and NP-complete, respectively. Nevertheless, for probabilistic networks whose moralised

graphs have *bounded treewidth*, computing the posterior probability of a variable in a network using these algorithms takes time, exponential *only* in the treewidth of the moralised graph, and linear in the number of variables. In this chapter, we show that a small treewidth of the moralised graph is actually a *necessary* condition for a network[1] to render inference efficient by an algorithm accepting arbitrary inference instances, under the assumption that the so-called *Exponential Time Hypothesis* (ETH) holds. In other words, we show that no algorithm can exist that solves arbitrary inference instances with unbounded treewidth in polynomial time, unless the ETH fails. Nevertheless, there might be some specific structural graph property (that we are unaware of), for example in the *arc orientation*, that may be exploited by an algorithm to solve *particular* instances in polynomial time.

Whether or not small treewidth is a necessary condition for algorithms to run in polynomial time has also been investigated recently for binary constraint-satisfaction and graph-homomorphism problems (Grohe, 2007) and for inference in undirected graphical models (Chandrasekaran et al., 2008). Using conventional assumptions in computational complexity theory[2], they showed that these problems have no algorithm solving instances with unbounded treewidth in polynomial time. Marx (2007) proved a subexponential lower bound on the running time for the constraint-satisfaction and graph-homomorphism problems with unbounded treewidth, assuming that the ETH holds. These results are summarised in Table 8.1.

In this chapter, we introduce *treewidth-preserving reductions* and show that a polynomial-time treewidth-preserving reduction from Constraint Satisfaction to Positive Inference exists. This proves that if an algorithm for Positive Inference exists that can solve arbitrary instances with large treewidth in subexponential time, then this algorithm can also solve such instances of Constraint Satisfaction in subexponential time, which in turn would contradict the ETH according to Marx' (2007) result. This result also holds for Inference and Exact Inference, since a solution to instances of these problems also yields a solution to a corresponding Positive Inference

[1] Note that we are interested in properties of the graph, rather than of the represented probability distribution. There exists a trivial probability distribution (the uniform distribution where the probability distribution of each variable is independent of its parents) for every network in which inference is trivial.

[2] In particular the assumption that $\mathsf{FPT} \neq \mathsf{W}[1]$, respectively that $\mathsf{NP} \not\subseteq \mathsf{P/poly}$.

Study	Topic	Assumptions	Reduction	Result
Grohe (2007)	CSP, graph homomorphism	FPT $\neq$ W[1], unbounded variable cardinality	k-Clique	no polyn. algorithms for instances with unbounded treewidth
Marx (2007)	CSP, graph homomorphism	ETH, unbounded variable cardinality	n-variable 3Sat	$f(\mathbf{G})^{\omega(\frac{\text{tw}(\mathbf{G})}{\log \text{tw}(\mathbf{G})})}$ lower bound for algorithms solving instances with unbounded treewidth
Chandrasekaran et al. (2008)	inference in graphical models	NP $\neq$ P/poly, Graph Minor Hypothesis	non-uniform Max2Sat	no polyn. algorithms for instances with unbounded treewidth and arbitrary potential distribution
current study	inference in probabilistic networks	ETH, unbounded variable cardinality	treewidth-preserving reduction from CSP	$f(\mathbf{G})^{\omega(\frac{\text{tw}(\mathbf{G})}{\log \text{tw}(\mathbf{G})})}$ lower bound for algorithms solving arbitrary instances with unbounded treewidth

Table 8.1 Comparison between various lower bound results

instance. Treewidth-preserving reductions, as a means to prove conditional[3] lower bounds, may be of independent interest.

We conclude from our result that Marx' (2007) conditional lower bound of $f(\mathbf{G})^{\omega(\frac{\text{tw}(\mathbf{G})}{\log \text{tw}(\mathbf{G})})}$ for the Constraint Satisfaction problem also holds for inference on probabilistic networks, showing that the algorithms discussed in Section 2.4 are optimal, up to a logarithmic factor in the exponent, for arbitrary networks. In Section 8.1, we will sketch our approach, and we present our main results in Section 8.2. These results will be discussed in Section 8.3.

8.1 Approach

In Marx' (2007) paper, the following result with respect to the Constraint Satisfaction-problem was proven.

Theorem 8.1 (CSP Lower Bound). *If there is a recursively enumerable class*

[3] I.e., under the assumption that the ETH holds.

$\mathcal{G}$ of graphs with unbounded treewidth $\mathrm{tw}(\mathbf{G})$ *and a computable function f such that CSP($\mathcal{G}$) can be decided by an algorithm running in time $f(\mathbf{G}) \cdot \|\mathcal{I}\|^{o(\frac{\mathrm{tw}(\mathbf{G})}{\log \mathrm{tw}(\mathbf{G})})}$ for instances $\mathcal{I}$ with a primal graph $\mathbf{G} \in \mathcal{G}$, then the* ETH *fails.*

This result also holds for CSPs with binary relations, i.e., relations between no more than two variables; in the remainder, we assume that all CSP instances are binary.

The *Exponential Time Hypothesis* (ETH), introduced by Impagliazzo and Paturi (2001), states that there exists a constant $c > 1$ such that deciding any 3Sat instance with n variables takes at least $\Omega(c^n)$ time. Note that the ETH is a stronger assumption than the assumption that $\mathsf{P} \neq \mathsf{NP}$. A subexponential but not polynomial-time algorithm for 3Sat (i.e., with a similar running time as the General Number Field Sieve algorithm for Integer Factorisation, see Lenstra et al. (1990)) would contradict the ETH but not $\mathsf{P} \neq \mathsf{NP}$.

In the remainder of this chapter, we build on Marx' (2007) result. We will reduce Constraint Satisfaction to Positive Inference, using a polynomial-time, treewidth-preserving reduction. More specifically, given an instance $\mathcal{I}$ of Constraint Satisfaction, we construct in polynomial time an instance $\mathcal{P} = (\mathcal{B}, C, c, \mathbf{E}, \mathbf{e})$ of Positive Inference with the same treewidth (up to an additive constant) such that a solution to $\mathcal{P}$ yields also a solution to $\mathcal{I}$. This means intuitively that, if an algorithm A exists that solves arbitrary instances of Positive Inference with high treewidth in subexponential time, then we can construct an algorithm B solving instances of Constraint Satisfaction with high treewidth in subexponential time, thus contradicting the ETH. We formalise our result in the following main theorem, which we will prove in Section 8.2.

Theorem 8.2 (Inference Lower Bound). *If there is a computable function f such that an arbitrary instance $\mathcal{P} = (\mathcal{B}, C, c, \mathbf{E}, \mathbf{e}, q)$ with moralised graph $\mathbf{G_M}$ of the* Positive Inference *problem on probabilistic networks can be decided by an algorithm running in time $f(\mathbf{G_M}) \cdot \|\mathcal{P}\|^{o(\frac{\mathrm{tw}(\mathbf{G_M})}{\log \mathrm{tw}(\mathbf{G_M})})}$, then the* ETH *fails.*

8.2 Complexity of Inference

It is often desired that a reduction between decision problems not only preserves polynomial time decidability, but also preserves some other property. For example, parsimonious reductions (Garey and Johnson, 1979) preserve the number of solutions, and approximation-preserving reductions (Ausiello et al., 1998) preserve approximation results. We introduce a reduction on graphical structures that preserves a structural property of a problem instance, namely their *treewidth* (see Section 2.2). Since high treewidth is often an indicator of intractability of a problem, it is desirable that a reduction preserves the treewidth of an instance.

Definition 8.3 (Polynomial-time treewidth-preserving reductions). *Let A and B be problems such that treewidth is defined on instances of both A and B. A is polynomial-time treewidth-preserving reducible to B (denoted $A \leq^p_{\text{tw}} B$), if there exists a polynomial-time computable function g and a linear function l such that for all instances $x : x \in A \iff g(x) \in B$ and $\text{tw}(g(x)) = l(\text{tw}(x))$. We call such a reduction (g, l) a* tw*-reduction.*

We show that Constraint Satisfaction is tw-reducible to Positive Inference, using a polynomial time treewidth-preserving reduction. We will first formally review the Constraint Satisfaction problem, which was introduced in Chapter 7.

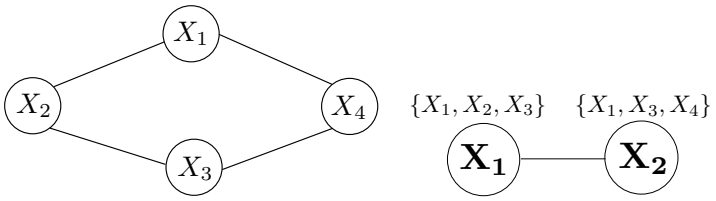
Constraint Satisfaction
Instance: $(\mathbf{V}, \mathbf{D}, \mathbf{C})$, where $\mathbf{V}$ denotes a set of variables, $\mathbf{D}$ a domain of values, and $\mathbf{C}$ a set of constraints $\langle\ t, \mathbf{R}\rangle$, where t is a pair of variables from $\mathbf{V}$ and $\mathbf{R}$ is a binary relation over $\mathbf{D}$.
Question: Is there a function f from $\mathbf{V}$ to $\mathbf{D}$ such that each constraint is satisfied, that is, for each constraint $\langle t, R\rangle$ with $t = (v_1, v_2)$, $(f(v_1), f(v_2)) \in R$?

Given a Constraint Satisfaction instance $\mathcal{I} = \langle \mathbf{V}, \mathbf{D}, \mathbf{C}\rangle$, there is a straightforward way to construct a probabilistic network $\mathcal{B}_\mathcal{I} = (\mathbf{G_I}, \Gamma)$ that simulates $\mathcal{I}$, which will be described below.

However, care must be taken to ensure that the construction preserves treewidth. We will first show how to construct a network $\mathcal{B}_\mathcal{I}$ that captures the information

from the Constraint Satisfaction instance $\mathcal{I}$, and then show how the tree-decomposition of $\mathcal{B}_\mathcal{I}$ can be used to construct a network $\mathcal{B}'_\mathcal{I}$ with the same treewidth as the primal graph of $\mathcal{I}$. This network $\mathcal{B}'_\mathcal{I}$ includes a designated variable A_1 with values true and false, for which we will then show that in $\mathcal{B}'_\mathcal{I}$, $\Pr(A_1 = \text{true}) > 0$ if and only if $\mathcal{I}$ has a solution.

