Query-Specific Learning and Inference for Probabilistic Graphical Models

Anton Chechetka

CMU-RI-TR-11-18

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

August 2011

Thesis Committee:
Carlos Guestrin, Chair
J. Andrew Bagnell
Eric Xing
Pedro Domingos, University of Washington

Abstract

In numerous real world applications, from sensor networks to computer vision to natural text processing, one needs to reason about the system in question in the face of uncertainty. A key problem in all those settings is to compute the probability distribution over the variables of interest (the *query*) given the observed values of other random variables (the *evidence*). Probabilistic graphical models (PGMs) have become the approach of choice for representing and reasoning with high-dimensional probability distributions. However, for most models capable of accurately representing real-life distributions, inference is fundamentally intractable. As a result, optimally balancing the expressive power and inference complexity of the models, as well as designing better approximate inference algorithms, remain important open problems with potential to significantly improve the quality of answers to probabilistic queries.

This thesis contributes algorithms for learning and approximate inference in probabilistic graphical models that improve on the state of the art by emphasizing the computational aspects of inference over the representational properties of the models. Our contributions fall into two categories: learning accurate models where exact inference is tractable and speeding up approximate inference by focusing computation on the query variables and only spending as much effort on the remaining parts of the model as needed to answer the query accurately.

First, for a case when the set of evidence variables is not known in advance and a single model is needed that can be used to answer any query well, we propose a polynomial time algorithm for learning the structure of tractable graphical models with quality guarantees, including PAC learnability and graceful degradation guarantees. Ours is the first efficient algorithm to provide this type of guarantees. A key theoretical insight of our approach is a tractable upper bound on the mutual information of arbitrarily large sets of random variables that yields exponential speedups over the exact computation.

Second, for a setting where the set of evidence variables is known in advance, we propose an approach for learning tractable models that tailors the structure of the model for the particular value of evidence that become known at test time. By avoiding a commitment to a single tractable structure during learning, we are able to expand the representation power of the model without sacrificing efficient exact inference and parameter learning. We provide a general framework that allows one to leverage existing structure learning algorithms for discovering high-quality evidence-specific structures. Empirically, we demonstrate state of the art accuracy on real-life datasets and an order of magnitude speedup.

Finally, for applications where the intractable model structure is a given and approximate inference is needed, we propose a principled way to speed up convergence of belief propagation by focusing the computation on the query variables and away from the variables that are of no direct interest to the user. We demonstrate significant speedups over the state of the art on large-scale relational models. Unlike existing approaches, ours does not involve model simplification, and thus has an advantage of converging to the fixed point of the full model.

More generally, we argue that the common approach of concentrating on the structure of representation provided by PGMs, and only structuring the computation as representation allows, is suboptimal because of the fundamental computational problems. It is the computation that eventually yields answers to the queries, so directly focusing on structure of computation is a natural direction for improving the quality of the answers. The results of this thesis are a step towards adapting the structure of computation as a foundation of graphical models.

Acknowledgments

First and foremost, I am deeply grateful to my advisor, Carlos Guestrin, for all his support, guidance, encouragement, challenging questions and patience. Carlos is really the best advisor I could ask for, always a source of deep insights and inspiration. From minute details of algorithms, to handling research setbacks and even simply looking on the positive side of things, I have learned a lot from him. Thank you, Carlos!

Thank you to the rest of my thesis committee: Drew Bagnell, Eric Xing, and Pedro Domingos, for their valuable feedback that made this thesis better. I am also thankful to Geoff Gordon for the many insightful comments on this work at the SELECT lab meetings. Thank you to Katia Sycara for guiding me through the first two and a half years at CMU.

Summer internships were a great experience, and I would like to thank Moises Goldszmidt, Michael Isard, Denver Dash, David Petrou and Gabriel Taubman for the opportunities to tackle exciting new problems and for passing on their knowledge.

I am thankful to the SELECT labmates, Danny Bickson, Byron Boots, Joseph Bradley, Kit Chen, Miroslav Dudik, Khalid El-Arini, Stanislav Funiak, Joseph Gonzalez, Arthur Gretton, Sue Ann Hong, Jonathan Huang, Andreas Krause, Aapo Kyrola, Yucheng Low, Dafna Shahaf, Sajid Siddiqi, Ajit Singh, Le Song, Gaurav Veda, Yisong Yue, and Brian Ziebart, for being both great labmates and good friends, for great memories and for making life at CMU much more fun. Thank you to Mary Koes and Young-Woo Seo for helping me along for the first two years of grad school. Thank you to Kostya Salomatin for being a great roommate.

Michelle Martin and Suzanne Lyons Muth have always been extremely helpful and made sure that at school, research is the only thing that required effort. I am especially thankful to Suzanne for her patience with my seemingly constant pushing of deadlines.

Finally, I will always be grateful to my family for their love and encouragement, and for always supporting me along the way no matter which direction in life I took. Thank you!

Contents

Chapter 1

Introduction

Reasoning under uncertainty over high-dimensional structured spaces is a fundamental problem of machine learning with a multitude of applications. To optimally control air conditioning in a building, it is necessary to infer the true distribution of the temperature throughout the building by only observing noisy measurements of several sensors. For intelligent image retrieval, it is necessary to infer the identities and locations of the objects in the images given the raw pixel brightness values, possibly corrupted by compression. For accurate web search, one needs to infer the topics that any given webpage covers given the text and the connectivity pattern of hyperlinks.

In all of the applications mentioned above, the state of the system can be modeled as a set of correlated random variables X with joint distribution $P(X)$. Moreover, a subset of variables E, which we will call **evidence** can be typically observed at test time. The fundamental problem then is to compute the conditional distribution of variables of interest Q, which we will call the **query,** given evidence:

$$P(Q \mid E) = \frac{P(Q, E)}{P(E)}. \tag{1.1}$$

A major difficulty in computing the conditional distribution (1.1) arises from the fact that both query and evidence may contain a very large number of variables. For example, a model of the Internet may have a separate random variable for every webpage, and a model in computer vision may have a separate random variable for every pixel of an image. The resulting random distributions have extremely high dimensionality, making enumeration of the state space, and consequently the straightforward representation and inference approaches, intractable.

Fortunately, *probabilistic graphical models* (PGMs, see Koller and Friedman (2009) for in-depth treatment) have emerged as a powerful formalism for compactly representing high-dimensional distributions and have been used successfully in a multitude of applications from computer vision (Li, 1995) to natural language understanding (Blei et al., 2003) to modeling protein structure (Yanover et al., 2007) and many others. Graphical models owe their success to two key features:

1. **Intuitive interpretation.** In graphical models, direct probabilistic dependencies between random variables are encoded via edges of a graph, whereby the absence of an edge between two variables means that those variables are conditionally independent given the rest of X. As a result, whenever the problem domain can be described using local interactions (for example, for a model of the Internet, by assuming only webpages that link to each other have directly dependent topics), such a description is straightforward to map to a graphical model structure.

2. **Efficient inference algorithms.** Although even the problem of approximate inference is provably hard for the vast majority of compact PGMs (Cooper, 1990; Dagum and Luby, 1993), a variety of approximate inference algorithms have been developed that often work well in practice (Pearl, 1988; Geman and Geman, 1984; Jordan et al., 1999). As a result, it is not necessary for practitioners to become experts in probabilistic inference. Instead, usually once can simply pass the model describing the problem domain to an off the shelf inference algorithm and obtain useful results.

While the approach of first, defining a PGM that captures all of the essential interactions of the problem domain in an easily interpretable way, and second, running a standard inference procedure to answer probabilistic queries is undeniably convenient for the practitioners, often such an approach leads to suboptimal performance. In this thesis we identify two issues that often manifest themselves in practice. First, too complex models are often chosen, which leads to excessive errors in inference results and high computational cost of approximate inference. Second, standard inference algorithms lack a notion of the importance of an unobserved variable to the end user, and have excessive computational cost in the presence of a large number of unobserved nuisance variables in the system. More generally, we claim the following thesis statement:

> **By relaxing the requirement that a probabilistic graphical model structure closely reflects the interactions in the underlying application domain, and better exploiting the information about query and evidence variables, it is possible to answer probabilistic queries more accurately and efficiently than the state of the art approaches.**

The main contributions of this thesis, discussed in more detail in section 1.2, are two novel approaches for learning accurate tractable PGMs, and a novel inference approach that, by focusing computation on the query variables at the expense of nuisance variables, significantly improves efficiency, especially **test time performance,** without sacrificing the result accuracy. Before describing the contributions in more detail, we review the issues with the typical way of applying PGMs that our approaches aim to resolve.

1.0.1 Tradeoffs in model design

The root of the model design problem lies in the need to trade off the *accuracy* that can be achieved by using the model and the *computational resources* necessary to process the model. Somewhat counterintuitively, the nature of the basic tradeoff is **not** simply "richer models are more accurate, but have higher computational complexity". Let us briefly review the impact of model complexity on both accuracy and computational efficiency.

The **eventual accuracy** experienced by the end user of a graphical model is affected by two factors: errors due to the simplified representation of the true distribution $P(X)$ using a graphical model *(representation error)* and the errors due to the approximate nature of the inference algorithms *(inference error)*. In general, simpler models with fewer direct dependencies yield larger representation errors, but more accurate inference results. Complex models better reflect the properties of the problem domain and have smaller representation error, but pose more difficult inference problems. The resulting tradeoff is show schematically in Fig. 1.1.

Reaching the optimal tradeoff point in Fig. 1.1 is difficult in practice, because the two error sources possess very different properties:

1. **Representation error** is straightforward to understand by thinking about causal dependencies in

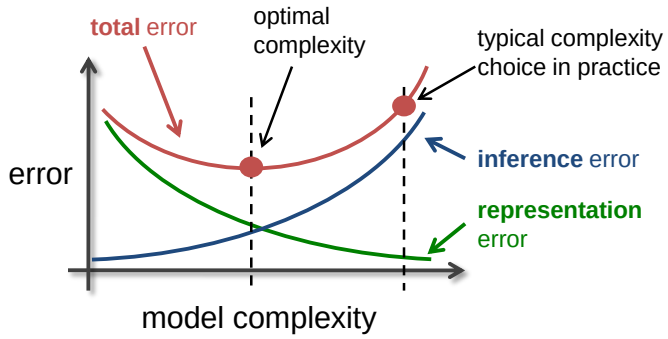


Figure 1.1: A qualitative illustration of a tradeoff between complexity of a probabilistic graphical model and inference accuracy. Simpler models have low inference error, but do not approximate the true system distribution very well. On the other hand, complex model can approximate a distribution of interest very accurately, but high inference error does not allow one to realize this accuracy in practice. Often, overly complex models are chosen in practice, because approximation error is easier to both understand and affect for the practitioners.

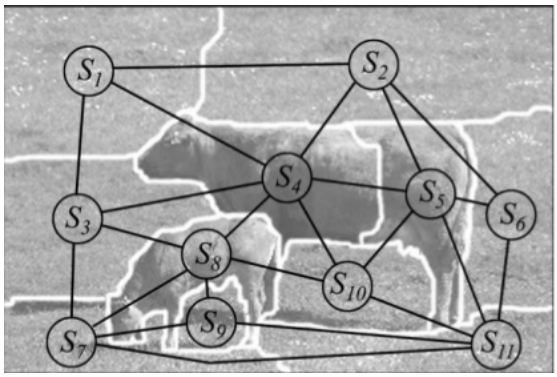
the system or the strongest direct interactions. Moreover, representation error is also easy to affect directly by adding edges to the model to capture more of the direct interactions between the elements.

Consider Fig. 1.2, which shows the structure (i.e., direct dependencies) of two typical PGMs: for labeling image segments with identities of the objects in the picture in Fig. 1.2a and for hypertext classification in Fig. 1.2b. Both models have been constructed using straightforward pieces of intuition: that adjacent segments tend to have correlated object identities for image segmentation (a cow is often next to grass), and that webpages that link to each other tend to have correlated topics (professors tend to link to webpages of their students). These simple pieces of intuition have resulted in densely connected models, however, and it is counterintuitive that removing some of the dependencies from the model can improve the model performance.

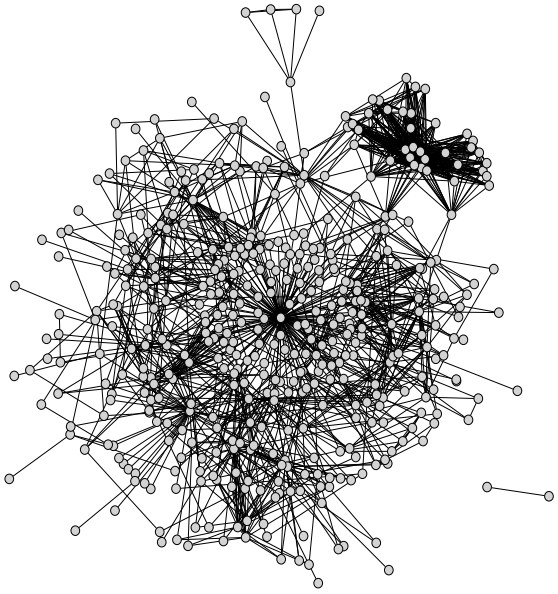
2. Inference, even approximate, is provably intractable in most graphical models. Therefore, inference algorithms need to introduce approximations to achieve acceptable computational cost. The nature of those approximations is specific to a concrete inference approach. As a result, **inference error** is harder to understand and interpret, especially for a non-expert. Moreover, the limits in which the inference error can be affected by adjusting parameters of an inference algorithm are also quite narrow: the fundamental choices of the approximation are mostly made at the algorithm design stage, not at parameter selection stage.

As a result, during the model design too much attention is often paid to optimizing the representation error, as it is a more intuitive component of the total error: practitioners are reluctant to give up well-understood benefits of a richer model for hard-to-characterize improvements in inference accuracy of a simpler model.

The impact of model complexity on **computational efficiency** is more straightforward. The complexity of *exact inference* grows extremely rapidly with the complexity of the model to the point of being infeasible for most commonly used models, such as those in Fig. 1.2. *Approximate inference* is much more efficient. However, for commonly used models even approximate inference is often not efficient enough. The main



(a) A graphical model for labeling image segments with the identities of corresponding objects. Every node corresponds to a random variable. This picture is from Gould et al. (2008).



(b) The structure of a graphical model used by Taskar et al. (2002) for hypertext documents classification. Every node corresponds to a webpage, every edge - to a hyperlink.

Figure 1.2: Connectivity patterns of typical graphical models. Both graphs above have a large number of loops, making it necessary to use heuristic inference approaches without any quality guarantees.

problem with approximate inference is that not only does the cost of a single iteration grow with model complexity, but also the *number of iterations required for convergence* grows. Moreover, while the cost of single iteration grows in a predictable manner (linearly with the model size), the number of required iterations can grow abruptly, and can be very different for two seemingly similar models. As a result, existing approximate inference techniques are often too costly, especially during *test time* for applications that are interactive or require near real-time results.

The issue of inference complexity is further exacerbated by the limits in theoretical understanding and ability to predict the inference complexity given a model. Merely predicting whether belief propagation, a popular approximate inference approach, will *converge at all* for a given model is a hard open problem, with solutions only for special cases (Mooij and Kappen, 2007). The limits in understanding the total computational cost of approximate inference lead practitioners to concentrating on the complexity of a single iteration (notice the analogy with the two sources of approximation error discussed above). As a result, the overall inference complexity is often underestimated.

In this thesis, we argue that by eliminating the requirement that the model be interpretable in terms of the application domain, and instead learning simpler *models that admit efficient exact inference,* one can not only obtain the same or better accuracy as the state of the art approaches involving approximate inference, but also significantly reduce the computational requirements. We propose two novel approaches for learning tractable model (chapters 2 and 3) and demonstrate their performance empirically on real-world datasets.

1.0.2 Inference in presence of nuisance variables

In general, the variables of the full model can be subdivided into three groups: query Q, evidence E and nuisance N. At test time, the assignment to only E is known, and the user is interested in the conditional distribution $P(Q \mid E)$. However, the standard inference procedures (e.g., Pearl, 1988; Geman and Geman, 1984), only distinguish between known and unknown variables. As a result, not only does computing the distribution of interest $P(Q \mid E)$ involve implicit computation of $P(Q, N \mid E)$, but also the conditionals $P(Q \mid E)$ and $P(N \mid E)$ are computed *with the same accuracy,* thereby wasting computation on the irrelevant information.

In chapter 4, we introduce an approximate inference approach that improves on the successful residual belief propagation algorithm by focusing the computation on the query and only inferring the nuisance conditional *to the extent necessary for approximating the query conditional well*. Unlike existing approaches, ours does not affect the accuracy of the inference results for the query at convergence, and as we demonstrate empirically on large-scale real-life models, brings significant speedups over the state of the art.

Although the three general topics of this thesis are quite distinct in terms of the technical approach, they all share the focus on significantly improving the **testing-time computational efficiency**. Fast computations during testing make it possible for a whole new class of applications that are either interactive or require close to real time inference to benefit from the formalism of probabilistic graphical models and related techniques. Note that the *training time efficiency* is not as crucial for many applications, because the model can be trained in advance.

1.1 Probabilistic graphical models

To place the contributions of this thesis in context, here we review the formalism of probabilistic graphical models, along with the key complexity results and their impact on the standard process of applying PGMs in practice.

There are several different formulations of probabilistic graphical models, but they all share a key idea of approximately representing a high-dimensional probability distribution as a product of low-dimensional components:

$$P(X) = \frac{1}{Z} \prod_{\psi_\alpha \in \mathcal{F}} \psi_\alpha(X_\alpha), \tag{1.2}$$

where[1] the low-dimensional nonnegative components ψ_α are called *factors* of *potentials* and Z is the normalization constant, also called a partition function. The set $\mathcal{F}$ of model factors induces a *graphical structure* on X, whose properties determine the complexity of inference in the model. Generally, the graph edges $\mathbb{T}$ link the variables of X that belong to the same factor ψ_α. Depending on the edges, there exist directed and undirected graphical models:

1. **Undirected** graphical models. The most common instance of this class is the formalism of **Markov networks,** where there is a node in the graph for every variable $x_i \in X$, and an edge $(i-j) \in \mathbb{T}(\mathcal{F})$ if and only if there exists a factor that depends both on x_i and x_j : $\exists \psi_\alpha \in \mathcal{F}$ s.t. $x_i, x_j \in X_\alpha$. An

[1] Notation remark: throughout the document, we use small letters (x, y) to denote variables, capital letters (X, C) to denote sets of variables, boldface $(\boldsymbol{x}, \boldsymbol{C})$ to denote assignments, and double-barred font $(\mathbb{C}, \mathbb{D})$ to denote sets of sets.

advantage of Markov networks, besides their simplicity and versatility, is the fact that most of the inference and learning complexity results are stated in terms of the Markov network graph structure, making such models easier to analyze theoretically.

There are also alternative formalisms for undirected graphical models, which are somewhat more specialized, but also have useful properties that Markov networks lack. We will introduce two such formalisms, *junction trees* and *factor graphs* in the main body of this thesis, as required to describe and analyze our learning and inference approaches.

2. **Directed** graphical models. The most common models of this class are **Bayesian networks,** where there is a node for every variable of X and the directed edges are such that the resulting graph is acyclic. Given a directed acyclic graph (DAG) over X, a Bayesian network defines a factorized distribution as a product of conditionals:

$$P(X) = \prod_{x_i \in X} P(x_i \mid \text{Pa}(x_i)), \tag{1.3}$$

where $\text{Pa}(x_i)$ is a set of parents of x_i in the DAG defined by the directed edges $\mathbb{T}$.

Observe that there are two key differences between the Bayesian network factorization (1.3) and the general factorization (1.2). First, there is no normalization constant Z in (1.3) - implicitly, $Z = 1$. Second, instead of arbitrary nonnegative factors ψ_α, Bayesian network factorization uses conditional distributions. Efficient parameter learning via using conditional distributions as potentials, along with the absence of a normalization constant, are attractive properties of Bayesian networks. However, inference in general is no easier in Bayesian networks than in undirected models, and keeping the graph with edges $\mathbb{T}$ acyclic may be problematic in some models, especially in relational settings. As a result, neither directed nor undirected graphical model formulation can be said to be uniformly better than the other.

All of the novel approaches described in this thesis operate with undirected graphical models. When necessary to put our work in context, we will also discuss relevant existing results for directed models.

Because the factorized distributions (1.2) and (1.3) have fewer degrees of freedom (the number of independent values of the factors) than the total number of assignments to X, it follows that not every distribution $P(X)$ can be represented exactly as a compact factorized model. To better understand the constraints that the factorization (1.2) places on the set of distributions that can be represented exactly, it is useful to think about the statistical dependencies encoded by the graphical model structure. For a Markov network with variables X and edges $\mathbb{T}$, denote $MB(x_i, \mathbb{T}) = \{x_j \mid (i-j) \in \mathbb{T}\}$ the **Markov blanket** of x_i. In other words, the Markov blanket of a variable in a Markov network is simply the set of the immediate neighbors of that variable in a graph. It follows that a variable is independent of all other variables given its Markov blanket:

Theorem 1 (Hammersley and Clifford, 1971)**.** *For any distribution $P(X)$ that factorizes as* (1.2) *with the corresponding Markov network edges $\mathbb{T}$, it holds that*

$$\forall x_i \in X \quad (x_i \perp X \setminus (x_i \cup MB(x_i, \mathbb{T})) \mid MB(x_i, \mathbb{T}))\,. \tag{1.4}$$

Moreover, with some restrictions, the reverse also holds. Denote $\mathbb{I}(\mathbb{T})$ to be the set of conditional independence assertions (1.4) induced by the edges $\mathbb{T}$. Analogously, for an arbitrary distribution $P(X)$, denote $\mathbb{I}(P)$ to be the set of conditional independence assertions that hold for $P(X)$. Then $P(X)$ can be

factorized according to the Markov network structure $\mathbb{T}$ whenever the graph structure does not introduce additional independence assertions compared to P :

Theorem 2 (Hammersley and Clifford, 1971)**.** *For a positive distribution $P(X)$ and a set of edges $\mathbb{T}$ between the elements of X such that $\mathbb{I}(\mathbb{T}) \subseteq \mathbb{I}(P)$, there exists a set of factors $\mathcal{F}$ such that $P(X) = \frac{1}{Z}\prod_{\psi_\alpha \in \mathcal{F}} \psi_\alpha(X_\alpha)$ and $\mathbb{T}(\mathcal{F}) = \mathbb{T}$.*

An analogous set of theorems connecting the form of factorization (1.3) with the set of conditional independence assertions for the factorized distribution $P(X)$ also holds for the Bayesian networks, with $\mathbb{I}(\mathbb{T}) = \cup_{x_i \in X}(x_i \perp NonDescendants(x_i) \mid Pa(x_i))$.

Theorems 1 and 2 formalize the intuitive notion that edges in probabilistic graphical models encode direct dependencies of the variables on each other, which is key to understanding the approximations that are introduced by a given PGM structure $\mathbb{T}$.

1.1.1 Key problems and complexity results

Having established the connection between the factorization (1.2) corresponding to a probabilistic graphical model and the induced approximations in the form of conditional independence assertions, we now review the basic problems that need to be solved in order to use a PGM in practice. A factorized distribution (1.2) and the corresponding graphical model are only approximation tools, and the real-world data $\mathcal{D}$ is not sampled from graphical models. Therefore, to successfully apply a PGM in a certain problem domain, one needs to first identify a graphical model that approximates the underlying real-world distribution accurately, and second, to be able to perform accurate *inference* in that model. In turn, identifying an accurate graphical model is typically decomposed into *structure learning,* where one determines the scopes X_α of the potentials ψ_α in (1.2), and *parameter learning,* where the actual values of factors ψ_α are determined given the known scopes X_α. In this thesis, we restrict the attention to learning PGMs from the **fully observed data,** where $\mathcal{D}$ is assumed to be a set of samples $\boldsymbol{X}$ from the true unknown distribution $P_{\mathcal{D}}(X)$, where the value of every $x_i \in X$ is known for every sample.

Next, we review the existing complexity results, discuss their implications for the typical workflow of applying graphical models, and propose alternatives that eliminate some of the existing drawbacks.

Probabilistic inference

Inference is a fundamental problem in probabilistic graphical models that arises not only at test time, but also often during parameter and structure learning. The basic operation of probabilistic inference is *marginalization:*

$$P(Y) = \sum_{\{\mathbf{x_{i_1}}, \ldots, \mathbf{x_{i_m}}\} \in X \setminus Y} P(Y, \mathbf{x_{i_1}}, \ldots, \mathbf{x_{i_m}}),$$

where $P(X)$ is a factorized distribution (1.2). It is assumed that the set of factors $\mathcal{F}$ is known, but the value of the normalization constant Z is unknown. In general, even for compact models, not only is exact inference NP-hard in general (Cooper, 1990), but even computing bounded approximations is NP-hard (Dagum and Luby, 1993). However, there exists an exact inference algorithm, namely sum-product (Shafer and Shenoy, 1990), with complexity exponential in the *treewidth* of the graph induced by the model edges $\mathbb{T}(\mathcal{F})$. Treewidth (Robertson and Seymour, 1984) is the size of the largest clique in the

triangulated graph defined by the edges $\mathbb{T}$. It follows that for models with low treewidth exact inference is tractable[2].

The existence of an exact inference algorithm with complexity exponential only in graph treewidth does not contradict the NP-hardness results for the general case, because even compact models with bounded-degree graph can have large treewidth. For example, a 2D grid over variables X, where every variable is directly connected to at most 4 others, has treewidth of $\sqrt{|X|}$, making exact inference in grid-structured models intractable. In fact, most of the "naturally occurring" PGM structures have high treewidth.