We will illustrate the construction for the example problem $\mathcal{I}_{\text{ex}} = \langle \mathbf{V}, \mathbf{D}, \mathbf{C} \rangle$, with $\mathbf{V} = \{X_1, X_2, X_3, X_4\}, \mathbf{D} = \{a, b, c\}$, and $\mathbf{C} = \{ \langle (X_1, X_2), \{(a,a),(b,a)\}\rangle, \langle (X_1, X_4), \{(a,a),(a,b),(b,a),(c,a)\}\rangle, \langle (X_2, X_3), \{(a,b),(b,a),(b,c),(c,b)\}\rangle, \langle (X_3, X_4), \{(b,a),(b,b)\}\rangle \}$. Note that $\mathcal{I}_{\text{ex}}$ is satisfiable; an example solution f_{ex} sets X_1 to b, X_2 to a, X_3 to b, and X_4 to a. The primal graph of $\mathcal{I}_{\text{ex}}$ is given in Figure 8.2(a) and the associated tree-decomposition in Figure 8.2(b).



(a) Primal graph (b) Tree-decomposition

Figure 8.2 The primal graph (a) and associated tree-decomposition (b) of CSP instance $\mathcal{I}_{\text{ex}}$

For every variable $X_i \in \mathbf{V}$ in $\mathcal{I}$, we introduce a root node X_i to $\mathcal{B}_\mathcal{I}$ with the domain D for its values, with a uniform distribution. For every relation $R_j \in \mathbf{R}$ in $\mathcal{I}$, we add a node R_j to $\mathcal{B}_\mathcal{I}$, with the variables that occur in the relation as parents, and values true and false. We set $\Pr(R_j = \text{true} \mid \mathbf{x}) = 1$ for any combination of values $\mathbf{x}$ for its parents such that $\mathbf{x}$ is a tuple in R_j, and we set $\Pr(R_j = \text{true} \mid \mathbf{x}) = 0$ if $\mathbf{x}$ is *not* a tuple in R_j. See Figure 8.3 for the graph of the network $\mathcal{B}_{\mathcal{I}_{\text{ex}}}$ constructed from the example $\mathcal{I}_{\text{ex}}$.

With respect to the treewidth of the thus constructed graph $\mathcal{B}_\mathcal{I}$ we observe the following. Standard arguments from the theory of treewidth show that the treewidth of the moralised graph of $\mathcal{B}_\mathcal{I}$ equals $\max(2, \text{tw}(\mathcal{I}))$: the nodes of the form R_j are simplicial in the moralised graph, i.e., they are adjacent to a complete set of nodes, and for any simplicial node V with degree d in a graph

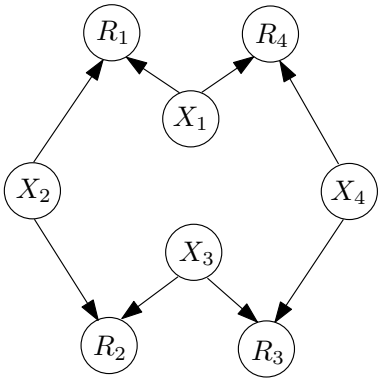


Figure 8.3 Graph $\mathbf{G}_{\text{ex}}$ constructed from CSP instance $\mathcal{I}_{\text{ex}}$

$\mathbf{G}$, the treewith of $\mathbf{G}$ equals the maximum of d and the treewidth of $\mathbf{G} \setminus V$ (see e.g. Bodlaender et al., 2005).

To join the information from the nodes representing the various relations of $\mathcal{I}$, we add extra variables to $\mathcal{B}_{\mathcal{I}}$ that mimic an 'and'-operator, which is necessary since a solution to $\mathcal{I}$ must satisfy every constraint in $\mathcal{I}$. This has to been done with care to avoid an exponential blow-up of the treewidth of the moralised graph $\mathbf{G_M}$ of $\mathcal{B}_{\mathcal{I}}$. Such a blow-up might appear, for example, if we just add one designated node A with all $R_j \in \mathbf{R}$ as its parents, or if we construct a log-deep binary tree to connect the relations in $\mathbf{R}$ with A. We now use the structure of the tree-decomposition of $\mathbf{G_M}$ to construct $\mathcal{B}'_{\mathcal{I}}$ from $\mathcal{B}_{\mathcal{I}}$.

Let $\mathbf{G_M}$ be the moralised graph of $\mathcal{B}_{\mathcal{I}}$, and let $\mathbf{T_G}$ be a tree-decomposition of $\mathbf{G_M}$ which is rooted and in which every node has at most two children; we denote the number of nodes of $\mathbf{T_G}$ with m. Both requirements are met if $\mathbf{T_G}$ is a *nice* tree-decomposition. As we have seen in Chapter 2, every graph has a nice tree-decomposition; moreover a nice tree-decomposition can be computed in time $\mathcal{O}(f(\text{tw}(\mathbf{G_M})) \cdot \|\mathbf{G_M}\|)$ for a particular computable function f (Bodlaender, 1997; Kloks, 1994). We assume in the proof of Theorem 8.4 that the tree-decomposition is nice: *niceness* imposes more constraints on a tree-decomposition than actually needed in our construction, yet is used for reasons of convenience. We use a non-nice tree-decomposition in our example, however, since a nice tree-decomposition of $\mathbf{G_M}$ would simply be too large to fit as an example.

We now obtain $\mathcal{B}'_{\mathcal{I}}$ by adding new variables $A_1, \ldots, A_m$ and arcs (R_j, A_k) to $\mathcal{B}_{\mathcal{I}}$ as follows. For every node k in $\mathbf{T_G}$, we add a node A_k to $\mathbf{G}$, and for every edge (k, l) (where k is the parent of l) in $\mathbf{T_G}$, we add an arc (A_l, A_k). Furthermore, we add arcs (R_j, A_k), for every node R_j that is contained in bag $\mathbf{X_k}$. Every node $A_i, i = 1 \ldots m$ gets a CPT that corresponds to a logical 'and' of its parents, i.e., $\Pr(A_i = \text{TRUE} \,|\, \pi(A_i)) = 1$ if and only if all parents of A_i have the value TRUE, and 0 otherwise; if A_i has no parents, then we set $\Pr(A_i = \text{TRUE}) = 1$. Note that eventually all nodes R_j are 'chained' together in A_1, and $\Pr(A_1 = \text{TRUE}) > 0$ if $\Pr(\bigwedge_j R_j = \text{TRUE}) > 0$. We then make a tree-decomposition $\mathbf{T'_G}$ of the moralised graph $\mathbf{G'_M}$ of $\mathcal{B}'_{\mathcal{I}}$, by enhancing any bag $\mathbf{X_k}$ in $\mathbf{T_G}$ with the variable A_k and the variables A_l that are contained in the bags of the children of k.

A tree-decomposition (with at most two children per node) of the moralised graph of our example network $\mathcal{B}_{\mathcal{I}_{\text{ex}}}$ is given in Figure 8.4. We assume that X_1 is the root of the tree. This tree-decomposition has six nodes so we add nodes $A_1, \ldots, A_6$ to $\mathbf{G}_{\text{ex}}$ (Figure 8.5). For the first bag, we add A_1 and A_2 and add the arcs (A_2, A_1) (since $\mathbf{X_1}$ is the parent of $\mathbf{X_2}$ in the tree-decomposition) and (R_1, A_1) (since R_1 is contained in bag $\mathbf{X_1}$). For consecutive bags $\mathbf{X_i}$, we similarly add nodes $A_3, \ldots, A_6$, arcs $(A_3, A_2), (A_4, A_2), (A_5, A_4)$, and (A_6, A_4), and arcs from the relation nodes R_j in bags $\mathbf{X_i}$ to A_i. In the thus constructed fragment, every node A_i either has a CPT that corresponds to the truth table of a logical 'and' operator on its parents, or $\Pr(A_i = \text{TRUE}) = 1$ for nodes without incoming arcs. The resulting tree-decomposition of $\mathcal{B}_{\mathcal{I}_{\text{ex}}}$ is given in Figure 8.6.

We claim that, for any instance $\mathcal{I}$ of Constraint Satisfaction, in the thus constructed probabilistic network $\mathcal{B}_{\mathcal{I}}$, $\Pr(A_1 = \text{TRUE}) > 0$ if and only if $\mathcal{I}$ is satisfiable; furthermore we claim that the treewidth of the moralised graph of $\mathcal{B}_{\mathcal{I}}$ is equal, up to a linear term, to the treewidth of the primal graph of $\mathcal{I}$.

Theorem 8.4. Constraint Satisfaction *tw-reduces to* Positive Inference.

Proof. To show that the construction described above indeed gives a treewidth-preserving polynomial-time reduction f from Constraint Satisfaction to Positive Inference, we need to prove that f maps instances x of Constraint Satisfaction to instances of Positive Inference, such that $f(x)$ is a 'yes' instance of Positive Inference if and only if x is a 'yes' instance of Constraint Satisfaction; that f is computable in polynomial time and

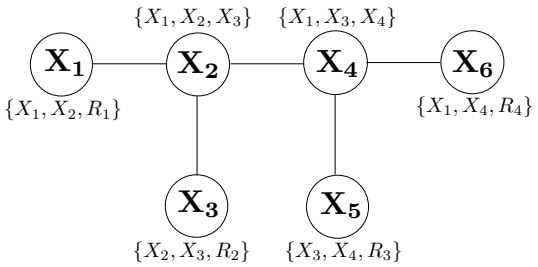


Figure 8.4 Tree-decomposition of the moralised graph of $\mathbf{G}_{\text{ex}}$

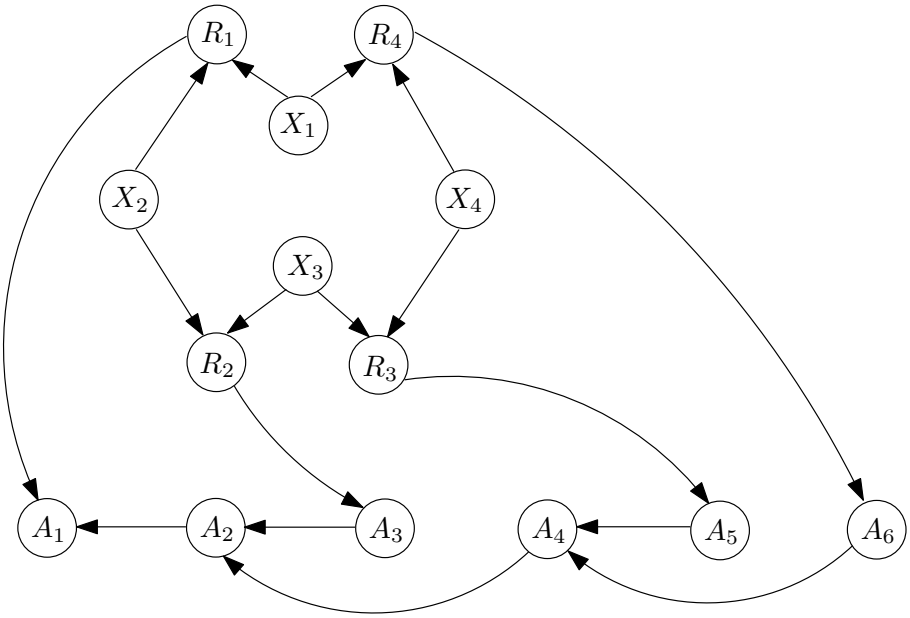


Figure 8.5 Adding nodes A_i and related arcs to $\mathbf{G}_{\text{ex}}$ to construct $\mathbf{G}'_{\text{ex}}$

that f preserves treewidth up to a linear factor, i.e., $\text{tw}(f(x)) = l(\text{tw}(x))$.