Low-treewidth models are the most extensively studied class of tractable models, but there also other classes, such as feature graphs (Gogate et al., 2010) and arithmetic circuits (Lowd and Domingos, 2008), which allow for high treewidth and instead restrict the internal structure of the potentials ψ_α. To obtain efficient exact inference, high-treewidth tractable models rely on exploiting *context-specific independence* (Boutilier et al., 1996), whereby certain variable assignments $\boldsymbol{X}$ "disable" some of the dependencies for the rest of the variables $X \setminus \boldsymbol{X}$ that are in general present in the model. For example, in a car engine diagnostic system, the variable "engine starts" is in general dependent on the variable "battery charge level", but if the variable "tank has gas" is false, then the engine will not start regardless of the battery charge, making the two variables independent.

To summarize, even approximate inference in compact graphical models is intractable in general, unless some extra properties, such as low treewidth or context-specific independence, hold for the factorization (1.2). As a result, approximate inference algorithms with few guarantees on the result quality (Pearl, 1988; Geman and Geman, 1984; Jordan et al., 1999) are typically used in practice.

MAP inference

Often, the user is interested in inferring the most likely state of the system in question given the evidence. This class of problems, called *structured prediction* problems, spans areas from optical character recognition (Kassel, 1995) to natural language processing (Taskar et al., 2004) to image segmentation (Ladicky et al., 2009) and scene understanding in computer vision. In the context of probabilistic graphical models, structured prediction problems give rise to the problem of *maximum a posteriori (MAP) inference:*

$$X^* = \arg\max_X P(X),$$

where $P(X)$ is a factorized distribution (1.2). In general, similar to the marginalization problem, MAP inference is NP-hard in general for compact PGMs (Shimony, 1994). Moreover, computing the MAP assignment for a marginal $P(Y)$ for $Y \subset X$ is $\mathrm{NP}^{\mathrm{PP}}$-complete (Park, 2002). However, for important special cases the MAP inference problem is tractable.

First, the graph structure $\mathbb{T}(\mathcal{F})$ of the graphical model for (1.2) can be exploited in a similar way to the marginalization problem. Whenever exact marginalization is possible due to low treewidth of the graph or context-specific independence, exact MAP assignment can be found using the *max product* algorithm (essentially, dynamic programming), which replaces summations in the sum-product algorithm with maximizations (Pearl, 1988).

Second, for intractable PGMs, when (a) max-product algorithm converges and (b) a joint assignment $\boldsymbol{X}^*$ is found that is consistent with all the local max-marginals, the resulting $\boldsymbol{X}^*$ can be shown to be a

[2]We will also say that a model is tractable if it admits tractable exact inference.

strong local optimum (Weiss and Freeman, 2001a). The term strong here refers to the fact that any $\boldsymbol{X}'$ that differs from $\boldsymbol{X}^*$ in only a few variable values (the number of the different variables depends on the model) is guaranteed to have $P(\boldsymbol{X}') < P(\boldsymbol{X}^*)$. Moreover, for convexified variations of max-product, a joint assignment consistent with max-marginals at convergence can be shown to be globally optimal (Wainwright et al., 2005). However, the max-product algorithm is not guaranteed to converge in general, and in practice tends to have more brittle convergence properties than sum-product.

Finally, in the case of binary variables and submodular pairwise factors $\psi_{ij}(x_i, x_j)$ (which is an important special case of PGMs, popular in e.g., computer vision), the MAP inference problem can be cast as a graph cut problem (Kolmogorov and Zabih, 2004). As a result, the most probable assignment can be found exactly and efficiently using any existing graph cut algorithm even for densely connected models (Boykov and Kolmogorov, 2004). Importantly, the marginalization problem for such models remains intractable. This mismatch in complexity of the two problems stems from the fact that computing the normalization constant Z from (1.2) is only required for marginalization, but not for MAP inference. Still, even though many models important in practice are covered by the exact graph cuts-based techniques, such approaches are not universally applicable. Even restricting attention to pairwise interactions, many models do not admit a graph cut MAP formulation.

We also notice that in practice one needs to choose carefully between MAP approaches and maximizing one-dimensional marginals $P(x_i)$ of $P(X)$, depending on the actual penalty function of the problem domain. For example, in the OCR setting one can argue that it is better to measure accuracy by the number of *words* that have been correctly decoded as opposed to the number of *individual characters*, and therefore MAP decoding is the optimal answer. On the other hand, in image segmentation problems (Shotton et al., 2006) the accuracy is typically measured by the total number of correctly labeled pixels, not by how many full images have been labeled perfectly. In the applications where such Hamming distance-like error measures are used, it is typically better to use the assignment that maximizes the one-dimensional marginals $P(x_i)$, thereby returning to the marginalization problem.

Parameter learning

Given the model structure, it is desirable to find the parameters that result in the best approximation of the true distribution $P_{\mathcal{D}}(X)$ that generated the data $\mathcal{D}$ with the factorized distribution (1.2). With KL divergence (Kullback and Leibler, 1951) as the measure of approximation quality, we need to minimize

$$KL(P_{\mathcal{D}}(X)||P) = \sum_{\boldsymbol{X}} P_{\mathcal{D}}(\boldsymbol{X}) \log \frac{P_{\mathcal{D}}(\boldsymbol{X})}{P(\boldsymbol{X})} = -H(P_{\mathcal{D}}) - \sum_{\boldsymbol{X}} P_{\mathcal{D}}(\boldsymbol{X}) \log P(\boldsymbol{X}),$$

where the first component is the entropy of the true distribution that does not depend on the model, and the second component is the negative log-likelihood of the model. Because the true distribution $P_{\mathcal{D}}(X)$ cannot be observed directly, it is replaced with the empirical distribution $\overline{P}_{\mathcal{D}}(X)$ of the data, resulting in the log-likelihood

$$LLH(P \mid \mathcal{D}) = \sum_{\boldsymbol{X}} \overline{P}_{\mathcal{D}}(\boldsymbol{X}) \log P(\boldsymbol{X}) = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{X} \in \mathcal{D}} \log P(\boldsymbol{X}).$$

It follows that one needs to maximize log-likelihood of the model. Because of the finite number of the datapoints, the empirical distribution $\overline{P}_{\mathcal{D}}(X)$ is only an approximation of the unknown underlying

distribution from which the data is generated. Therefore, to prevent overfitting (i.e., learning the noise in the data), a regularization term, which introduces a bias towards uniform potentials, is added to the log-likelihood.

For Bayesian networks, maximizing the likelihood is achieved simply by plugging in the empirical conditional probabilities $\overline{P}_{\mathcal{D}}\left(x_i \mid \text{Pa}(x_i)\right)$, so the parameter learning problem is trivial. For undirected models, however, there is no closed form expression for the optimal parameters. Moreover, even computing the value of the likelihood itself requires inference in the model, which is typically intractable, as was discussed above. As we discuss in detail in chapter 3, if exact inference with the model is feasible, then optimizing the log-likelihood is a convex optimization problem that can be solved very efficiently with state of the art techniques such as L-BFGS (Liu and Nocedal, 1989), despite high dimensionality of the parameters. Therefore, for low-treewidth and other tractable models efficient exact parameter learning is possible. For some low-treewidth PGM formalisms, closed-form expressions for optimal parameters also exist (c.f. chapter 2). There also exist closed-form expressions for approximately optimal parameters for high-treewidth models (Abbeel et al., 2006), but the more dependencies not reflected by the PGM structure the true underlying distribution has, the worse is the approximation quality of such parametrization, and in practice it is used rarely.

In general, for high-treewidth models there are few alternatives to optimizing log-likelihood, even though computing the objective exactly is intractable. It follows that one needs to introduce approximations to make the objective tractable. Two main types of approximations are possible. First, one can replace the log-likelihood objective with a more tractable alternative, such as pseudolikelihood (Besag, 1974). Pseudolikelihood is the sum of likelihoods of single variables conditioned on their respective Markov blankets:

$$PLLH(P \mid \mathcal{D}, w) = \sum_{\boldsymbol{X} \in \mathcal{D}} \sum_{x_i \in X} \log P(\mathbf{x_i} \mid \mathbf{MB}(\mathbf{x_i}), \boldsymbol{E}, w), \tag{1.5}$$

where $\mathbf{MB}(\mathbf{x_i})$ denotes the values of all the variables of X that share a feature with x_i. Second, one can apply an approximate inference method to the model and use the results as if the inference was exact. Pseudolikelihood objective is attractive because it is both tractable, and, for sufficiently expressive models, in the large sample limit yields the same parameters as optimizing the log-likelihood (Gidas, 1988). However, the requirement of sufficient expressivity of the model is essential: it is required that the model be able to represent the generating distribution $P_{\mathcal{D}}$ exactly. Because graphical models used in practice usually represent simplifications of the true underlying distribution, the requirement of the sufficient expressive power of the model is often violated, invalidating the guarantee.

Using approximate, instead of exact, inference for computing the value and gradient of the log-likelihood during parameter learning is problematic, because even approximate inference is NP-hard, and therefore there are no guarantees on the accuracy of the computed objective, or on the quality of the resulting parameters. Moreover, log-likelihood convexity may be violated with approximate inference, resulting in convergence issues of the convex optimization techniques.

To summarize, just as with the problem of inference in probabilistic graphical models, with the problem of parameter learning there exists a sharp contrast between tractable models, where efficient exact parameter learning is possible, and the general compact high-treewidth models, where one needs to resort to approximations with little, if any, guarantees on the result quality.

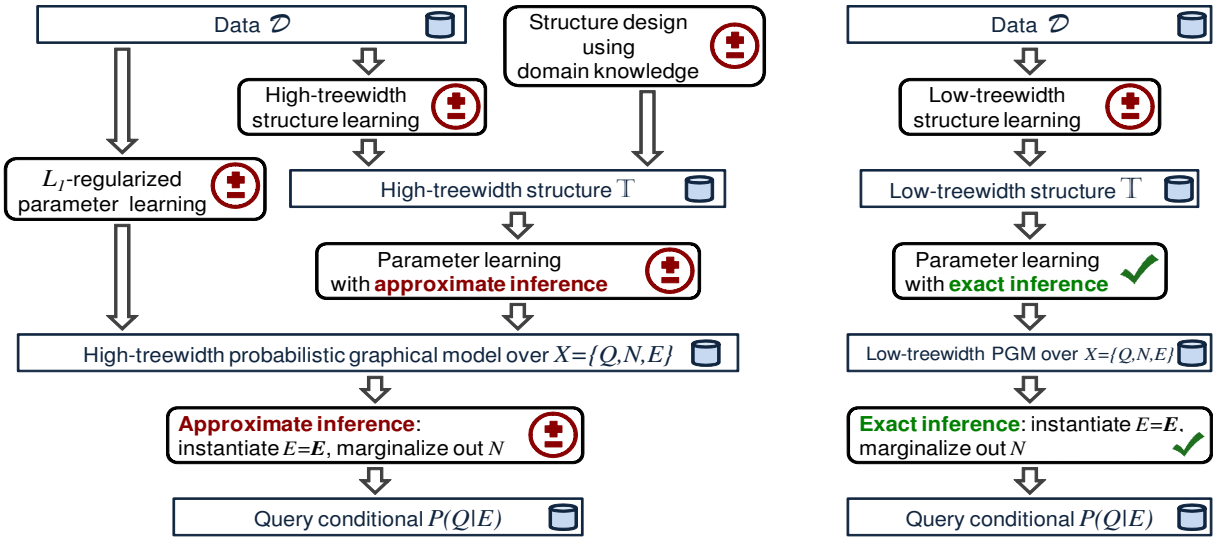
Structure learning

Learning the optimal structure of a factorized model (1.2) from finite data $\mathcal{D}$ has two conflicting objectives. On the one hand, one needs to discover the direct dependencies that hold in the true distribution $P_{\mathcal{D}}$ to obtain a model that is expressive enough to approximate the true distribution well. On the other hand, it is desirable to avoid spurious dependencies that exist in the empirical distribution $\overline{P}_{\mathcal{D}}(X)$, but not in the true generating distribution, because of the finite-sample noise. Spurious edges, and the corresponding potentials introduced into the factorization (1.2), not only make the model more prone to overfitting during the parameter learning stage, but also increase the computational complexity of inference in the model. Maximizing the likelihood of the structure $\mathbb{T}$ favors including as many edges into the learned structure as possible. In fact, a fully connected PGM can represent any distribution over X exactly, and therefore will have the highest possible likelihood. Therefore, to prevent overfitting and overly complex models, one needs to bias the learning process towards sparser models. There are two main methods of introducing such a bias towards simplicity. First, one can explicitly restrict the space of models over which the likelihood maximization is performed, for example, to low-treewidth models, or to graphs with a uniform bound on the variable degree. Secondly, one can maximize over the space of all possible models, but introduce a regularization term penalizing model complexity. In both cases, however, structure learning typically contains intractable steps, and the approaches used in practice have few quality guarantees.

Learning optimal models over restricted space of structures is provably intractable for most settings. A notable exception is the classical result of Chow and Liu (1968), which states that the most likely *tree structures* can be learned efficiently in $O(|X|^2 \log |X|)$ time. However, even extending the space of structures to polytrees (directed trees where variables are allowed to have more than one parent) makes learning the most likely structure an NP-hard problem (Dasgupta, 1999). Learning most likely low-treewidth models is NP-complete (Karger and Srebro, 2001) for treewidth greater then 1, as is learning the structure of general directed models (Chickering, 1996). However, graphical models with limited degree graphs can be learned in the probably approximately correct (PAC) sense (Abbeel et al., 2006). Also, one can learn in polynomial time low-treewidth structures which are within a constant factor of the log-likelihood of the optimal structure (Karger and Srebro, 2001).