Let $\mathcal{I} = (\mathbf{V}, \mathbf{D}, \mathbf{C})$ be an instance of Constraint Satisfaction, and let $(\mathcal{B}'_{\mathcal{I}}, A_1, \text{true}, \varnothing, \varnothing)$ be the Positive Inference instance constructed from $\mathcal{I}$ as shown above; note that we don't use evidence in this instance. If $\Pr(A_1 =$

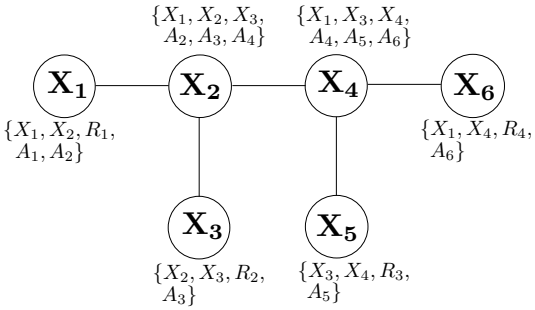


Figure 8.6 Resulting tree-decomposition of the moralisation of $\mathbf{G}'_{\text{ex}}$

TRUE) > 0 in $\mathcal{B}'_{\mathcal{I}}$, then the 'and'-construction modelled with the A_i nodes guarantees that $\Pr(\bigwedge_j R_j = \text{TRUE}) > 0$. For any node R_j, $\Pr(R_j = \text{TRUE} \,|\, \mathbf{x}) = 1$ for a particular joint value assignment $\mathbf{x}$ to the variables $\mathbf{X}$ if and only if that value assignment satisfies the relation R_j. Thus, if $\Pr(\bigwedge_j R_j = \text{TRUE}) > 0$ then there must be a joint value assignment to all variables that satisfies all relations; hence, $\mathcal{I}$ is satisfiable. On the other hand, if there is a satisfying assignment to $\mathcal{I}$ then $\Pr(\bigwedge_j R_j = \text{TRUE}) > 0$ and hence $\Pr(A_1 = \text{TRUE}) > 0$. Clearly, the above reduction can be carried out in polynomial time, since we introduce only a polynomial number of nodes for each variable in $\mathcal{I}$.

What remains to be shown is that the reduction preserves treewidth up to a linear factor. The graph $\mathbf{G}$ constructed using the X_i and R_j nodes has a treewidth of $\max(2, \text{tw}(\mathcal{I}))$ and thus has a treewidth increased by at most 1. We now show that the A_k nodes and the arcs (R_j, A_k) and (R_k, A_l) added in the construction of $\mathbf{G}'$ increase the treewidth by at most three. To facilitate our proof, we assume that the tree-decomposition $\mathbf{T_G}$ of $\mathbf{G}$ is *nice*, thus every node in $\mathbf{T_G}$ is either a LEAF NODE, INSERT NODE, FORGET NODE, or JOIN NODE. We show that our claim holds for all of these nodes.

- LEAF NODES: Suppose i is a LEAF NODE of $\mathbf{T_G}$. Then i has no children. In the construction of $\mathbf{T_G}'$, only the node A_i is added to the bag $\mathbf{X_i}$, and the treewidth is increased by at most 1.
- INSERT/FORGET NODES: Suppose i is an INSERT or FORGET NODE. Then i has one child j in $\mathbf{T_G}$. In the construction of $\mathbf{T_G}'$, the nodes A_i and A_j are added to the bag $\mathbf{X_i}$, and the treewidth is increased by at most 2.

- join nodes: Suppose i is a join node. Then i has two children j and k in $\mathbf{T_G}$. In the construction of $\mathbf{T_G}'$, A_i, A_j, and A_k are added to $\mathbf{X_i}$, and the treewidth is increased by at most 3.

We conclude that any operation on one of the nodes in the tree-decomposition can increase the size of the corresponding bag by at most three. Note that we showed before that the treewidth of the moralised graph of $\mathcal{B}_\mathcal{I}$ is equal to $\max(2, \mathrm{tw}(\mathcal{I}))$; however, in the special case that the primal graph of $\mathcal{I}$ is a path (with treewidth 1), $\mathcal{B}_\mathcal{I}$ has no join nodes and thus the treewidth is increased in either case by at most three. We can easily verify that the tree-decomposition $\mathbf{T_G}'$ indeed is a tree-decomposition of the moralised graph of $\mathcal{B}'_\mathcal{I}$, that is, for example, any node A_i and all its parents in the probabilistic network belong to the bag of node $\mathbf{X_i}$. The above reduction preserves treewidth up to a constant term, and we conclude that Constraint Satisfaction tw-reduces to Positive Inference. □

Our main result in Theorem 8.2 now follows from Theorem 8.4 and Marx' (2007) result as formulated in Theorem 8.1. For convenience, we repeat Theorem 8.2 here.

Theorem 8.2 (Inference Lower Bound.) *If there is a computable function f such that an arbitrary instance $\mathcal{P} = (\mathcal{B}, C, c, \mathbf{E}, \mathbf{e})$ with moralised graph $\mathbf{G_M}$ of the* Positive Inference *problem on probabilistic networks can be decided by an algorithm running in time $f(\mathbf{G_M}) \cdot \|\mathcal{P}\|^{o(\frac{\mathrm{tw}(\mathbf{G_M})}{\log \mathrm{tw}(\mathbf{G_M})})}$, then the* ETH *fails.*

Proof (of Theorem 8.2). Assume that there exists an algorithm A that solves arbitrary instances $\mathcal{P}$ of the Positive Inference problem with unbounded treewidth in time $f(\mathbf{G_M}) \cdot \|\mathcal{B}\|^{o(\frac{\mathrm{tw}(\mathbf{G_M})}{\log \mathrm{tw}(\mathbf{G_M})})}$, where f is a computable function and $\mathbf{G_M}$ denotes the moralised graph of $\mathcal{B}$. Let $\mathcal{I}$ be an instance of Constraint Satisfaction whose primal graph $\mathbf{G_P}$ has sufficiently large (i.e., unbounded) treewidth. Following Theorem 8.4, we can reduce $\mathcal{I}$ to $\mathcal{B}'_\mathcal{I}$ (with moralised graph $\mathbf{G_I}$) in polynomial time, where $\mathrm{tw}(\mathbf{G_I}) = \mathrm{tw}(\mathbf{G_P}) + 3$. Since we assumed that A solves $\mathcal{B}'_\mathcal{I}$ in time $f(\mathbf{G_I}) \cdot \|\mathcal{B}\|^{o(\frac{\mathrm{tw}(\mathbf{G_I})}{\log \mathrm{tw}(\mathbf{G_I})})}$, there exists a function g such that $\mathcal{I}$ can be solved in time $g(\mathbf{G_P}) \cdot \|\mathcal{I}\|^{o(\frac{\mathrm{tw}(\mathbf{G_P})}{\log \mathrm{tw}(\mathbf{G_P})})}$. By Theorem 8.1 this contradicts the ETH. □

8.3 Conclusion

In this chapter, we have proven that there cannot exist an algorithm solving arbitrary instances of Positive Inference with high treewidth in polynomial time, unless the ETH fails. In fact, we showed that any algorithm solving arbitrary instances of Positive Inference must have a running time of $f(\mathbf{G_M}) \cdot \|\mathcal{B}\|^{\omega(\frac{\mathrm{tw}(\mathbf{G_M})}{\log \mathrm{tw}(\mathbf{G_M})})}$, i.e., exponential in the treewidth of the moralised graph, up to a logarithmic factor in the exponent, even when there is no evidence. This result is weaker than Marx' (2007) result for Constraint Satisfaction and Graph Homomorphism, which shows a lower bound for algorithms solving these problems on *any* recursively enumerable class of graphs. Our result still allows algorithms to use specific properties of a network, like a particular arc direction, or planarity of the moralised graph, to arrive at sub-exponential running times, whereas Marx' (2007) result holds even for restricted classes of graphs. Nevertheless, our result is derived in a different way, namely using a treewidth-preserving reduction from Constraint Satisfaction. This technique may be of independent interest. For example, it can be used to prove (conditional) lower bounds on the running time of other problems (in probabilistic networks or otherwise) whose instances have treewidth as a structural property.

Chapter 9

Conclusion

In this thesis, we investigated the computational complexity of several problems related to the construction, analysis, and use of probabilistic networks. Already known results, and results obtained in this thesis, are summarised in Table 9.1. From this table, it will be clear why these problems are interesting also from a complexity-theoretical stance: various realistic problems in probabilistic networks are complete for classes that have few—if any—other practical complete problems.

We have explored problems that intuitively combine *selecting*, *verifying*, *enumerating*, and *probabilistic inference*. All these problems turn out to be complete for complexity classes in the Counting Hierarchy, and (as Littman et al. (1998) already noticed), it is likely that the Counting Hierarchy "...characterises many problems of interest in the area of uncertainty in artificial intelligence" (Littman

et al., 1998, p.3). Studying the exact complexity of such problems—rather than just proving them to be NP-hard—may give us some more insight in the *source* of the complexity: what makes these problems hard? This knowledge can then be used to construct efficient approximation algorithms or to construct exact algorithms that run efficiently on restricted inputs.

We have investigated under which restrictions otherwise infeasible problems can be solved in polynomial time. We have found, that efficient algorithms for Parameter Tuning are likely to exist only if both the network topology *and* the number of parameters that can be changed are restricted; and that Tunable Loose Monotonicity-D is likely to remain infeasible even under such restrictions. Furthermore, we have found that Parameter Tuning cannot be efficiently parameterised (in the number of parameters that need to be changed) unless FPT= W[1], and that Max-Local Monotonicity cannot be approximated with arbitrary precision unless APX= PTAS.

We have made some steps towards the conjecture that small treewidth is necessary for polynomial time inference algorithms. While we did not succeed in proving the conjecture for arbitrary networks, we did prove that no *general* algorithm can solve instances with high treewidth in polynomial time, unless the ETH fails. Furthermore, we introduced treewidth-preserving reductions as a means to prove conditional lower bounds. These reductions may be used to prove conditional lower bounds (under the assumption of the ETH) for other problems, both within the field of probabilistic networks, and beyond.

Of course, many interesting questions remain unanswered. Only a small number out of many interesting problems were studied, and while several results have been obtained, we were obliged to leave others for further research. For example, despite considerable efforts, we did not yet succeed in finding either an NP-hardness proof or a polynomial time algorithm for Local Monotonicity, and we were not able to pinpoint the exact complexity of Pseudo Interval Inference nor give hardness proofs for less general QPN problem variants. We leave these problems for further research.

Problem	Complexity	Study	Remarks
Positive Inference	NP-complete	Cooper (1990)	See also Chapter 2
Inference	PP-complete	Littman et al. (2001)	See also Chapter 2
Exact Inference	#P-hard	Roth (1996)	See also Chapter 2
Parameter Tuning (all problem variants)	$\mathsf{NP}^{\mathsf{PP}}$-complete	Chapter 3	remains NP-complete in polytrees and PP-complete when the number of CPTs containing parameters and the indegree of these CPTs is bounded
Parameter Tuning (parameterised variant)	W[1]-hard	Chapter 3	k = number of parameters changed
Strong Monotonicity-M	co-$\mathsf{NP}^{\mathsf{PP}}$-complete	Van der Gaag et al. (2004)	
Strong Monotonicity-D	co-$\mathsf{NP}^{\mathsf{PP}}$-complete	H.L. Bodlaender (personal communication)	
Weak Monotonicity-M	co-$\mathsf{NP}^{\mathsf{PP}}$-complete	Chapter 4	
Loose Monotonicity-D	co-$\mathsf{NP}^{\mathsf{PP}}$-complete	Chapter 4	
Tunable Loose Monotonicity-D	$\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete	Chapter 4	
MPE (decision variant)	NP-complete	Shimony (1994)	reduction from Vertex Cover
MPE (functional variant)	$\mathsf{FP}^{\mathsf{NP}}$-complete	Chapter 5	
Kth MPE (enumeration)	$\mathsf{FP}^{\mathsf{PP}}$-complete	Chapter 5	
Partial MAP (decision variant)	$\mathsf{NP}^{\mathsf{PP}}$-complete	Park and Darwiche (2004)	NP-complete in polytrees
Partial MAP (functional variant)	$\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete	Chapter 5	$\mathsf{FP}^{\mathsf{NP}}$-complete in polytrees
Kth Partial MAP (enumeration)	$\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete	Chapter 5	$\mathsf{FP}^{\mathsf{PP}}$-complete in polytrees
Pseudo Interval Inference	NP-hard	Chapter 6	not known whether in or outside NP
Max Local Monotonicity	NP-complete	Chapter 7	

Table 9.1 Summary of complexity results

Appendix

1

Completeness Proofs

In this chapter, we give completeness proofs for the CH-Distance and KL-Distance-problems from Chapter 3 and the KthNumSat and LexNumSat-problems from Chapter 5. The D_{CD} distance between two joint probability distributions $\Pr_{\mathbf{o}}$ and $\Pr_{\mathbf{o}'}$ was defined in Chapter 3 as:

$$D_{CD}(\Pr_{\mathbf{o}}, \Pr_{\mathbf{o}'}) \stackrel{def}{=} \ln \max_{\omega} \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)} - \ln \min_{\omega} \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)}$$

where ω ranges over the joint probabilities of the variables in the network. The CH-Distance-problem is defined as follows.

Appendix

CHDistance
Instance: Joint probability distributions $\Pr_{\mathbf{o}}$ and $\Pr_{\mathbf{o}'}$ and a rational number $s \in \mathbb{Q}^+$.
Question: Is $D_{CD}(\Pr_{\mathbf{o}}, \Pr_{\mathbf{o}'}) > s$?

Theorem 1.1. *Deciding* CH-Distance *is* NP*-complete.*

Proof. Given appropriate maximum and minimum joint value assignments, we can easily verify that $D_{CD} \geq s$. This proves membership in NP. To prove hardness, we will reduce CH-Distance from Most Probable Explanation, defined as follows.

Most Probable Explanation
Instance: A probabilistic network $\mathcal{B} = (\mathbf{G}, \Gamma)$, where $\mathbf{V}$ is partitioned into a set of evidence nodes $\mathbf{E}$ with a joint value assignment $\mathbf{e}$ and a set of MAP variables $\mathbf{M}$; a rational number $q \in \mathbb{Q}^+$.
Question: Is there an instantiation $\mathbf{m}$ to $\mathbf{M}$ such that $\Pr(\mathbf{m} \,|\, \mathbf{e}) \geq q$?

Most Probable Explanation has been proven NP-complete by Shimony (1994).

First observe that, if $\Pr_{\mathbf{o}'}$ is uniformly distributed, CH-Distance degenerates to determining whether $\ln \max_\omega \Pr_{\mathbf{o}}(\omega) - \ln \min_\omega \Pr_{\mathbf{o}}(\omega) \geq s$. We therefore let $\Pr_{\mathbf{o}'}$ be the uniform distribution. Let $\mathcal{B}$ be a network for which we want to solve the Most Probable Explanation-problem. We obtain $\mathcal{B}'$ by adding an additional variable P, with values p and $\bar{p}$, where $\Pr(p)$ is set to a sufficiently small value ϵ. Furthermore we add an arc from P to every variable in $\mathcal{B}$, and add a value *min* to every variable in $\mathcal{B}$. The conditional probabilities are adjusted for all variables X such that $\Pr_{\mathcal{B}'}(X = x \,|\, \bar{p}) = \Pr_{\mathcal{B}}(X = x)$, $\Pr_{\mathcal{B}'}(X = x \,|\, p) = 0$, $\Pr_{\mathcal{B}'}(X = \mathit{min} \,|\, p) = 1$, and $\Pr_{\mathcal{B}'}(X = \mathit{min} \,|\, \bar{p}) = 0$.

In $\mathcal{B}'$, the joint value assignment that has the maximum probability is exactly the most probable explanation of $\mathcal{B}$ with P set to $\bar{p}$. The assignment with the minimum probability is the assignment in which P is set to p and all other variables are set to *min*. This assignment has probability ϵ, thus, with $s \approx \ln(\frac{q}{\epsilon})$, MPE can be reduced to CHDistance in polynomial time. □

The D_{KL} measure was defined as:

$$D_{KL}(\Pr_{\mathbf{o}}, \Pr_{\mathbf{o}'}) \stackrel{def}{=} \sum_{\omega} \Pr_{\mathbf{o}}(\omega) \ln \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)}$$

If $\Pr_{\mathbf{o}} = \Pr_{\mathbf{o}'}$, then $D_{KL} = 1$ by definition. The corresponding KL-DISTANCE-problem is defined as follows.

KLDISTANCE
Instance: Let $\Pr_{\mathbf{o}}, \Pr_{\mathbf{o}'}$ denote two joint probability distributions; let $s \in \mathbb{Q}^+$.
Question: Is $D_{KL}(\Pr_{\mathbf{o}}, \Pr_{\mathbf{o}'}) > s$?

Theorem 1.2. *Deciding* KL-DISTANCE *is* PP*-complete.*

Proof. To prove membership in PP, one has to show that KLDISTANCE can be decided by a probabilistic Turing Machine. This can be done using only a slight modification of the PP-membership proof of INFERENCE shown in Chapter 2. Instead of computing $\Pr_{\mathbf{x}}(\omega)$ (by probabilistically choosing value assignments of the variables in the network according to their probabilities), now each computation path of the probabilistic Turing Machine computes $\ln \frac{\Pr_{\mathbf{x}}(\omega)}{\Pr_{\mathbf{x}'}(\omega)}$. We accept with probability $\frac{1}{2} + \epsilon$ if this computed value is at least $\frac{1}{s}$, and reject with probability $\frac{1}{2} - \epsilon$ if it isn't. Now, the probability of arriving in the accepting state is strictly larger than $\frac{1}{2}$ if and only if $D_{KL} \geq s$.

To prove PP-hardness, we reduce MAJSAT to KLDISTANCE. Let ϕ be a Boolean formula with n variables. We construct $\mathcal{B}_\phi$ using the technique described in Chapter 2, with nodes X_i for every variable X_i in ϕ and nodes for every operator in ϕ. Again, the node corresponding to the top level operator will be denoted as V_ϕ. We add an additional node P with values p and $\overline{p}$, and an additional node C, mimicking an *and*-operator, with parents P and V_ϕ. The sole parameter in which $\mathbf{o}$ and $\mathbf{o}'$ differ will be p, and we define $\Pr(p) = 1 - \epsilon$ for probability distribution $\mathbf{o}$, and $\Pr(p) = \frac{1}{\mathrm{e}}$ for probability distribution $\mathbf{o}'$, where e is Euler's number.

Now, for any joint value assignment ω that corresponds to a satisfying truth assignment to ϕ, the probability $\Pr(C = \text{TRUE})$ is equal to $1 - \epsilon$ for probability distribution $\mathbf{o}$, and $\Pr(C = \text{TRUE}) = \frac{1}{\mathrm{e}}$ for probability distribution $\mathbf{o}'$. The

computed value $\Pr_{\mathbf{o}}(\omega) \ln \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)}$ will then approximate to $1-\epsilon$. In case of a joint value assignment that corresponds to an unsatisfying truth value assignment to ϕ, $\Pr(C_\phi = \text{TRUE}) \approx \epsilon$ in both cases, and $\Pr_{\mathbf{o}}(\omega) \ln \frac{\Pr_{\mathbf{o}}(\omega)}{\Pr_{\mathbf{o}'}(\omega)} \approx 0$. It is easy to see that if $D_{KL} \geq \frac{1}{2} + \frac{1}{2^n}$, then the majority of the truth assignments to ϕ satisfies ϕ, and vice versa if the MAJSAT instance is satisfiable, then $D_{KL} \geq \frac{1}{2} + \frac{1}{2^n}$. This proves PP-hardness of KL-Distance. □

In Chapter 5 we introduced KthNumSat and LexNumSat and asserted that these problems were $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete and $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$-complete, respectively. Here we show that Kth NumSAT is $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$-complete, and thus also $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete (by Toda's theorem), and that $\mathsf{FP}^{\mathsf{NP}^{\#\mathsf{P}}}$-completeness of LexNumSat follows as a corollary.