Explicitly regularizing the likelihood to favor simpler structures can in turn take two forms. First, one can use a regularization term that penalizes the number of parameters in the model, such as BIC score (Schwarz, 1978). Then, a local search is performed in the space of structures $\mathbb{T}$ to maximize the regularized likelihood (e.g., Teyssier and Koller, 2005). Second, one can use *sparsity-inducing prior for the* model *parameters,* such as an L_1 penalty, and reduce the structure learning to (a) parameter learning in the full model and (b) dropping the uniform factors from the model afterwards (e.g., Lee et al., 2006). Both general approaches to structure regularization require inference to compute the objective. Therefore, for high-treewidth models there are no guarantees on the quality of the resulting structure, because even approximate inference is intractable. Notably, if inference were tractable, L_1-regularized likelihood would be possible to optimize exactly, because the corresponding optimization problem is convex. For local search approaches, even with exact likelihood computation the resulting structure is only guaranteed to be a local optimum in the space of possible structures. In practice, both structure learning techniques based on local search, and L_1-regularized parameter learning with approximate inference are quite popular because of their moderate computational efficiency and good quality of the learned models in practice. However, the fundamental limitations arising from intractability of inference lead to a lack of formal quality guarantees.

To summarize, in most settings learning the optimal PGM structure is provably intractable. Moreover, the



(a) Typical approaches of applying probabilistic graphical models involve high-treewidth structures. As a result, parameter learning and inference are intractable and one needs to resort to algorithms without quality guarantees. Different paths from top to bottom of the diagram represent different complete approaches. For example, learning the PGM parameters with a sparsity-inducing L_1 regularization can be replaced with a combination of separate structure learning and parameter learning steps.

(b) An alternative workflow based on low-treewidth graphical models. Although the set of available models is not as expressive as in the high-treewidth case, parameter learning and inference can be done exactly, reducing the number of sources of error to only one (structure learning).

Figure 1.3: A comparison of standard high-treewidth graphical models workflows (a) and an analogous process in the low-treewidth setting (b). Symbol $\pm$ denotes stages of computation where there are no quality guarantees for the approaches that are used in practice. A check mark symbol ✓ denotes the stages where the computation can be performed exactly.

existing structure learning approaches that do posses quality guarantees in the finite sample case restrict the model connectivity in advance (Chow and Liu, 1968; Karger and Srebro, 2001; Abbeel et al., 2006).

1.1.2 Typical use cases and directions for improvement

The overview of complexity of the different stages in working with high-treewidth probabilistic graphical models is roughly summarized schematically in Fig. 1.3a. Typically, practitioners choose to use high treewidth models, because of the following advantages:

1. **Representational power.** The set of compact high-treewidth models is more broad than that of low-treewidth models. Therefore, high-treewidth models are likely to be able to approximate the true underlying distribution of the system more accurately.

2. **Computational benefits in structure learning.** Maintaining low treewidth of the candidate structures during learning adds a significant burden either computationally, because even computing the

treewidth of an arbitrary graph is NP-complete (Arnborg et al., 1987), or in terms of representation and algorithm design, to guarantee that every candidate structure has low treewidth and does not need explicit checking.

3. **Straightforward use of domain knowledge.** Sometimes, information about the direct dependencies in the system in question is available. For example, in a plant monitoring scenario, the connectivity of different subsystems of a plant is known, and it is highly desirable for the model structure to reflect that information. In such cases, an expert may even design the structure of a model by hand, according to the domain knowledge and without regard to the treewidth.

   Also, high-treewidth structures arise naturally in *relational* models (Friedman et al., 1999; Richardson and Domingos, 2006; Taskar et al., 2002), where every edge corresponds to an *instance* of a relation, such as friendship for social network modeling, and the connectivity of the model is then determined by a social graph of the population in question.

4. **Interpretability.** Sometimes the structure of a PGM not only serves as an intermediate result of computing the query conditional $P(Q \mid \boldsymbol{E})$, but is of interest in its own right to the practitioner. For example, the edges of a model learned from data may be used to select possible causal links between different components of the system that need to be investigated (Friedman, 2004).

However, if computing the query conditional $P(Q \mid \boldsymbol{E})$ is the only goal of applying a graphical model, as is often the case, then some of the concerns above are irrelevant (interpretability and to some extent domain knowledge issues). Moreover, as Fig. 1.3a indicates, even the advantages of high-treewidth models that are directly related to the accuracy of answering the probabilistic query (representational power and extra difficulties with learning low-treewidth models), are counterbalanced by intractability of the fundamental problems on every step of the process and lack of quality guarantees for the involved algorithms.

Because the high-treewidth models are typically unable to realize the full potential of the representation power due to inference difficulties, it is natural to attempt to achieve the same end results using simpler models, where the decrease in the representative power is compensated for by the improvements in inference accuracy. This is exactly the approach we take in chapters 2 and 3 of this thesis. The key idea of our alternative approach, shown schematically in Fig. 1.3b, is to restrict consideration to the models that admit efficient exact inference. In particular, we restrict the models to have *low treewidth.* Let us review the advantages of the low-treewidth workflow of Fig. 1.3b.

1. **Exact computations on most of the steps of the pipeline.** In contrast to Fig. 1.3a, in the low-treewidth workflow of Fig. 1.3b the only source of approximation errors is the structure selection stage. In this thesis, we argue that in many applications the advantages of the low-treewidth approach, namely exact inference and parameter learning, will compensate for the smaller expressive power of the low-treewidth structures. In other words, we argue that while high-treewidth models can *potentially* approximate the query conditional distribution more accurately than low-treewidth models, the lack of algorithms to construct such an approximation and process it efficiently largely prevents one from *realizing* that potential advantage.

2. **Significantly higher test-time computational efficiency.** Exact inference in low-treewidth graphical models is much faster than approximate inference in high-treewidth models. Moreover, the complexity of exact inference in low-treewidth models is *highly predictable.* As a result, low-treewidth models are well-suited for applications that are sensitive to the response time. Because the total complexity of approximate inference for high-treewidth models is often hard to predict (e.g., Mooij and Kappen, 2007), high-treewidth are often inapplicable in latency-sensitive settings.

Let us briefly illustrate some of the opportunities that models with low test-time latency open up. In a web search setting, the user may wish to restrict the search to the webpages of a certain type, such as homepages of university professors. A model exists that uses the link structure of the webpages to improve the classification accuracy (Craven et al., 1998; Taskar et al., 2002) see also experimental results in chapters 3 and 4 of this thesis). However, classifying every webpage in advance using such a model on a full index of the internet stored by a search engine is problematic for two reasons. First, the sheer scale of the web graph, with modern search engines having petabyte-sized web indices (Peng and Dabek, 2010), makes inference extremely demanding computationally. Second, search engines continuously change their web indices, with many popular pages being updated once every several minutes (Peng and Dabek, 2010). As a result, the underlying model does not stay fixed for a long enough time for the inference to finish. An alternative to pre-classifying every web page in an index is to first retrieve the results without regard to the webpage type, and then, for the final filtering, to use the model for type inference only on the much smaller subgraph corresponding to those results. Such an on-demand approach is much less demanding computationally, but requires the inference in the model to be guaranteed to run in a fraction of a second - otherwise the user will have to wait too long for the search results.

In computer vision, applying graphical models to the problem of detecting objects in images yields state of the art accuracy (Gould et al., 2008). However, for applications such as pedestrian detection and terrain labeling for autonomous driving (Thrun et al., 2006; Enzweiler and Gavrila, 2011) it is crucial to provide near-realtime results, which is beyond the capabilities of approaches using approximate inference with high-treewidth models. In the experimental results of chapter 3, one can see that traditional high-treewidth models take 1 second on average to perform inference. Our approach building on low-treewidth models, on the other hand, takes only 0.02 seconds per image, sufficient for real-time processing of video data, without sacrificing accuracy.

3. **Lack of algorithm-specific adjustments.** Approximate inference approaches for high-treewidth models often involve algorithm-specific parameters that have to be adjusted for the algorithm to work well on a given model. As result, the end user typically has to bear an additional burden of configuring the approximate inference algorithm (setting the damping level for belief propagation (Mooij and Kappen, 2005) or the umber of samples for Gibbs sampling (Geman and Geman, 1984)). Because inference is typically required in the inner loop to compute model likelihood and gradient during parameter learning, the tweaking of an approximate inference algorithm is required during both training and testing. In contrast, exact inference in low-treewidth models does not require any parameters. As a result, the only parameter the end-user needs to set with low-treewidth models is the regularization constant during training (for example, using cross-validation).

In this thesis, we identify and explore two directions of improving the graphical model workflows of Fig. 1.3, both for the case when the exact form of the model does not matter to the end user, and for the case when a high-treewidth model is required because of domain-specific concerns:

1. For cases when the exact form of the model is not important to the practitioner, and the accuracy and efficiency of computing the query conditional distribution $P(Q \mid \boldsymbol{E})$ are the only criteria for success, we argue that the low-treewidth approach of Fig. 1.3b is often a better alternative than the high-treewidth approach of Fig. 1.3a. As the resulting accuracy of the low-treewidth is effectively determined by the quality of a structure learned from data, we propose *novel algorithms for efficiently learning accurate low-treewidth models* for both propositional and relational settings. We demonstrate empirically that the resulting tractable models yield the same or better accuracy of the

conditional query distribution $P(Q \mid \boldsymbol{E})$ compared to the high-treewidth approaches.

2. For settings where a high-treewidth model is necessary, we propose an approach that allows one to *speed up marginalizing the nuisance* unobserved *variables* N out of the conditional $P(Q, N \mid \boldsymbol{E})$. In many settings, it is often the case that only a small number of unobserved variables Q is of interest to the user, but there is a large number of unobserved nuisance variables that have to be marginalized out. For example, in an automated system for patient monitoring (Beinlich et al., 1988), the only variable of direct interest may be whether the patient needs immediate attention of the hospital staff. In a smart home setting (Pentney et al., 2006), the variable of interest may be whether a certain room is likely to be occupied in the near future: to save energy, the smart home would turn the air conditioning off in rooms that are not likely to be occupied soon.

   Our approach exploits the observation that not every nuisance variable affects the query conditional $P(Q \mid \boldsymbol{E})$ to the same extent. It incrementally refines the estimate of the query conditional by taking the strongest dependencies into account first, and only touches the nuisance variables to the extent necessary for computing the query conditional. For example, in the extreme case that $(x_i \perp Q \mid \boldsymbol{E})$, the nuisance variable x_i would be ignored altogether. As a result, we demonstrate significant speedups in the convergence of the estimate of the query conditional $P(Q \mid \boldsymbol{E})$.

1.2 Thesis overview and contributions

Here, we outline the organization of the thesis and the main contributions.

Chapter 2. Here, we consider a problem of learning a low-treewidth probabilistic graphical model that accurately approximates a given distribution $P(X)$ given data $\mathcal{D}$. We propose an novel polynomial time algorithm with quality guaranteed for learning fixed-treewidth graphical models. More specifically, we claim the following contributions:

1. A polynomial time algorithm for learning fixed-treewidth graphical models with PAC learnability guarantees for distributions exactly representable with strongly connected maximal fixed-treewidth graphical models, and graceful degradation guarantees for distributions that are only approximately representable with fixed-treewidth graphical models.

2. A theoretical result that provides a polynomial time upper bound on conditional mutual information of arbitrarily large sets of random variables, which not only forms a basis of our structure learning algorithm, but can also be used by other constraint-based structure learning approaches.

3. A heuristic version of the algorithm mentioned above that forgoes the result quality guarantees of the original version, but works much faster in practice.

4. Evaluation on real datasets showing that low-treewidth model have competitive approximation quality with high-treewidth models.

Chapter 2 is a significantly extended version of (Chechetka and Guestrin, 2007).

Chapter 3. In this chapter, we consider a problem of learning a low-treewidth probabilistic graphical model that accurately approximates the query conditional $P(Q \mid \boldsymbol{E})$. Compared to the setting of chapter 2, here we have additional information about both the set of evidence variables E and its assignment at test time $\boldsymbol{E}$. This extra information lets us *tailor the structure* of low-treewidth models *to a particular value of the evidence* variables *at test time*. We propose a novel algorithm for learning adaptive low-treewidth

conditional models with evidence-specific structure. The use of evidence-specific structure lets one expand the expressive power of the model beyond the capabilities of any single tractable structure, and at the same time retain the advantages of efficient exact inference and parameter learning. More specifically, we claim the following contributions:

1. A novel way to exploit information about the *values* of variables which are observed at test time to select the structure of discriminative probabilistic graphical models that is specific to the evidence values at hand. The key advantage of our approach over existing work on learning tractable evidence-specific models is the ability to guarantee low treewidth of the resulting models, and thus tractability of exact inference, not only in a propositional, but also in a relational setting.

2. A general framework that allows one to leverage the existing work on learning the structure of propositional tractable models and low-dimensional conditional density estimation to construct algorithms for learning discriminative models with evidence-specific structure.