Kth NumSAT
Instance: Boolean formula $\phi(x_1, \ldots, x_m, \ldots, x_n)$, natural numbers k, l.
Output: The lexicographically k-th assignment $x_1 \ldots\ x_m \in \{0,1\}^m$ such that exactly l assignments $x_{m+1} \ldots\ x_n \in \{0,1\}^{n-m}$ satisfy ϕ, or the empty set if such an assignment does not exist.

The $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$-completeness proof of KthNumSat is based on the $\mathsf{FP}^{\mathsf{PP}}$-completeness proof of Kth SAT by Toda (1994), and uses a result by Torán (1991) that states that the Counting Hierarchy (and thus $\mathsf{P}^{\mathsf{PP}^{\#\mathsf{P}}}$ in particular) is closed under polynomial time many-one reductions (and consequently, the functional counterpart $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$ is closed under polynomial time one-Turing reductions). Thus, any computation in $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$ can be modelled by a metric TM that calculates a bit string q based on its input x, then queries its #P oracle and writes down a number based on q and the result of the oracle, thus only querying the oracle once.

Theorem 1.3. KthNumSat *is* $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$*-complete.*

Proof. Since Toda's proof (Toda, 1994) relativises[1], a function f is in $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$ if there exists a metric TM $\widehat{\mathcal{M}}$ with access to an oracle for #P-complete problems

[1] I.e., also holds with respect to oracles

such that $f \leq_{1-T}^{\mathsf{FP}} \text{KthValue}_{\widehat{\mathcal{M}}}$. It is easy to see that a metric TM, that nondeterministically computes a satisfying assignment to $x_1 \ldots x_m$ (using an oracle for counting the number of satisfying assignments to $x_{m+1} \ldots x_n$), and writing the binary representation of this assignment on its output tape, suffices.

To prove hardness, let $\widehat{\mathcal{M}}$ be a metric TM with a $\#\mathsf{P}$ oracle. Given an input x to $\widehat{\mathcal{M}}$, we can construct (using Cook's theorem (1971)) a tuple of two Boolean formulas $(\phi_x(q), \psi_x(r))$ such that ϕ_x is true if and only if q specifies a computation path of $\widehat{\mathcal{M}}$ that is presented to the $\#\mathsf{P}$ oracle, which returns the number l of satisfying truth assignments to $\psi_x(r)$ such that $F(q, l)$ is the output of $\widehat{\mathcal{M}}$ for a particular function F. To show that this tuple exists, we need to show that multiple oracle queries and answers, during the computation of $\widehat{\mathcal{M}}$, can be translated to a single query. Assume that $\widehat{\mathcal{M}}$ writes n strings X_i (corresponding to SATISFIABILITY instances) to the oracle tape, and that the oracle each time returns the number m_i of satisfying solutions to X_i. We construct one large string X from the strings X_i (with length n_i) as follows. First we construct strings X_i', defined recursively as:

1. $X_1' = X_1$
2. $X_i' = X_i \vee \bigvee_{j=1}^{k} y_{ij}$, with $k = n_1 \cdot n_2 \cdot \ldots \cdot n_{i-1}$

using additional variables y_{ij}. Note that the number of satisfying solutions m_i' for X_i' is defined as follows:

1. $m_1' = m_1$
2. $m_i' = 2^k \cdot m_i$, with $k = n_1 \cdot n_2 \cdot \ldots \cdot n_{i-1}$

Furthermore we introduce additional variables $z_1, \ldots, z_n$ and we define X as

$$\bigvee_i ((X_i' \wedge z_i) \wedge (z_1 \vee \ldots \vee z_n) \wedge \bigwedge_{i'=1\ldots n, j=1\ldots n, i' \neq j} (\neg z_i' \vee \neg z_j))$$

Now, the number of solutions to X equals

$$m_1 + 2^{n_1} \cdot m_2 + 2^{n_1 \cdot n_2} \cdot m_3 + \cdots + 2^{n_1 \cdot \ldots \cdot n_{n-1}} \cdot m_n$$

Since the computation path that computes q is uniquely determined, q is the k-th satisfying assignment to ϕ_x for which l truth assignments to r satisfy $\psi_x(r)$, if and only if $F(q, l)$ is the k-th output of $\widehat{\mathcal{M}}$. Thus, we can construct a $\leq^{\mathsf{FP}}_{\text{1-T}}$-reduction from every function accepted by a metric TM with access to a $\#\mathsf{P}$ oracle to KthNumSat. □

Corollary 1.4. *Deciding* LexNumSat *is* $\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$*-complete.*

Proof. This can be shown using a similar argument but with $k = 1$. □

Bibliography

Abdelbar, A. M. and Hedetniemi, S. M. (1998), 'Approximating maps for belief networks is NP-hard and other theorems', *Artificial Intelligence* **102**, 21–38.

Arora, S. and Barak, B. (2009), *Complexity Theory: A Modern Approach*, Cambridge, UK: Cambridge University Press.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Spaccamela, A. M. and Protasi, M. (1998), *Complexity and Approximation*, Berlin: Springer.

Beinlich, I., Suermondt, G., Chavez, R. and Cooper, G. (1989), 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', Springer-Verlag, pp. 247–256.

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

Bodlaender, H. L. (1997), Treewidth: Algorithmic techniques and results, *in* 'Twenty-second International Symposium on Mathematical Foundations of Computer Science', Vol. LNCS 1295, Springer-Verlag, pp. 19–36.

Bodlaender, H. L. (2006), Treewidth: characterizations, applications, and computations, *in* 'Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science', pp. 1–14.

Bodlaender, H. L., Koster, A. M. C. A. and Eijkhof, F. v. d. (2005), 'Pre-processing rules for triangulation of probabilistic networks', *Computational Intelligence* **21**(3), 286–305.

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

Bolt, J. H., van der Gaag, L. C. and Renooij, S. (2005), 'Introducing situational influences in qpns', *International Journal of Approximate Reasoning* **38**, 333–354.

Castillo, E., Gutiérrez, J. M. and Hadi, A. S. (1997), 'Sensitivity analysis in discrete Bayesian networks', *IEEE Transactions on Systems, Man, and Cybernetics* **27**, 412–423.

Chan, H. and Darwiche, A. (2004), Sensitivity analysis in Bayesian networks: From single to multiple parameters, *in* 'Twentieth Conference on Uncertainty in Artificial Intelligence', AUAI Press, pp. 67–75.

Chan, H. and Darwiche, A. (2005), 'A distance measure for bounding probabilistic belief change', *International Journal of Approximate Reasoning* **38**(2), 149–174.

Chan, H. and Darwiche, A. (2006), On the robustness of most probable explanations, *in* 'Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence', pp. 63–71.

Chandrasekaran, V., Srebro, N. and Harsha, P. (2008), Complexity of inference in graphical models, *in* 'Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI'08)', AUAI Press, pp. 70–78.

Charitos, T. and van der Gaag, L. C. (2006), Sensitivity analysis for threshold decision making with DBNs, *in* 'Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence', AUAI Press, pp. 72–79.

Charniak, E. and Shimony, S. E. (1994), 'Cost-based abduction and map explanation', *Artificial Intelligence* **66**(2), 345–374.

Cofiño, A. S., Cano, R., Sordo, C. and Gutiérrez, J. M. (2002), Bayesian networks for probabilistic weather prediction, *in* F. van Harmelen, ed., 'Fifteenth European Conference on Artificial Intelligence', IOS Press, Amsterdam, pp. 695–699.

Cook, S. A. (1971), The complexity of theorem proving procedures, *in* 'Proceedings of the Third Annual ACM Symposium on Theory of Computing', pp. 151–158.

Cooper, G. F. (1984), NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge, Technical Report HPP-84-48, Stanford University.

Cooper, G. F. (1990), 'The computational complexity of probabilistic inference using bayesian belief networks', *Artificial Intelligence* **42**(2), 393–405.

Coupé, V. M. H., Jensen, F. V., Kjærulff, U. B. and van der Gaag, L. C. (2000), A computational architecture for n-way sensitivity analysis of bayesian networks, Technical report, Aalborg University.

Coupé, V. M. H. and van der Gaag, L. C. (2002), 'Properties of sensitivity analysis of Bayesian belief networks', *Annals of Mathematics and Artificial Intelligence* **36**(4), 323–356.

Coupé, V. M. H., van der Gaag, L. C. and Habbema, J. F. (2000), 'Sensitivity analysis: an aid for belief-network quantification', *Knowledge Engineering Review* **15**, 1–18.

Cozman, F. G. (2000), 'Credal networks', *Artificial Intelligence* **120**(2), 199–233.

Cozman, F. G., de Campos, C. P., Ide, J. S. and da Rocha, J. C. F. (2004), Propositional and relational Bayesian networks associated with imprecise and qualitative probabilistic assessments, *in* 'Proceedings of the 20th conference on Uncertainty in Artificial Intelligence', pp. 104–111.

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

Dagum, P. and Luby, M. (1993), 'Approximating probabilistic inference in bayesian belief networks is NP-hard', *Artificial Intelligence* **60**(1), 141–153.

de Campos, C. P. and Cozman, F. G. (2005), The inferential complexity of bayesian and credal networks, *in* 'International Joint Conference on Artificial Intelligence, Edinburgh, UK, 2005', pp. 1313–1318.

Dechter, R. (1998), Bucket elimination: a unifying framework for probabilistic inference, *in* 'Proceedings of the NATO Advanced Study Institute on Learning in Graphical Models', pp. 75–104.

Dechter, R. (1999), 'Bucket elimination: A unifying framework for reasoning', *Artificial Intelligence* **113**(1-2), 41–85.

Downey, R. and Fellows, M. (1999), *Parameterized complexity*, Berlin: Springer.

Downey, R. G. and Fellows, M. R. (1995), 'Fixed-parameter tractability and completeness ii: On completeness for w[1]', *Theoretical Computer Science* **141**(1-2), 109–131.