3. An extension of the general framework for learning discriminative models with evidence-specific structure to the relational setting. Importantly, with our generalization, one can still use *propositional* algorithms for structure learning and low-dimensional conditional density estimation as building blocks.

4. An empirical evaluation demonstrating that in the relational setting our approach has equal or better accuracy than the state of the art algorithms for densely connected models, and at the same time is much more efficient computationally.

Chapter 3 is an extended version of (Chechetka and Guestrin, 2010b).

Chapter 4. Here, we consider the problem of speeding up the convergence of a particular approximate inference algorithm, residual belief propagation (Elidan et al., 2006), for the query conditional $P(Q \mid \boldsymbol{E})$ in the presence of a large number of nuisance variables. In belief propagation, a belief (current approximation of a single-variable marginal) is maintained for every variable $x_i \in X$, and beliefs are updated iteratively depending on neighboring beliefs, according to local update rules. We analyze theoretically the impact that any given nuisance variable has on the query beliefs, and propose a belief update schedule that prioritizes updates according to their estimated eventual impact on the query, leading to faster convergence. More specifically, we claim the following contributions:

1. A principled measure of importance of belief for an arbitrary variable x_i with respect to a given query, which can be computed efficiently and characterizes the magnitude of eventual change in query beliefs per unit change of the belief for x_i.

2. A general framework of importance-weighted residual belief propagation that allows one to significantly speed up convergence for query variables by focusing computation on the more important beliefs. Unlike the previous approaches, ours does not involve any simplification of the original graphical model, and thus does not change the eventual approximation result for $P(Q \mid \boldsymbol{E})$.

3. An empirical evaluation demonstrating significant speedups for our approach compared to a state of the art existing variant of belief propagation on real-life large-scale models.

Chapter 4 is an extended version of (Chechetka and Guestrin, 2010a).

The scope of this thesis

This thesis does not attempt to propose a single unified approach to using probabilistic graphical models that would be optimal for every setting. The variety of possible applications, with the corresponding requirements and available resources, make it extremely difficult, if not impossible, to come up with such a framework. Indeed, we present here both approaches that deliberately construct low-treewidth models, aiming to exploit exact parameter learning and inference, and a family of approaches designed to speed up inference in high-treewidth models. We argue that for each of the approaches, there are circumstances where it makes sense to apply that particular algorithm.

Rather than aiming for a unifying framework, we attempt here to shed light on some of the inherent tradeoffs of graphical models, which are often forgotten, be it because of the established habits in a sub-community, the wide applicability of existing generic approaches, the desire to have an intuitive interpretation of the model, or the lack of suitable alternatives. Our results demonstrate that if one is conscious of (a) those tradeoffs, such as inference error versus representation power of a model, and (b) of the end goals of applying the model, then one can design approaches that significantly improve performance over the generic state of the art algorithms.

Part I

Low-Treewidth Graphical Models

Chapter 2

Learning Generative Low-Treewidth Graphical Models with Quality Guarantees

In this chapter, we address a fundamental problem of learning the structure of tractable (low-treewidth) probabilistic graphical models that accurately approximate the distribution $P(X)$. In this problem of *generative* structure learning, it is assumed that the evidence and query sets E and Q are not known during learning. The lack of commitment to any given query-evidence split provides an important advantage: from the definition of conditional probability (1.1), the same low-treewidth generative model can be used to approximate a conditional distribution $P(Q \mid E)$ *for an arbitrary split of* X *into* Q *and* E at test time without re-learning.

The ability to decide on the particular way to split the variables into query and evidence is important in many applications. For example, in **sensor networks**, which can be used to monitor environmental parameters for the purposes of climate research or air conditioning and lighting control in a smart building (Mainwaring et al., 2002; Pottie and Kaiser, 2000), nodes often fail in an unpredictable manner and communication errors lead to some of the sensor measurements being lost. To maintain an accurate estimate of the state of the complete deployment area, one needs to recover the most likely values of the missing measurements (query) conditioned on the measurements of the remaining nodes (evidence), which is makes it a natural domain for applying probabilistic graphical models (Paskin and Guestrin, 2004). Because the failures of the sensor nodes are impossible to predict in advance, it is important for the model to be able to accommodate any possible query-evidence split at test time.

In **system diagnosis** applications, such as troubleshooting printer failures (Breese and Heckerman, 1996), even though the set of variables that can *in principle* be measured directly is known in advance, the measurements are costly in terms of time, user workload, or other resources. Therefore, a model of the system needs to be able to handle an arbitrary set of evidence E and both infer the likely state of the unknown variables, and, based on the known variables, to recommend the most useful further measurements (Krause and Guestrin, 2005). Similarly to the sensor network setting, here it is infeasible to learn a separate model for every possible combination of known and unknown variable sets, and a single accurate model is needed.

Generative model learning is not always the optimal approach. When the evidence set E is known in

advance, one can exploit this extra knowledge to learn a *discriminative* model that directly approximates the conditional distribution $P(Q \mid E)$ an is typically more accurate (c.f. chapter 3) of this thesis. However, in chapter 3 we also show that a generative structure learning approach can be used as a building block for discriminative learning. To summarize, an approach for learning high-quality low-treewidth models in the generative setting is useful both directly in applications and as a basis for discriminative models.

Before proceeding to the algorithmic aspects of our structure learning approach, we review a PGM formalism, namely junction trees, that is especially suited for representing low-treewidth models. This special formalism is needed, because for general graphs the treewidth is intractable to even compute exactly (Arnborg et al., 1987). Junction trees serve as a more restricted fundamental data structure of our approach. Next, we describe the main theoretical result (Lemma 16) that underlies the quality guarantees of our approach, and the structure learning algorithm itself (Alg. 2.6) and its quality guarantees. Ours is the first algorithm to be able to probably approximately correctly (PAC, Valiant, 1984) learn a subset of low-treewidth distributions. Next, we describe the heuristics and efficiency improvements that are crucial for making the baseline algorithm practical. Finally, we show empirically on real-life datasets that our algorithm is able to learn models of better or equal quality than other low-treewidth approaches and competitive with the high treewidth methods.

2.1 Junction trees: bounded treewidth graphical models

In this section, we review junction trees (for details, see Cowell et al. 2003) and discuss the quality of approximation of distributions by junction trees. We review two alternative definitions of junction trees: the more commonly used one of Jensen and Jensen (1994), and an alternative based on the definition of Almond and Kong (1991). The representative power of the two definitions is the same, and mutual conversion is simple. We argue that the latter is better suited for structure learning, because it is less prone to the problem of having two different structures that represent the same family of distributions and thus, from the perspective of distribution representation, are indistinguishable. In particular, for *maximal* Almond-Kong junction trees, no two structures represent the same family of distributions.

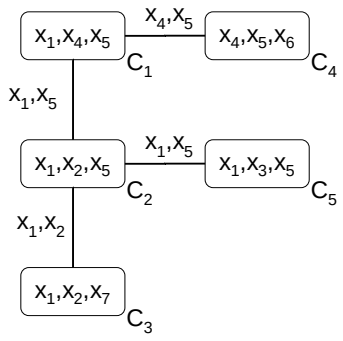
2.1.1 Junction trees of Jensen and Jensen

Let $\mathbb{C} = \{C_1, \ldots, C_m\}$ be a collection of subsets of X. Elements of $\mathbb{C}$ are called **cliques**. Let $\mathbb{T}$ be a set of edges connecting pairs of cliques such that $(\mathbb{T}, \mathbb{C})$ is a tree.

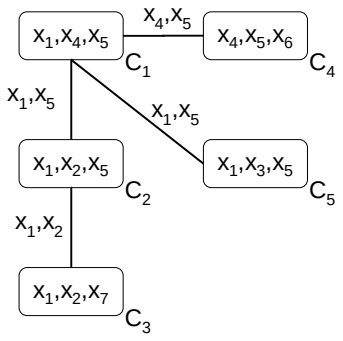
Definition 3. Tree $(\mathbb{T}, \mathbb{C})$ is a Jensen and Jensen **junction tree** if and only if it satisfies the **running intersection property (RIP)**: $\forall C_i, C_j \in \mathbb{C}$ and $\forall C_k$ on the (unique) simple path between C_i and C_j, it holds that $(C_i \cap C_j) \subseteq C_k$.

For an edge $(C_i - C_j) \in \mathbb{T}$, the set $S_{ij} \equiv C_i \cap C_j$ is called the **separator.** The size of a largest clique in a junction tree minus one is called the **treewidth** of that tree. For example, in a junction tree in Fig. 2.1a, variable x_1 is contained in both cliques C_3 and C_5, so it has to be contained in clique C_2, because C_2 is on the unique simple path between C_3 and C_5. The largest clique in Fig. 2.1a has size 3, so the treewidth of that junction tree is 2.

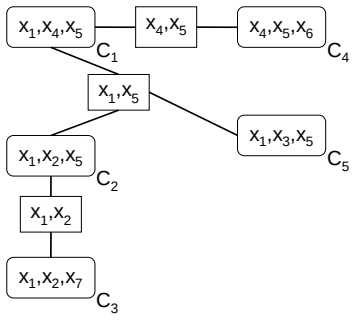
A distribution $P(X)$ is **representable** using junction tree $(\mathbb{T}, \mathbb{C})$ if for every separator S_{ij}, instantiating all variables of S_{ij} renders the variables on different sides of S_{ij} independent. Denote the fact that A is independent of B given S by $(A \perp B \mid S)$. Let $\mathbb{C}_{C_j \to C_i}$ be cliques that can be reached from



(a) Example of a Jensen and Jensen junction tree. Rounded rectangles are cliques, separators are marked on the edges.



(b) A Jensen and Jensen junction tree inducing the same factorization as that in Fig. 2.1a.



(c) An Almond-Kong junction tree inducing the same factorization as junction trees in Fig. 2.1a and 2.1b. Rounded rectangles are cliques, regular rectangles are separators.

Figure 2.1: Examples of junction trees.

C_i in $(\mathbb{T}, \mathbb{C})$ without using edge $(C_i - C_j)$. In particular, $C_i \in \mathbb{C}_{C_j \to C_i}$ and $C_j \notin \mathbb{C}_{C_j \to C_i}$. Denote $X_{C_j \to C_i}$ to be the set of variables that are covered by $\mathbb{C}_{C_j \to C_i}$, but are not in the separator S_{ij} : $X_{C_j \to C_i} \equiv \left(\bigcup_{C \in \mathbb{C}_{C_j \to C_i}} C \right) \setminus S_{ij}$.

For example, in Fig. 2.1a, $S_{12} = \{x_1, x_5\}$, $X_{2 \to 1} = \{x_4, x_6\}$, $X_{1 \to 2} = \{x_2, x_3, x_7\}$.
Definition 4. $P(X)$ **factors according to junction tree** $(\mathbb{T}, \mathbb{C})$ if and only if for every edge $(C_i - C_j) \in \mathbb{T}$, it holds that $\left(X_{C_j \to C_i} \perp X_{C_i \to C_j} \mid S_{ij} \right)$.

Let us define a **projection** $P_{(\mathbb{T},\mathbb{C})}$ of an arbitrary distribution $P(X)$ on a junction tree $(\mathbb{T}, \mathbb{C})$ as

$$P_{(\mathbb{T},\mathbb{C})}(X) = \frac{\prod_{C \in \mathbb{C}} P(C)}{\prod_{(C_i - C_j) \in \mathbb{T}} P(S_{ij})}. \tag{2.1}$$

If $P(X)$ factors according to $(\mathbb{T}, \mathbb{C})$, the projection $P_{(\mathbb{T},\mathbb{C})}$ is equal to P itself. The projection expression (2.1) is the key element of the representation of probability distributions using junction trees: if two junction trees have the same projection expression, then a distribution $P(X)$ will factor according to one of them if and only if it factors according to the other. Thus there would be no reason to prefer one of those two structures to the other, a troubling ambiguity for structure learning. Unfortunately, for junction trees of Jensen and Jensen this ambiguity is quite common: for every junction tree $(\mathbb{T}, \mathbb{C})$ that has at least two edges associated with the same separator, there exists at least one other junction tree that has the same projection expression as $(\mathbb{T}, \mathbb{C})$. For example, in Fig. 2.1a edges $C_1 - C_2$ and $C_2 - C_5$ are associated with the same separator: $\{x_1, x_5\}$. The alternative junction tree in Fig. 2.1b has the same projection as that in Fig. 2.1a, but a different graphical structure. To remedy such ambiguities, although not completely, we turn to an alternative definition of junction trees given by Almond and Kong (1991).

2.1.2 Almond-Kong junction trees

The main difference between the definition of Jensen and Jensen and that of Almond and Kong is that in the latter the separators are first-class objects. Let $\mathbb{C} = \{C_1, \ldots, C_m\}, \mathbb{S} = \{S_1, \ldots, S_s\}$ be collections of subsets of X. As before, C_i are called cliques, S_j – separators. Let $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ be a tree with edges T and cliques and separators as nodes. Following Almond and Kong (1991), define

Definition 5. Tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an Almond-Kong junction tree with unique nodes (**AKU junction tree**) iff

1. Every edge in $\mathbb{T}$ connects a clique $C \in \mathbb{C}$ to a separator $S \in \mathbb{S}$ such that $S \subset C$.
2. The running intersection property is satisfied: $\forall C_i, C_j \in \mathbb{C}$ and $\forall C_k, S_m$ on the (unique) simple path between C_i and C_j, it holds that $(C_i \cap C_j) \subseteq C_k$ and $(C_i \cap C_j) \subseteq S_m$.
3. No separator $S \in \mathbb{S}$ is a leaf in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$.
4. Every node of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ corresponds to a unique subset[1] of X.

Analogously to Def. 4, we can define what it means for a distribution $P(X)$ to factor according to an AKU junction tree:

Definition 6. $P(X)$ **factors according to an AKU junction tree** $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ iff for every edge $(C-S) \in \mathbb{T}$ connecting a clique $C \in \mathbb{C}$ and separator $S \in \mathbb{S}$ it holds that[2] $(X_{C\to S} \perp X_{S\to C} \mid S)$.

For a separator S from a junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, denote d_S to be the degree of node S (that is, the number of cliques that are neighbors of S in the graph). Similarly to the case of Jensen and Jensen junction trees, we can define the **projection** $P_{(\mathbb{T},\mathbb{C},\mathbb{S})}$ of any distribution $P(X)$ on $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ as

$$P_{(\mathbb{T},\mathbb{C},\mathbb{S})}(X) = \frac{\prod_{C\in\mathbb{C}} P(C)}{\prod_{S\in\mathbb{S}} P(S)^{d_S-1}}. \tag{2.2}$$

Lemma 7. *A distribution $P(X)$ factors according to $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ if and only if $P_{(\mathbb{T},\mathbb{C},\mathbb{S})}(X) = P(X)$.*

(The proofs of all the lemmas, propositions and theorems of this chapter are given in the Appendix A.2.)

Lemma 8. *Let $\mathcal{P}((\mathbb{T}, \mathbb{C}, \mathbb{S}))$ to be the set of all distributions that factorize according to $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Then for any $P(X)$, the projection $P_{(\mathbb{T},\mathbb{C},\mathbb{S})}(X)$ minimizes the KL divergence $KL(P||P')$ for $P' \in \mathcal{P}((\mathbb{T}, \mathbb{C}, \mathbb{S}))$:*

$$KL(P||P_{(\mathbb{T},\mathbb{C},\mathbb{S})}) = \min_{P'\in\mathcal{P}((\mathbb{T},\mathbb{C},\mathbb{S}))} KL(P||P')$$

We see that the properties of the two different formulations of junction trees are very similar, and indeed the two definition are equivalent in the following sense:

Lemma 9. *Whenever a Jensen and Jensen junction tree $(\mathbb{T}, \mathbb{C})$ exists, a AKU junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ with the same treewidth as $(\mathbb{T}, \mathbb{C})$ and the same projection expression exists, and vice-versa.*

[1] Although Almond and Kong (1991) did not have this uniqueness requirement, they have shown that for every junction tree satisfying parts 1–3 of Def. 5, there exists an AKU junction tree with the same projection expression. Therefore, the uniqueness requirement does not restrict expressive power.

[2] Notice that the independencies in Def. 6 are of the form "if we remove the variables of S from the graph, then the variables of every resulting connected component $X_{S\to C}$ are conditionally independent of *all other variables* given S" as opposed to a weaker statement "all connected components are *pairwise* conditionally independent". For example, in Fig. 2.1c it holds that $(x_4, x_6 \perp x_2, x_3, x_7 \mid x_1, x_5)$, not just $(x_4, x_6 \perp x_3 \mid x_1, x_5)$ and $(x_4, x_6 \perp x_2, x_7 \mid x_1, x_5)$.

In general, pairwise independence does not imply joint independence: for example, consider two independent uniformly distributed binary variables $\{x_1, x_2\}$ and $x_3 \equiv XOR(x_1, x_2)$. Then every two variables of the set $\{x_1, x_2, x_3\}$ are independent, but $(x_1 \not\perp x_2 x_3)$.

The drawback of Jensen and Jensen junction trees is that multiple trees may correspond to the same projection. Although different AKU junction trees, in general, may also have the same projection, we can avoid the ambiguity by restricting ourselves to the subclass of *maximal* junction trees:
Definition 10. A junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ of treewidth k is called **maximal** iff every clique $C \in \mathbb{C}$ has size $|C| = k + 1$, and every separator $S \in \mathbb{S}$ has size $|S| = k$.

Every maximal junction tree induces a unique projection:
Theorem 11. *If two maximal AKU junction trees* $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1)$ *and* $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$ *of the same treewidth and over the same variables* X *are different, then there exists a distribution* $P(X)$ *that factors according to* $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1)$*, but not to* $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$.

In fact, one can show that if two maximal AKU junction trees are different, then *almost all* (in the measure-theoretic sense) distributions that factor according to one of the junction trees do not factor according to the other. This uniqueness property will be useful in the theoretical guarantees for our structure learning algorithms. Intuitively, we will estimate a set of properties of the distribution in question, and the uniqueness property of Thm. 11 helps prove that those properties match a single structure, so there is no danger of trying to construct a junction tree whose properties are a mix of properties of two different structures.

2.1.3 Approximating distributions with junction trees

In practice, conditional independence is often too strong a notion: in real-life data the variables are rarely exactly conditionally independent, especially when the conditioning set is small. However, often the variables are *almost* conditionally independent (in other words, only weakly conditionally correlated) and the probability distribution can be well approximated by a graphical model. It is desirable then to extend the applicability and analysis of structure learning algorithms to such cases. A natural relaxation of the notion of conditional independence is to require sets of variables to have low **conditional mutual information** $I(\cdot, \cdot \mid \cdot)$. Denote by $H(A)$ the entropy of A and $H(A|S) \equiv H(AS) - H(S)$ the conditional entropy of A given S. Then by definition

$$I(A, B \mid S) \equiv H(A \mid S) - H(A \mid BS) \geq 0. \tag{2.3}$$

Conditional mutual information is always nonnegative, and zero if an only if $(A \perp B \mid S)$. Intuitively, $I(A, B \mid S)$ shows how much new information about A can one extract from B if S is already known. Using the low mutual information requirement instead of conditional independence, it is straightforward to relax the definition of a distribution factoring according to a JT:
Definition 12. $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an **AKU ε-junction tree** for $P(X)$ iff for every edge $(C - S) \in \mathbb{T}$ connecting a clique C and separator S it holds that $I(X_{C \to S}, X_{S \to C} \mid S) \leq \varepsilon$.
Definition 13. If there exists an AKU ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ of treewidth at most k for $P(X)$, we will say that P is **k-JT ε-representable**.

One can guarantee, in terms of Kullback-Leibler divergence, the quality of approximation of P by a projection of P on its ε-junction tree:
Lemma 14. *If* $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ *is an AKU* ε*-junction tree for* $P(X)$*, then* $KL(P, P_{(\mathbb{T}, \mathbb{C}, \mathbb{S})}) \leq n\varepsilon$.

The bound of Lemma 14 means that if we have an AKU ε-junction tree for $P(X)$, then instead of performing inference on P, which is intractable in general, we can use $P_{(\mathbb{T}, \mathbb{C}, \mathbb{S})}$ that both approximates P well and admits tractable exact inference.

Algorithm 2.1: Naïve approach to structure learning

Input: X, conditional mutual information oracle $I(\cdot, \cdot \mid \cdot)$, treewidth k, threshold δ

$\mathbb{L} \leftarrow \emptyset$ *// $\mathbb{L}$ is a set of "useful components"*

for *every* $S \subset X$ **s.t.** $|S| = k$ **do**

  for *every* $Q \subset X_{\setminus S}$ **do**

    if $I(Q, X_{\setminus QS} \mid S) \leq \delta$ **then**

      add (S, Q) to $\mathbb{L}$

return *FindConsistentTree($\mathbb{L}$)*

In the remainder of the chapter, we will only consider AKU junction trees and drop the AKU prefix for brevity.

2.2 Structure learning

In this chapter, we address the following problem: given data, such as multiple temperature readings from sensors in a sensor network, we treat each datapoint as an instantiation $\boldsymbol{X}$ of the random variables X and seek to find a good tractable approximation of $P(X)$. Specifically, we aim to find a $\hat{\varepsilon}$-junction tree of treewidth k for P with $\hat{\varepsilon}$ as small as possible. Note that the maximal treewidth k is considered to be a constant and not a part of problem input. The complexity of our approach is exponential in k. In practice, $k \ll n$.

The majority of existing approaches to structure learning belong to one of the two broad categories.[3] **Score-based** methods (for example, Teyssier and Koller, 2005; Choi et al., 2005; Singh and Moore, 2005) assign a measure of quality to every structure and try to find the structure with maximal quality. Usually the quality measure is some form of regularized likelihood. Exact maximization of the score is NP-complete, so, as a rule, these algorithms use local search over the structures and can only find a local optimum. **Constraint-based** algorithms (for example, Spirtes et al., 2001; Narasimhan and Bilmes, 2004) use hypothesis testing to enumerate the (approximate) conditional independencies of the underlying distribution. Constraint-based approaches then try to find a structure consistent with those independencies. Because the data is finite, the independencies recovered by hypothesis testing are usually different from the true ones. Thus most of the constraint-based approaches only have quality guarantees in the limit of infinite data. Also, most constraint-based algorithms need to test whether $(A \perp B \mid S)$ is true for sets A, B, S such that $|A| + |B| + |S| = \Theta(n)$. The tests used to decide conditional independence with fixed accuracy have complexity exponential in $|A| + |B| + |S|$, so most constraint-based approaches have complexity that is exponential in n. In contrast, our algorithm, which is also constraint-based, has polynomial in n complexity, as well as quality guarantees for the finite data case.

2.2.1 Constraint-based structure learning

Consider Alg. 2.1, a general naïve constraint-based approach to learning junction trees of treewidth k. For simplicity, let us start with a strict and unrealistic assumption: assume to have an oracle $I(\cdot, \cdot \mid \cdot)$ that can compute the mutual information $I(A, B \mid S)$ exactly for any disjoint subsets $A, B, S \subset X$. In Section 2.2.6, we will replace this unrealistic oracle with estimation of mutual information from data.

Because the value of ε for which an ε-junction tree for a given distribution $P(X)$ exists is in general not known, Alg. 2.1 takes a parameter $\delta \geq 0$ and aims to find a δ-junction tree. Recall that for a δ-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, for every separator $S \in \mathbb{S}$ and any clique $C \in \mathbb{C}$ that is directly connected to S it holds that

$$I(X_{C \to S}, X_{S \to C} \mid S) \leq \delta. \tag{2.4}$$

Using the oracle I and observing that $X_{C \to S}$, $X_{S \to C}$ and S are mutually disjoint and cover all of X, one could exhaustively evaluate for every possible pair of sets (S, Q) whether the necessary requirement (2.4) holds, that is whether (S, Q) can play a role of a pair $(S, X_{S \to C})$ in some δ-junction tree (c.f. lines 3–4 of Alg. 2.1). Since we are only concerned with maximal junction trees of treewidth k, the size of S has to be $|S| = k$ (line 2). Every pair (S, Q) such that[4] $I(Q, X_{\text{-}QS} \mid S) \leq \delta$ would be recorded into a list $\mathbb{L}$ (line 5). Then every δ-junction tree for $P(X)$ will be consistent with $\mathbb{L}$ in the following sense:

Definition 15. A junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is **consistent with** a **component list** $\mathbb{L}$ if and only if for every separator $S \in \mathbb{S}$ and clique $C \in \mathbb{C}$ such that S and C are connected: $(S - C) \in \mathbb{T}$, it holds that $(S, X_{S \to C}) \in \mathbb{L}$.

Because $\mathbb{L}$ consists of pairs (S, Q) such that $I(Q, X_{\text{-}QS} \mid S) \leq \delta$, any junction tree consistent with $\mathbb{L}$ is by definition an δ-junction tree for $P(X)$. Therefore, it is sufficient to set $\delta = \varepsilon$ and find a junction tree consistent with the resulting $\mathbb{L}$, which will then be an ε-junction tree for P.

Let us denote *FindConsistentTree* a procedure that takes $\mathbb{L}$ and outputs a junction tree consistent with $\mathbb{L}$ (or a failure to find one). Such a procedure could be implemented, for example, using constraint satisfaction. We provide a concrete form of *FindConsistentTree* in Section 2.2.4.

Unfortunately, using Alg. 2.1 directly is impractical because its complexity is exponential in the total number of variables n. To make the approach outlined in Alg. 2.1 tractable, one needs to address the following problems:

1. For every candidate separator S, there are 2^{n-k} possible subsets $Q \subseteq X_{\text{-}S}$. Thus, for every candidate separator S, Alg. 2.1 requires $O(2^n)$ mutual information computations (lines 3–4). (Addressed in Section 2.2.3).

2. Every call to the mutual information oracle on line 4 of Alg. 2.1 involves n variables. In general, the best known way to compute mutual information with fixed accuracy takes time exponential in the number of variables. (Addressed in Section 2.2.2).