Druzdzel, M. J. (1999), GeNIe: A development environment for graphical decision-analytic models, *in* 'Proceedings of the AMIA Symposium', American Medical Informatics Association, p. 1206.

Druzdzel, M. J. and Henrion, M. (1993*a*), Belief propagation in qualitative probabilistic networks, *in* N. P. Carrete and M. G. Singh, eds, 'Proceedings of the Third IMACS International Workshop on Qualitative Reasoning and Decision Technologies', Barcelona: CIMNE, pp. 451–460.

Druzdzel, M. J. and Henrion, M. (1993*b*), Efficient reasoning in qualitative probabilistic networks, *in* 'Proceedings of the Eleventh Annual Conference on Artificial Intelligence', pp. 548–553.

Druzdzel, M. J. and Henrion, M. (1993*c*), Intercausal reasoning with uninstantiated ancestor nodes, *in* 'Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence', pp. 317–325.

Druzdzel, M. J. and van der Gaag, L. C. (2000), 'Building probabilistic networks: Where do the numbers come from? - guest editors introduction', *IEEE Transactions on Knowledge and Data Engineering* **12**, 481–486.

Friedman, N., Geiger, D. and Goldszmidt, M. (1997), 'Bayesian network classifiers', *Machine Learning* **29**(2-3), 131–163.

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

Gasarch, W. I. (2002), 'The P=?NP poll', *SIGACT News* **33**(2), 3447.

Geenen, P. L., Elbers, A. R. W., van der Gaag, L. C. and van der Loeffen, W. L. A. (2006), Development of a probabilistic network for clinical detection of classical swine fever, *in* 'Proceedings of the Eleventh Symposium of the International Society for Veterinary Epidemiology and Economics', pp. 667–669.

Gill, J. T. (1977), 'Computational complexity of Probabilistic Turing Machines', *SIAM Journal of Computing* **6**(4).

Grohe, M. (2007), 'The complexity of homomorphism and constraint satisfaction problems seen from the other side', *Journal of the ACM* **54**(1), 1–24.

Halpern, J. Y. (2003), *Reasoning about Uncertainty*, Cambridge: MIT Press.

Henriksen, H. J., Rasmussen, P., Brandt, G., von Bülow, D. and Jensen, F. V. (2007), 'Public participation modelling using Bayesian networks in management of groundwater contamination', *Environmental Modelling and Software* **22**(8), 1101–1113.

Impagliazzo, R. and Paturi, R. (2001), 'On the complexity of k-SAT', *Journal of Computer and System Sciences* **62**(2), 367 – 375.

Jensen, F. V., Kjærulff, U., Kristiansen, B., Langseth, H., Skaanning, C., Vomlel, J. and Vomlelova, M. (2001), 'The SACSO methodology for troubleshooting complex systems', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **15**, 321 – 333.

Jensen, F. V., Lauritzen, S. and Olesen, K. (1990), 'Bayesian updating in recursive graphical models by local computation', *Computational Statistics Quarterly* **4**, 269–282.

Jensen, F. V. and Nielsen, T. D. (2007), *Bayesian Networks and Decision Graphs*, second edn, New York: Springer Verlag.

Kappen, B., Wiegerinck, W., Akay, E., Nijman, M., Neijt, J. and van Beek, A. (2003), Promedas: A diagnostic decision support system, *in* 'Proceedings of the 15th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'03)', pp. 455–456.

Kennett, R. J., Korb, K. B. and Nicholson, A. E. (2001), Seabreeze prediction using Bayesian networks, *in* 'Proceedings of the Fifth Pacific-Asia Conference on Advances in Knowledge Discovory and Data Mining', Vol. 2035, LNCS, pp. 148–153.

Kiiveri, H., Speed, T. P. and Carlin, J. B. (1984), 'Recursive causal models', *Journal of the Australian Mathematical Society. Series A, Pure mathematics* **36**(1), 30–52.

Kjærulff, U. B. and van der Gaag, L. C. (2002), Making sensitivity analysis computationally efficient, *in* 'Proceedings of the Sixteenth Conference in Uncertainty in Artificial Intelligence', San Francisco: Morgan Kaufmann Publishers, pp. 317–325.

Kloks, T. (1994), *Treewidth*, LNCS 842, Springer-Verlag, Berlin.

Krentel, M. W. (1988), 'The complexity of optimization problems', *Journal of Computer and System Sciences* **36**, 490–509.

Kullback, S. and Leibler, R. A. (1951), 'On information and sufficiency', *The Annals of Mathematical Statistics* **22**, 79–86.

Kwisthout, J. H. P. (2007), The computational complexity of monotonicity in probabilistic networks, *in* E. Csuhaj-Varj and Z. Esik, eds, 'Proceedings of the Sixteenth International Symposium on Fundamentals of Computation Theory', Vol. 4639 of *LNCS*, Springer-Verlag, pp. 388–399.

Kwisthout, J. H. P. (2008), Complexity results for enumerating mpe and partial map, *in* M. Jaeger and T. Nielsen, eds, 'Proceedings of the Fourth European Workshop on Probabilistic Graphical Models', pp. 161–168.

Kwisthout, J. H. P., Bodlaender, H. L. and Tel, G. (2007), Local monotonicity in probabilistic networks, *in* K. Mellouli, ed., 'Proceedings of the Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty', Vol. 4724 of *LNCS*, Springer-Verlag, pp. 548–559.

Kwisthout, J. H. P. and Tel, G. (2006), Complexity results for enhanced qualitative probabilistic networks, *in* M. Studený and J. Vomlel, eds, 'Proceedings of the Third European Workshop on Probabilistic Graphical Models', Action M Agency, pp. 171–178.

Kwisthout, J. H. P. and Tel, G. (2008), 'Complexity results for enhanced qualitative probabilistic networks', *International Journal of Approximate Reasoning* **48**(3), 879–888.

Kwisthout, J. H. P. and van der Gaag, L. C. (2008), The computational complexity of sensitivity analysis and parameter tuning, *in* 'Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence', AUAI Press, pp. 349–356.

Laskey, K. B. (1995), 'Sensitivity analysis for probability assessments in Bayesian networks', *IEEE Transactions on Systems, Man, and Cybernetics* **25**, 901–909.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988), 'Local computations with probabilities on graphical structures and their application to expert systems', *Journal of the Royal Statistical Society* **50**(2), 157–224.

Lenstra, A. K., Lenstra Jr., H. W., Manasse, M. S. and Pollard, J. M. (1990), The number field sieve, *in* 'Proceedings of the 22nd Annual ACM Conference on Theory of Computing', pp. 564–572.

Littman, M. L., Goldsmith, J. and Mundhenk, M. (1998), 'The computational complexity of probabilistic planning', *Journal of Artificial Intelligence Research* **9**, 1–36.

Littman, M. L., Majercik, S. M. and Pitassi, T. (2001), 'Stochastic boolean satisfiability', *Journal of Automated Reasoning* **27**(3), 251–296.

Lucas, P. J. F., de Bruijn, N., Schurink, K. and Hoepelman, A. (2000), 'A probabilistic and decision-theoretic approach to the management of infectious disease at the ICU', *Artificial Intelligence in Medicine* **3**(19), 251–279.

Majercik, S. M. and Littman, M. L. (1998), MAXPLAN: A new approach to probabilistic planning, *in* 'Proceedings of the Fourth International Conference on Artificial Intelligence Planning', Vol. 8693.

Marx, D. (2007), Can you beat treewidth?, *in* 'Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)', pp. 169–179.

Papadimitriou, C. H. (1984), 'On the complexity of unique solutions', *Journal of the Association for Computing Machinery* **31**(2), 392–400.

Papadimitriou, C. H. (1994), *Computational Complexity*, Addison-Wesley.

Park, J. D. and Darwiche, A. (2004), 'Complexity results and approximation settings for MAP explanations', *Journal of Artificial Intelligence Research* **21**, 101–133.

Pearl, J. (1986), 'Fusion, propagation, and structuring in belief networks', *Artificial Intelligence* **29**, 241–288.

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

Renooij, S. (2001), Qualitative Approaches to Quantifying Probabilistic Networks, PhD thesis, Institute for Information and Computing Sciences, Utrecht University, The Netherlands.

Renooij, S., Parsons, S. and Pardieck, P. (2003), Using kappas as indicators of strength in qualitative probabilistic networks, *in* T. N. Zhang and N.L., eds, 'Seventh European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty', Vol. 2711 of *LNAI*, Springer-Verlag, pp. 87–99.

Renooij, S. and van der Gaag, L. C. (1999), Enhancing QPNs for trade-off resolution, *in* 'Proceedings of the Fifteenth Conference on Uncertainy in Artificial Intelligence', pp. 559–566.

Renooij, S. and van der Gaag, L. C. (2002), From qualitative to quantitative probabilistic networks, *in* 'Proceedings of the Eighteenth Conference in Uncertainty in Artificial Intelligence', pp. 422–429.

Renooij, S. and van der Gaag, L. C. (2008), 'Enhanced qualitative probabilistic networks for resolving trade-offs', *Artificial Intelligence* **172**(12-13), 1470–1494.

Renooij, S., van der Gaag, L. C. and Parsons, S. (2002), 'Context-specific sign-propagation in qualitative probabilistic networks', *Artificial Intelligence* **140**(1/2), 207–230.

Robertson, N. and Seymour, P. (1986), 'Graph minors II: Algorithmic aspects of tree-width', *Journal of Algorithms* **7**, 309–322.

Roth, D. (1996), 'On the hardness of approximate reasoning', *Artificial Intelligence* **82**(1-2), 273–302.

Santos, E. (1991), On the generation of alternative explanations with implications for belief revision, *in* 'Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence', pp. 339–347.

Schrage, M. M., van IJzendoorn, A. and van der Gaag, L. C. (2005), Haskell ready to dazzle the real world, *in* 'Proceedings of the 2005 ACM SIGPLAN workshop on Haskell', ACM New York, NY, USA, pp. 17–26.

Shachter, R. D. (1986), 'Evaluating influence diagrams', *Operations Research* **34**(6), 871–882.

Shenoy, P. P. and Shafer, G. (1986), 'Propagating belief functions with local computations', *IEEE Expert* **1**(3), 43–52.

Shimony, S. E. (1994), 'Finding MAPs for belief networks is NP-hard', *Artificial Intelligence* **68**(2), 399–410.

Simon, J. (1977), On the difference between one and many, *in* 'Proceedings of the Fourth Colloquium on Automata, Languages, and Programming', Vol. 52 of *LNCS*, Springer-Verlag, pp. 480–491.