3. A concrete efficient form of *FindConsistentTree* is needed. (Addressed in Section 2.2.4).

In the rest of this section, we address all of the above problems. Then, in Section 2.2.6, we replace the exact $I(\cdot, \cdot \mid \cdot)$ oracle with estimation of mutual information from data. Together, our solutions form a polynomial-time structure learning algorithm with quality guarantees.

[3] This paragraph contains the minimal information necessary to put our work in context. We defer the detailed discussion of the related work (Section 2.5) until after the presentation of our approach.

[4] Notation note: for any sets A, B, C we will denote $A \setminus (B \cup C)$ as $A_{\text{-}BC}$ to lighten the notation.

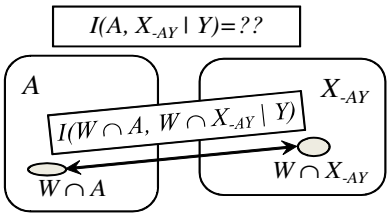


Figure 2.2: Illustration to the statement of Lemma 16. To upper-bound the conditional mutual information between *arbitrarily large* sets A and $X_{\text{-}AY}$ given Y, one only needs to compute the mutual information between the *fixed-sized* subsets such as $W \cap A$ and $W \cap X_{\text{-}AY}$.

2.2.2 Global independence assertions from local tests

Testing for (approximate) conditional independence, which can be cast as estimating conditional mutual information from data, is a crucial component of most constraint-based structure learning algorithms (Narasimhan and Bilmes, 2004; Spirtes et al., 2001). Similarly to Alg. 2.1, constraint-based approaches often need estimates of form $I(A, B \mid S)$ for sets A, B and S of total size $\Theta(n)$. Unfortunately, the best known way of computing mutual information, as well as estimating I from data with fixed accuracy, has time and sample complexity exponential in $|A| + |B| + |S|$. Previous work has not addressed this problem. In particular, the approach of Narasimhan and Bilmes (2004) has exponential complexity, in general, because it needs to estimate I for sets of size $\Theta(n)$.

Fortunately, our first new result (Lemma 16) shows that it is possible to *upper bound* the conditional mutual information of two *arbitrarily large* sets $A, X_{\text{-}AY}$ (see Fig. 2.2 for an illustration) given a small set Y, by only computing conditional mutual information for *fixed-sized* subsets of the large sets in question. For example, in Fig. 2.2 one wold only need to compute mutual information values of form $I(W \cap A, W \cap X_{\text{-}AY} \mid Y)$ for small sets W instead of computing $I(A, X_{\text{-}AY} \mid Y)$ directly. The fact that we only need to look at fixed-sized subsets reduces the complexity from *exponential* in $|X|$ to *polynomial.* The following lemma, stating the formal result, is not only the foundation of our structure learning approach, but is also applicable more broadly, to any setting where an upper bound on mutual information is needed:

Lemma 16. *Let $P(X)$ be a k-JT ε-representable distribution. Let $Y \subset X$, $A \subset X_{\text{-}Y}$. If*

$$\forall W \subseteq X_{\text{-}Y} \text{ s.t. } |W| \leq k+1, \text{ it holds that } I(A \cap W, X_{\text{-}AY} \cap W \mid Y) \leq \delta,$$

then

$$I(A, X_{\text{-}AY} \mid Y) \leq n(\varepsilon + \delta).$$

The tightness of the upper bound on the conditional mutual information depends, through the value of ε for which $P(X)$ is k-JT ε-representable, on how well the distribution P can be approximated by a junction tree of treewidth k. However, it is important to notice that applying Lemma 16 does not require knowledge of any ε-JT for $P(X)$, and no relationship of the conditioning set Y and elements of an (unknown) ε-JT for $P(X)$ is needed. Moreover, the size of the set Y on which one conditions does not depend on the treewidth k of the true ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ for $P(X)$. Although in the rest of the chapter we will apply Lemma 16 to sets Y of size k in an attempt to recover the "true" structure, one just as well use this upper bound for smaller or larger candidate separators Y in alternative approaches to structure learning.

We can use Lemma 16 to bound $I(A, X_{\text{-}AY} \mid Y)$ from above using $O\binom{n}{k+1} = O(n^{k+1})$ calls to the mutual information oracle. Each call will involve at most $|Y| + k + 1$ variables. For discrete variables with

cardinality r, the total complexity is $O\left(n^{k+1}r^{|Y|+k+1}\right)$. In Alg. 2.1, candidate separators S play the role of Y, so $|Y| = k$. Replacing the exact computation of $I\left(A, X_{\text{-}AS} \mid S\right)$ with the upper bound of Lemma 16, we reduce the complexity of every call on line 4 from $O\left(r^n\right)$, exponential in n, to $O\left(n^{k+1}r^{2k+1}\right)$, polynomial in n.

One can use Lemma 16 not only to discover weak conditional dependencies, but also to bound the quality of approximation of $P(X)$ by a projection on any junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$:

Corollary 17. *If for every separator S and clique C of a junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ such that $(S - C) \in \mathbb{T}$, the conditions of Lemma 16 hold with $Y = S$ and $A = X_{S \to C}$, then $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a $n(\varepsilon + \delta)$-junction tree for $P(X)$.*

2.2.3 Partitioning algorithm for weak conditional independencies

Now that we have an efficient upper bound for conditional mutual information, let us turn to reducing the number of calls to this bound in Alg. 2.1 from exponential (2^{n-k} for every candidate separator S) to polynomial. For every S, Alg. 2.1 finds all subsets $Q \subset X_{\text{-}S}$ such that

$$I\left(Q, X_{\text{-}QS} \mid S\right) \leq \varepsilon. \tag{2.5}$$

For every S there may be, in general, exponentially many such subsets. For example, if every variable $x \in X_{\text{-}S}$ is conditionally independent of all other variables, that is, $(x \perp X_{\text{-}Sx} \mid S)$, then every $Q \subset X_{\text{-}S}$ satisfies property (2.5). The fact that even enumerating all possible "good" subsets may require exponential time suggests that it is necessary to relax the requirements of the problem to get a tractable algorithm. We will use the following relaxation: instead of looking for *all* $Q \subset X_{\text{-}S}$ satisfying (2.5), look for *one partitioning* $\mathbb{Q}_S$ of $X_{\text{-}S}$ such that

- Elements of $\mathbb{Q}_S$ do not intersect: $\forall Q_i, Q_j \in \mathbb{Q}_S$ s.t. $i \neq j:\ Q_i \cap Q_j = \emptyset$.
- $\mathbb{Q}_S$ covers all of $X_{\text{-}S}$: $\cup_{Q \in \mathbb{Q}_S} Q = X_{\text{-}S}$.
- $\forall Q \in \mathbb{Q}_S$, it holds that $I\left(Q, X_{\text{-}QS} \mid S\right) \leq \hat{\varepsilon}$, where $\hat{\varepsilon}$ is some function of ε (the concrete form will be provided shortly).

Narasimhan and Bilmes (2004) considered the same relaxation and presented a solution that relies on the existence of an efficient *approximation* of oracle $I\left(\cdot, \cdot \mid \cdot\right)$ (as opposed to an efficient *upper bound* that we provide). Their key observation was that the function $F_S(Q) \equiv I\left(Q, X_{\text{-}QS} \mid S\right)$ is **symmetric submodular**: $F_S(A) = F_S(X_{\text{-}SA})$ and $F_S(A) + F_S(B) \geq F_S(A \cup B) + F_S(A \cap B)$. A symmetric submodular function of n arguments can be minimized using an algorithm by Queyranne (1998) at a cost of $O(n^3)$ function evaluations. Starting with all variables $X_{\text{-}S}$ being in the same partition ($\mathbb{Q}_S = \{X_{\text{-}S}\}$), Narasimhan and Bilmes (2004) combine Queyranne's algorithm with divide-and-conquer approach to iteratively refine $\mathbb{Q}_S$. Unfortunately, their algorithm requires evaluations of mutual information for sets of size $\Theta(n)$. The best known way to estimate mutual information with fixed accuracy has complexity exponential in n (for example, see Höffgen, 1993). Therefore, algorithm of Narasimhan and Bilmes (2004), in general, has complexity exponential in n.

"Low-Treewidth Conditional Independencies" partitioning algorithm

Our approach (Alg. 2.2), called LTCI for "Low-Treewidth Conditional Independencies", in contrast to that of Narasimhan and Bilmes (2004), has polynomial complexity, provided that the following two quantities

Algorithm 2.2: LTCI: find Conditional Independencies in Low-Treewidth distributions

Input: X, candidate separator S, oracle $I(\cdot,\cdot \mid \cdot)$, threshold δ, max set size q

1 $\mathbb{Q}_S \leftarrow \cup_{x \in X_{\text{-}S}} \{x\}$ *// $\mathbb{Q}_S$ is a set of singletons*

// In the loop below, choose sets W in the order of increasing size

2 **for** $W \subset X_{\text{-}S}$ **s.t.** $|W| \leq q$ **AND** $\nexists Q \in \mathbb{Q}_S$ *s.t.* $W \subseteq Q$ **do**

3 **if** $\min_{U \subset W} I(U, W_{\text{-}U} \mid S) > \delta$ *// Find min with Queyranne's algorithm*

4 **then**

5 merge all $Q_i \in \mathbb{Q}_S$, s.t. $Q_i \cap W \neq \emptyset$

6 **return** $\mathbb{Q}_S$

are $O(1)$: size of separator S in question and the maximum size q of subsets of $X_{\text{-}S}$ that LTCI considers. In our structure learning algorithm, which uses Alg. 2.2 as a subroutine, these requirements hold: $|S| = k$ and $q = k + 2$.

To gain intuition for LTCI, suppose there exists a ε-junction tree for $P(X)$, such that S is a separator and cliques $C_1, \ldots, C_{d_S}$ are directly connected to S. Then the sets $X_{S \to C_1}$, …, $X_{S \to C_{d_S}}$ partition $X_{\text{-}S}$. For example, in Fig. 2.3a, $X_{\text{-}S}$ is partitioned by $X_{S \to C_1}$, $X_{S \to C_2}$, $X_{S \to C_3}$. Consider a set $W \subseteq X_{\text{-}S}$ such that W has variables from more than one set $X_{S \to C_i}$:

$$\exists i : W \cap X_{S \to C_i} \neq \emptyset \text{ and } W \cap X_{C_i \to S} \neq \emptyset.$$

By definition of an ε-junction tree, $I(X_{S \to C_i}, X_{C_i \to S} \mid S) \leq \varepsilon$. Therefore, by the monotonicity[5] of conditional mutual information, we have

$$I(W \cap X_{S \to C_i}, W \cap X_{C_i \to S} \mid S) \leq \varepsilon. \tag{2.6}$$

Notice that $X_{\text{-}S} = X_{S \to C_i} \cup X_{C_i \to S}$ and $W \subseteq X_{\text{-}S}$, so $\{W \cap X_{S \to C_i}, W \cap X_{C_i \to S}\}$ is a partitioning of W. Such partitioning is shown by a dashed line in Fig. 2.3a. It follows from (2.6) that

$$\exists U \subset W \text{ s.t. } I(U, W_{\text{-}U} | S) \leq \varepsilon, \text{ namely } U \equiv W \cap X_{S \to C_i}. \tag{2.7}$$

Note that Equation 2.7 is a *necessary property* of every W that includes variables from sets $X_{S \to C_i}$ for more than one clique C_i. Therefore a contrapositive of (2.7),

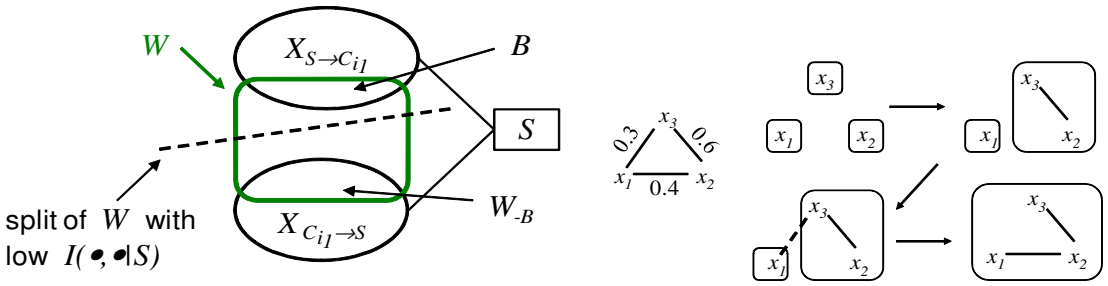
$$\forall U \subset W \text{ it holds that } I(U, W_{\text{-}U} | S) > \varepsilon, \tag{2.8}$$

is a *sufficient condition* for W to be within a set $X_{S \to C_i}$ for a single clique C_i.

To find the partitioning, Alg. 2.2 starts with every variable of $X_{\text{-}S}$ forming its own singleton partition (line 1). Alg. 2.2 then checks for every subset $W \subseteq X_{\text{-}S}$ of size at most q whether condition (2.8) holds (line 3). If yes, then all variables of W are assigned to the same partition. Being in the same partition is a transitive relationship, so all partitions that have variables in W are merged (line 4).