Sipser, M. (1992), The history and status of the P versus NP question, *in* 'Twenty-fourth Annual ACM Symposium on the Theory of Computing', pp. 603–619.

Srinivas, S. and Nayak, P. (1996), Efficient enumeration of instantiations in Bayesian networks, *in* 'Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence', pp. 500–508.

Stockmeyer, L. (1977), 'The polynomial-time hierarchy', *Theoretical Computer Science* **3**, 1–22.

Sy, B. K. (1992), Reasoning MPE to multiply connected belief networks using message-passing, *in* 'Proceedings of the Tenth National Conference on Artificial Intelligence', pp. 570–576.

Toda, S. (1991), 'PP is as hard as the polynomial-time hierarchy', *SIAM Journal of Computing* **20**(5), 865–877.

Toda, S. (1994), 'Simple characterizations of P(#P) and complete problems', *Journal of Computer and System Sciences* **49**, 1–17.

Torán, J. (1991), 'Complexity classes defined by counting quantifiers', *Journal of the ACM* **38**(3), 752–773.

van der Gaag, L. C., Bodlaender, H. L. and Feelders, A. (2004), Monotonicity in Bayesian networks, *in* 'Twentieth Conference on Uncertainty in Artificial Intelligence', AUAI Press, pp. 569–576.

van der Gaag, L. C., Geenen, P. L. and Tabachneck-Schijf, H. J. M. (2006), Verifying monotonicity in bayesian networks with domain experts, *in* L. C. van der Gaag and R. Almond, eds, 'Proceedings of the 4th Bayesian Modelling Applications Workshop: Bayesian Models Meet Cognition'.

van der Gaag, L. C. and Renooij, S. (2001), Analysing sensitivity data, *in* 'Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence', Morgan Kaufmann Publishers, San Francisco, pp. 530–527.

van der Gaag, L. C., Renooij, S. and Coupé, V. M. H. (2007), Sensitivity analysis of probabilistic networks, *in* P. Lucas, J. Gámez and A. Salmeron, eds, 'Advances in Probabilistic Graphical Models, Studies in Fuzziness and Soft Computing', Vol. 213, Berlin: Springer Verlag, pp. 103–124.

van der Gaag, L. C., Renooij, S., Witteman, C. L. M., Aleman, B. M. P. and Taal, B. G. (2002), 'Probabilities for a probabilistic network: a case study in oesophageal cancer', *Artificial Intelligence in Medicine* **25**, 123–148.

van Kouwen, F., Renooij, S. and Schot, P. (2009), 'Inference in qualitative probabilistic networks revisited', *International Journal of Approximate Reasoning* **50**(5), 708–720.

Wagner, K. W. (1986), 'The complexity of combinatorial problems with succinct input representation', *Acta Informatica* **23**, 325–356.

Wasyluk, H., Onisko, A. and Druzdzel, M. J. (2001), 'Support of diagnosis of liver disorders based on a causal Bayesian network model', *Medical Science Monitor* **7**, 327–332.

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

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

Zhang, C., Sun, S. and Yu, G. (2004), A Bayesian networks approach to time series forecasting of short-term traffic flows, *in* 'Seventh International IEEE Conference on Intelligent Transportation Systems', pp. 216–221.

Samenvatting

Probabilistische of Bayesiaanse netwerken worden veel gebruikt om te redeneren met onzekerheid. Een probabilistisch netwerk bestaat uit een gerichte acyclische graaf en een verzameling conditionele kansen. De knopen van de graaf representeren stochastische variabelen, de pijlen representeren afhankelijkheden tussen deze variabelen, en de conditionele kansen geven, voor iedere variabele, een kansverdeling over zijn waarden, gegeven een bepaalde combinatie van waarden van zijn ouders in de graaf. Het geheel modelleert onzekerheidsrelaties binnen een bepaald domein; met behulp van het netwerk en de conditionele kansen kunnen we tal van relevante kansen in het netwerk berekenen. In een medisch domein kunnen we bijvoorbeeld berekenen wat de kans op een bepaalde infectie is, gegeven observaties van ziektebeelden en testuitslagen. Aan de andere kant kunnen we ook berekenen wat de kans op bepaalde ziektebeelden is, gegeven dat de patient een bepaalde infectie heeft. We kunnen uitrekenen wat de meest waarschijnlijke infectie is met of zonder gegevens over ziektebeelden, enzovoorts.

Algoritmen voor dergelijke vraagstukken nemen doorgaans veel tijd in beslag: de looptijd is in het slechtste geval exponentieel in de omvang van het netwerk. Het

is ook niet aannemelijk dat er algoritmen gevonden kunnen worden die significant sneller zijn; echter, veel algoritmen kunnen goed uit de voeten met kleinere netwerken waarvan de boombreedte (*informeel*: de mate waarin de graaf op een boom lijkt) beperkt is. Onderzoek naar de moeilijkheid van computationele problemen vindt doorgaans vaak plaats met behulp van computationele complexiteitstheorie. Bekende complexiteitsklassen zijn P en NP: een probleem Π zit in P als er een algoritme bestaat dat arbitraire instanties van Π oplost in een tijd die hooguit polynomiaal is in de omvang van deze instantie. Een probleem zit in NP als er een algoritme bestaat dat, gegeven een oplossing voor een instantie van Π, in polynomiale tijd kan controleren of deze oplossing voldoet. Alle problemen in P zitten ook in NP (als je een oplossing snel kunt berekenen, kun je een gegeven oplossing natuurlijk ook snel op juistheid controleren), maar het omgekeerde is zeer waarschijnlijk niet het geval: de vraag "gegeven alle gemeentehuizen van Nederland, is er een route die begint op het gemeentehuis van Breda, alle andere gemeentehuizen in Nederland precies één keer bezoekt, en weer in Breda eindigt en maximaal x kilometers telt" is een instantie van het zogenaamde Handelsreizigers-probleem waarvan wordt aangenomen dat er geen polynomiaal algoritme gevonden kan worden dat het probleem in polynomiale tijd oplost; het Handelsreizigers-probleem is namelijk NP-moeilijk, wat inhoudt dat een polynomiaal algoritme voor dit probleem bewijst dat $\mathsf{P} = \mathsf{NP}$. Een aantal van deze moeilijke problemen zijn echter snel benaderbaar, als de oplossing niet exact hoeft te zijn, of kunnen snel opgelost worden als een bepaalde parameter van de instantie - zoals de boombreedte - klein is. Andere problemen zijn zelfs 'moeilijker dan NP', althans binnen gangbare aannames in de complexiteitstheorie. Inferentie in probabilistische netwerken is bijvoorbeeld PP-volledig.

In dit proefschrift bespreken we de computationele complexiteit van een aantal voor probabilistische netwerken relevante vraagstukken. In de inleiding van het proefschrift, bestaande uit de hoofdstukken 1 en 2, bespreken we een aantal definities en eigenschappen van probabilistische netwerken en computationele complexiteitstheorie. In het eerste inhoudelijke deel, bestaande uit hoofdstukken 3, 4, en 5, bekijken we vraagstukken die te maken hebben met de *analyse* van netwerken. In hoofdstuk 3 bekijken we de vraagstukken Sensitivity Analysis en Parameter Tuning. In beide vraagstukken willen we weten wat het effect is van het aanpassen van een verzameling van conditionele kansen op de conditionele kansverdeling van een bepaalde verzameling variabelen. We bewijzen dat dit probleem $\mathsf{NP}^{\mathsf{PP}}$-volledig is in zijn algemeenheid, dat het NP-volledig blijft als het netwerk de sterk beperkte structuur van een polytree heeft, dat het prob-

leem PP-volledig blijft als het aantal kansen wat we aanpassen beperkt is. Dit maakt het aannemelijk dat dit probleem alleen binnen polynomiale tijd opgelost kan worden als aan beide restricties wordt voldaan: een beperkte boombreedte en een beperkt aantal aan te passen kansen.

In hoofdstuk 4 bekijken we varianten van het Monotonicity-probleem: gegeven twee verzamelingen variabelen **E** en **C** in het netwerk, is het zo dat een hogere waardetoekenning aan **E** de kans op een hogere waarde voor **C** nooit lager (of: nooit hoger) kan maken; oftewel, is **C** monotoon in **E**? Van dit probleem is bekend dat het co-$\mathsf{NP}^{\mathsf{PP}}$-volledig is; in dit hoofdstuk bekijken we varianten van dit probleem en bewijzen we dat al deze varianten co-$\mathsf{NP}^{\mathsf{PP}}$-volledig zijn, met andere woorden: ook als inferentie in polynomiale tijd kan, blijft het onwaarschijnlijk dat een snelle oplossing voor dit probleem gevonden kan worden. We combineren de problemen in hoofdstuk 3 en 4 tot het Tunable Monotonicity-probleem: gegeven een netwerk, kun je een verzameling kansen zo aanpassen dat **C** monotoon wordt in **E**? Hiervan bewijzen we dat dit probleem $\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$-volledig is, en dat een polynomiaal algoritme onwaarschijnlijk is, zelfs als zowel het aantal kansen als de boombreedte van de graaf beperkt is.

In hoofdstuk 5 komt het vinden van de meest waarschijnlijke waarden van variabelen aan de orde. Uit de literatuur is bekend, dat het bepalen of er een waardetoekenning is aan een verzameling variabelen **M**, gegeven een observatie voor de overige variabelen **E**, met waarde groter dan q, NP-volledig is. Als **M** en **E** samen niet de volledige verzameling variabelen vormen, is het probleem zelfs $\mathsf{NP}^{\mathsf{PP}}$-volledig. In dit hoofdstuk breiden we dit probleem uit naar het vinden van de k-de meest waarschijnlijke waardetoekenning, en we bewijzen dat het vinden van een dergelijke toekenning $\mathsf{FP}^{\mathsf{PP}}$-volledig is en zelfs $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-volledig als **M** en **E** samen niet de volledige verzameling variabelen vormen.

In het tweede inhoudelijke deel van het proefschrift, bestaande uit de hoofdstukken 6 en 7, bespreken we *abstractie* van probabilistische netwerken. In hoofdstuk 6 bespreken we uitgebreide kwalitatieve probabilistische netwerken. In een kwalititatief probabilistisch netwerk zijn de conditionele kansen samengevat in tekens: '+' als hogere waarden voor de ene variabele niet-lagere waarden voor de andere variabele voorspellen, voor iedere combinatie van de ouders, '-' als het effect negatief is, en '?' als de relatie non-monotoon is. Inferentie in dit soort netwerken kan in polynomiale tijd; echter, in de abstractie gaat veel informatie verloren. Van een aantal uitbreidingen (met sterke en zwakke invloeden)

is onderzocht wat de complexiteit is; indien de abstractie een interval is (het effect van variabele A op variabele B bevindt zich in het interval $[a, b]$) is het inferentieprobleem NP-moeilijk, indien we toestaan dat een probleeminstantie (een graaf en een verzameling intervallen) geen onderliggend probabilistisch netwerk kan hebben.

In hoofdstuk 7 bekijken we lokale monotonie. Vaak hebben variabelen een logische ordening, zoals veel > weinig > niets, maar soms is deze ordening abstract en willekeurig, zoals de ordening van de waarden van een variabele 'bloedgroep'. Een 'verkeerde' ordening kan er echter wel voor zorgen dat er non-monotone invloeden in een kwalitatieve abstractie van het netwerken ontstaan die voorkomen kunnen worden bij een andere ordening. We onderzoeken de vraag hoe je een optimale ordening kunt vinden. We bewijzen dat dit probleem NP-volledig is, en dat het moeilijk te benaderen is. We geven een algoritme dat in het slechtste geval exponentiële tijd nodig heeft om het probleem op te lossen.

Het laatste inhoudelijke deel, hoofdstuk 8, bespreekt het inferentieprobleem in probabilistische netwerken. Met behulp van een resultaat uit de grafentheorie laten we zien dat er geen snellere algoritmen kunnen bestaan voor dit probleem, dan de nu bestaande algoritmen die exponentieel zijn in de boombreedte van het netwerk, gegeven een aantal vooronderstellingen uit de complexiteitstheorie. We introduceren een nieuwe reductiemethode - die ook voor andere problemen gebruikt kan worden - om te laten zien dat als je een algoritme hebt dat een willekeurig netwerk met een hoge boombreedte in polynomiale tijd oplost, dat dan een bekende hypothese uit de complexiteitstheorie onhoudbaar is. In hoofdstuk 9, tenslotte, concluderen we dit proefschrift met een samenvatting en inkadering van deze resultaten.

List of Complexity Classes

APX	Class of optimisation problems that can be approximated in polynomial time with some constant ratio r, *see page 119*
co-NP	Class of problems for which *non*-membership is verifiable in polynomial time by a deterministic Turing Machine using a certificate, *see page 10*
co-$\mathsf{NP}^{\mathsf{PP}}$	Class of problems for which non-membership can be verified in polynomial time on a deterministic Turing Machine with access to an oracle for problems in PP, *see page 11*
FNP	Class of functions that can be computed on a deterministic Turing Machine in polynomial time, *see page 64*
$\mathsf{FP}^{\mathsf{NP}}$	Class of functions that can be calculated in polynomial time on a deterministic Turing Machine with access to an oracle for problems in NP, *see page 64*
$\mathsf{FP}^{\mathsf{NP}^{\mathsf{PP}}}$	Class of functions that can be calculated in polynomial time on a

deterministic Turing Machine with access to an oracle for problems in $\mathsf{NP}^{\mathsf{PP}}$, *see page 71*

$\mathsf{FP}^{\mathsf{PP}}$ Class of functions that can be calculated in polynomial time on a deterministic Turing Machine with access to an oracle for problems in PP, *see page 64*

$\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$ Class of functions that can be computed in polynomial time on a deterministic Turing Machine with access to an oracle for problems in $\mathsf{PP}^{\mathsf{PP}}$, *see page 71*

FPT Class of problems that have a parameter k such that membership can be decided in $f(k) \cdot n^c)$ for any function f and constant c, *see page 42*

NP Class of problems decidable in polynomial time by a non-deterministic Turing Machine; alternatively, class of problems for which membership is verifiable in polynomial time by a deterministic Turing Machine using a certificate, *see page 9*

$\mathsf{NP}^{\mathsf{NP}^{\mathsf{PP}}}$ Class of problems decidable in polynomial time on a non-deterministic Turing Machine with access to an oracle for problems in $\mathsf{NP}^{\mathsf{PP}}$, *see page 57*

$\mathsf{NP}^{\mathsf{PP}}$ Class of problems decidable on a non-deterministic Turing Machine with access to an oracle for problems in PP, *see page 11*

$\#\mathsf{P}$ Class of counting problems whose decision variant is in NP, *see page 10*

P Class of problems decidable in polynomial time by a deterministic Turing Machine, *see page 9*

P^{NP} Class of problems decidable on a deterministic Turing Machine with access to an oracle for problems in NP, *see page 64*

PP Class of problems decidable in polynomial time on a Probabilistic Turing Machine, *see page 10*

$\mathsf{P/poly}$ Class of problems decidable in polynomial time by a deterministic Turing machine with a polynomial-bounded advice function, *see page 128*

P^{PP} Class of problems decidable in polynomial time on a deterministic Turing Machine with access to an oracle for problems in PP, *see page 64*

PTAS Class of optimisation problems that can be approximated with every ratio, *see page 119*

W[1] Class of problems equivalent to the following problem: 'Given a non-deterministic Turing Machine $\mathcal{M}$ and an input x: does $\mathcal{M}$ accept x after k steps?', *see page 42*

List of Problems

3Sat, 89
A-Majsat, 12, 51
Bounded Kth Partial MAP, 71
CH-Distance, 144
Clique, 42
cond-MAP, 54
Exact Inference, 19
E-Majsat, 12
Evidence Mode Tuning, 32
Evidence Parameter Tuning Range, 31
Graph 3-Colourability, 116
Inference, 19
KL-Distance, 145
Kth MPE, 66
KthNumSat, 71
KthSat, 67

Loose Monotonicity, 50
Local Monotonicity, 108
Max-3-Colourability, 119
Majsat, 10, 12
Maxsat, 37
Minimal Change Mode Tuning, 32
Minimal Change Parameter Tuning Range, 32
Max-Local Monotonicity, 115
Most Probable Explanation, 61
Minimal Parameter Mode Tuning, 32
Minimal Parameter Tuning Range, 32
Mode Tuning, 32
#Sat, 10
Positive 3Sat, 43
Pseudo Interval Inference, 89
    with relaxation scheme, 97
Positive Inference, 18
Partial MAP, 20, 62
Parameterized Parameter Tuning, 43
Parameter Tuning, 31
Parameter Tuning Range, 31
Satisfiability, 8
Strong Monotonicity, 50
Tautology, 10
Tunable Monotonicity, 50
Vertex Cover, 42
Weak Monotonicity, 50

Index

$\Omega(X)$, 14
$\pi(Y)$, 14
$\sigma(X)$, 14

ancestors, 14
approximation ratio, 119
arc constraints, 110

Bayes' rule, 15
Bodlaender theorem, 13
branch-and-bound, 121

certificate, 9
chain rule, 15
clique tree propagation, 18
collider, 16
combination of values, 27
completeness, 9
conditioning, 15
constraint satisfaction problem, 110
Counting Hierarchy, 11
cumulative distribution function, 15

d-connection, 16
d-separation, 16
descendants, 14
distance measures, 29
    D_{CD}, 29
    D_E, 30
    D_{KL}, 30

encoding, 9
enhanced QPNs, 86
Exponential Time Hypothesis, 130

fork, 16

hardness, 9

influence sign, 83
    non-monotonicity, 83
    properties, 84, 88
interval-based network, 88

joint probability distribution, 14

local monotonicity, 103, 108
    maximal, 108
loop cutset, 17
loop cutset conditioning, 18

marginalisation, 15
monotonicity
    loose, 49
    strong, 48
    weak, 48
monotonicity function, 106

partial, 106
monotonicity matrix, 107
partial, 107

ordering, 103, 104

parameterisation, 42
parameters, 27
polytree, 37
probabilistic network, 15
complexity, 16
definition, 14
probability, 15

QPNs, 81
definition, 82
inference algorithm, 85
use of, 82

reductions
L-, 120
many-one, 9
one-Turing, 65
treewidth-preserving, 131
relaxation scheme, 96

sensitivity analysis, 25
n-way, 28
one-way, 27
sensitivity function, 27
stochastic dominance, 47

time complexity, 8
tree-decomposition, 12
nice, 13
treewidth, 13
tuning, 26, 29
Turing Machine, 8
deterministic, 8
metric, 65
non-deterministic, 8
Oracle, 11
Probabilistic, 10
Turing Transducer, 8
metric, 65
variable elimination, 18

Curriculum Vitae

Johan Kwisthout was born on May 19th 1976 in Breda, the Netherlands. In 1994 he received his VWO-diploma from the Graaf Engelbrecht College in Breda. From 1997 to 2005 he studied computer science ('Technische Informatica') at the Open University. His master's thesis from the Open University was entitled "Modelling Uncertainty in 3APL" and was supervised by prof. dr. J.T. Jeuring, dr. M.M. Dastani, and dr. ir. S.E. de Vos. From 2002 to 2006 he studied artificial intelligence ('Kunstmatige Intelligentie') at the Radboud University Nijmegen. His master's thesis there was entitled "Joint Attention, Language Evolution, and Development" and was supervised by prof. dr. A.F.J. Dijkstra and dr. W.F.H. Haselager. In 2005 he started as a Ph.D. student in the Department of Information and Computing Sciences at Utrecht University in the group of prof. dr. J. van Leeuwen.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue.* Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms.* Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications.* Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures.* Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability.* Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions.* Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments.* Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages.* Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations.* Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication*. Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations.* Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data.* Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space.* Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization.* Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML.* Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models.* Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications.* Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols.* Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition.* Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Dis-*

tributed Systems. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems*. Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML*. Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery*. Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding*. Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks*. Faculty

of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data*. Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy*. Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS*. Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems*. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates*. Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines*. Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras*. Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit*. Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks*. Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes*

and Commitments. Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design*. Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems*. Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms*. Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation*. Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems*. Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification*. Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development*. Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation*. Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Al-*

gorithms for Service Discovery and Provisioning. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean*. Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques*. Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time*. Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation*. Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML*. Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web*. Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory*. Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems*. Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis*. Faculty of Mathematics and Natural Sciences, UL. 2009-22

J.H.P. Kwisthout. *The Computational Complexity of Probabilistic Networks*. Faculty of Science, UU. 2009-23

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

Colofon

This thesis was typeset with LaTeX 2_ε.

The design of the cover was done by Bas Maes.

ISBN: 978-90-939-5196-3