An example trace of LTCI is depicted in Fig. 2.3b (for simplicity, the separator S is not shown). The pairwise values of $I(\cdot,\cdot \mid S)$ are on the left. The threshold in this example is $\delta = 0.35$ and $q = 2$. First, LTCI checks the edge $x_2 - x_3$. The mutual information is above the threshold, so x_2 and x_3 are merged.

[5] **Monotonicity** of conditional mutual information: for every $P(X)$, every non-intersecting $A, B, S \subset X$ and every $C \subseteq A$, $D \subseteq B$ it holds that $I(C, D \mid S) \leq I(A, B \mid S)$.



(a) Intuition for LTCI. For every W that intersects $X_{S\to C_i}$ for more than one C_i, there exists a split (dashed line) into two weakly dependent subsets, for example, $B \equiv W \cap C_{i_1}$ and W_{-B}. Thus if no such split can be found, W must be a subset of a single $X_{S\to C_i}$.

(b) Example trace of LTCI. The values of pairwise conditional mutual information are on the left. $\delta = 0.35$.

Figure 2.3

Then $x_1 - x_3$ is checked, the dependency strength is not high enough, so nothing happens. Finally, LTCI checks $x_1 - x_2$, the mutual information is again above threshold, so $\{x_1\}$ and $\{x_2 x_3\}$ are merged to form a single connected component.

As Lemma 18 shows, the complexity of LTCI is exponential in k and q, but polynomial in n. Therefore, for small k and q, LTCI is a tractable algorithm.

Lemma 18. *The time complexity of LTCI with $|S| = k$ is $O\left(n^q\left(q^3 J^{MI}_{k+q} + n\right)\right)$, where J^{MI}_{k+q} is the time complexity of computing $I(A, B \mid S)$ for $|A| + |B| + |S| = k + q$.*

Partitioning quality guarantees

Suppose we call the partitioning algorithm for a separator $S \in \mathbb{S}$ of a true ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ for $P(X)$. It is natural to require that the partitioning algorithm returns partitions that are as close as possible to the true sets of variables $X_{S\to C}$ on the same side of separator S. In this section, we will show that even though LTCI, in general, may not find exactly the correct partitions, it finds a solution close to optimal. Let $\mathbb{Q}_S$ be the result of calling LTCI for separator $S \in \mathbb{S}$ of an ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Then two types of mistakes are possible in $\mathbb{Q}_S$ with respect to $(\mathbb{T}, \mathbb{C}, \mathbb{S})$:

1. There are variables x, y that are on the different sides of S in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$:

$$x \in X_{S\to C_i}, y \in X_{S\to C_j}, i \neq j,$$

but x and y are in the same partition in $\mathbb{Q}_S$. We will say that a partitioning algorithm is **correct** if and only if it never makes a mistake of this type. Formally, we have

Definition 19. A partitioning algorithm $\mathcal{A}$ is called **correct** if and only if for every distribution $P(X)$ with an ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, algorithm $\mathcal{A}$ called with $S \in \mathbb{S}$ and the value of threshold set to $\delta = \varepsilon$ will output a set $\mathbb{Q}_S$ such that

$$\forall Q \in \mathbb{Q}_S \text{ it holds that } \exists C \in \mathbb{C} \text{ s.t. } (S - C) \in \mathbb{T} \text{ and } Q \subseteq X_{S\to C}.$$

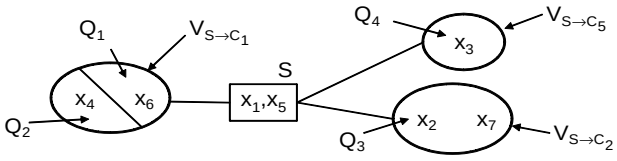


Figure 2.4: Example result of running a *correct* (Def. 19) partitioning algorithm for separator $S = \{x_1x_5\}$ for the junction tree in Fig. 2.1c.

A correct algorithm, called with with $\delta = \varepsilon$, is guaranteed to output a *refinement* of the "true" partitioning $\left\{X_{S\to C_1}, \ldots, X_{S\to C_{d_S}}\right\}$. For example (see Fig. 2.4), for separator $S = \{x_1x_5\}$ of the junction tree in Fig. 2.1c,

$$X_{S\to C_1} = x_4x_6,\ X_{S\to C_2} = x_2x_7, X_{S\to C_5} = x_3.$$

Assume that the junction tree in Fig. 2.1c is an ε-JT for $P(X)$. Then, for separator S, a possible partitioning result of a correct algorithm with $\delta = \varepsilon$ would be

$$\mathbb{Q}_{x_1x_5} = \{\{x_4\}, \{x_6\}, \{x_2x_7\}, \{x_3\}\}.$$

2. The other type of error is the opposite of type 1: there are variables x, y that are on the same side of S in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$: $x, y \in X_{S\to C}$, but the partitioning algorithm puts x and y in different partitions in $\mathbb{Q}_S$. Let us define the class of algorithms that guarantee a limited magnitude of such mistakes:
Definition 20. A partitioning algorithm $\mathcal{A}$ is called **α-weak** if and only if for every distribution $P(X)$ with an ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ of treewidth k, algorithm $\mathcal{A}$, when called with a candidate separator S of size k, maximum subset size $q \geq k+1$, and the value of threshold δ, will output a set $\mathbb{Q}_S$ such that

$$\forall Q \in \mathbb{Q}_S \text{ it holds that } I\left(Q, X_{\text{-}QS} \mid S\right) \leq \alpha(\varepsilon, \delta).$$

Intuitively, an α-weak algorithm will not mistakenly put variables in different partitions, provided that the dependence between those variables is strong enough.

It is desirable for a partitioning algorithm to be correct and α-weak for as small an α as possible, ideally $\alpha = \delta$. For $\delta = \varepsilon$ and a separator S from the true ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, by Def. 19, a correct algorithm would always separate variables that are on different sides of S in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. At the same time, provided that there is no way to split the variables of $X_{S\to C}$ into several weakly independent subsets (intuitively, this requirement means that the conditional independencies of $P(X)$ that are *not reflected* by $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ are not very significant), a δ-weak algorithm with $\delta = \varepsilon$ will not separate the variables on the same side of S. Therefore, a correct and δ-weak algorithm would recover the true graph-theoretic partitioning $\mathbb{Q}_S = \{X_{S\to C_1}, \ldots, X_{S\to C_{d_S}}\}$. LTCI, which we use instead of lines 3–5 in Alg. 2.1, satisfies the first requirement (correctness) and a relaxed version of the second (δ-weakness):
Lemma 21. *For k-JT ε-representable distributions, LTCI, for $q \geq k+1$, is correct and $n(\varepsilon + k\delta)$-weak.*

To summarize, we can use LTCI to *efficiently* find a partitioning $\mathbb{Q}_S$ for any candidate separator S. Moreover, if S is an actual separator of an ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, the partitioning $\mathbb{Q}_S$ is guaranteed to be *similar to the graph-theoretical partitioning* imposed by S on $(\mathbb{T}, \mathbb{C}, \mathbb{S})$.

2.2.4 Implementing *FindConsistentTree* using dynamic programming

After the exhaustive tests for low conditional mutual information on lines 3–5 of Alg. 2.1 are replaced with LTCI (Alg. 2.2), a concrete form of *FindConsistentTree* procedure is the only remaining step needed to make Alg. 2.1 practical. For this purpose, we adopt a dynamic programming approach of Arnborg et al. (1987). Narasimhan and Bilmes (2004) used the same dynamic programming algorithm to construct a junction tree, but did not address the additional complications that arise in the structure learning setting as opposed to the original problem that Arnborg et al. (1987) solved. This difference will be discussed in detail after we review the intuition for the dynamic programming approach.

Intuition

Consider a junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Let $S \in \mathbb{S}$ be a separator, and $C \in \mathbb{C}$ be a clique directly connected to S. Let $\mathbb{C}_{S\to C}, \mathbb{S}_{S\to C}$ be the set of cliques (including C itself) and separators reachable from C without using edge $(S - C)$, and $\mathbb{T}_{S\to C}$ the set of edges from $\mathbb{T}$ that connect cliques and separators from $\mathbb{C}_{S\to C}, \mathbb{S}_{S\to C}$. If $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an ε-junction tree for $P(X)$, then $(\mathbb{T}_{S\to C}, \mathbb{C}_{S\to C}, \mathbb{S}_{S\to C})$ is an ε-junction tree for $P(X_{S\to C} \cup S)$. Moreover, the subtree $(\mathbb{T}_{S\to C}, \mathbb{C}_{S\to C}, \mathbb{S}_{S\to C})$ consists of a clique C and several sub-subtrees that are each connected to C via some separator other than S.

For example, in Fig. 2.1a the subtree over cliques C_1, C_2, C_4, C_5 can be decomposed into clique C_2 and two sub-subtrees: one including cliques $\{C_1, C_4\}$ and one with clique C_5. This recursive structure suggests a dynamic programming approach: to check whether a component $(S, Q) \in \mathbb{L}$ can play a role of $(S, X_{S\to C})$ in some junction tree, check if smaller already known subtrees, corresponding to subcomponents (S', Q'), can be put together to form a larger subtree that covers exactly the variables of (S, Q). Formally, we require the following property defined in a recursive way:

Definition 22. (S, Q) is **$\mathbb{L}$-decomposable** if and only if $S \cap Q = \emptyset$ and either $|Q| = 1$ (base case) or there exist

1. $x \in Q$ ($C \equiv S \cup \{x\}$ would be the clique for which $(S, Q) = (S, X_{S\to C})$).
2. $\mathbb{D} = \cup_i \{(S_i, Q_i)\}$, $\mathbb{D} \subseteq \mathbb{L}$. $\mathbb{D}$ is the set of subcomponents with associated subtrees that together will form a subtree over (S, Q).

such that

1. $\forall (S_i, Q_i) \in \mathbb{D}$, it holds that (S_i, Q_i) is $\mathbb{L}$-decomposable: there is a subtree associated with each (S_i, Q_i).
2. $\bigcup_{(S_i,Q_i)\in\mathbb{D}} Q_i = Q \setminus \{x\}$: the subtrees cover *exactly* the variables of $(S \cup Q) \setminus C$.
3. $\forall (S_i, Q_i) \in \mathbb{D}$, it holds that $S_i \subset S \cup \{x\}$: each subcomponent can be connected directly to the clique $S \cup x$.
4. $\forall (S_i, Q_i), (S_j, Q_j) \in \mathbb{D}$, s.t. $i \neq j$, it holds that $Q_i \cap Q_j = \emptyset$: ensure the running intersection property within the subtree over $S \cup Q$.

The set $\mathbb{D}$ is called a **decomposition** of (S, Q).

Note that any component (S, Q) with $|Q| = 1$ is trivially decomposable with $\mathbb{D} = \emptyset$ and $x = Q$ – such components correspond to leaf cliques of a junction tree. The dynamic programming algorithm (Alg. 2.3)

Algorithm 2.3: FindConsistentTreeDP (adapted from Arnborg et al. 1987)

Input: List $\mathbb{L}$ of components (S, Q)

1 **for** $(S, Q) \in \mathbb{L}$ *in the order of increasing* $|Q|$ **do**
2 $\mathbb{D}(S, Q) = FindDecomposition((S, Q), \mathbb{L})$
3 **if** $\exists S$ *s.t. for every* $Q \in \mathbb{Q}_S$ *it holds that* $\mathbb{D}(S, Q) \neq \emptyset$ **OR** $|Q| = 1$ **then**
  // Recursively unroll the decompositions to get a junction tree
4 $\mathbb{T} = \emptyset, \mathbb{C} = \emptyset, \mathbb{S} = \{S\}$
5 **for** *every* $Q \in \mathbb{Q}_S$ **do**
6 $\{C', (\mathbb{T}', \mathbb{C}', \mathbb{S}')\} \leftarrow$ GetSubtree$((S, Q))$
7 $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbb{C}', \ \mathbb{S} \leftarrow \mathbb{S} \cup \mathbb{S}', \ \mathbb{T} \leftarrow \mathbb{T} \cup \mathbb{T}' \cup (S' - C')$
8 **return** $(\mathbb{T}, \mathbb{C}, \mathbb{S})$
9 **else return** failure

checks for all $(S, Q) \in \mathbb{L}$, in the order of increasing size of Q, whether (S, Q) is $\mathbb{L}$-decomposable and records the decomposition whenever it exists.

Suppose a separator S is found such that for every $Q \in \mathbb{Q}_S$ the pair (S, Q) is $\mathbb{L}$-decomposable. Then a junction tree consistent with $\mathbb{L}$ can be constructed by creating a separator node S and connecting the subtrees corresponding to every $(S, Q) \in \mathbb{L}$ to S, which is exactly what Alg. 2.3 does on lines 4-8. To guarantee the running intersection property in the result of Alg. 2.3, we need to make mild assumptions on the contents of $\mathbb{L}$ (these assumptions will hold for our method of constructing $\mathbb{L}$):

Lemma 23. *Suppose the set* $\mathbb{L}$ *is such that*

- *For every* $(S, Q) \in \mathbb{L}$ *it holds that* $|S| = k$.
- *For every pair* $(S, Q'), (S, Q'') \in \mathbb{L}$ *such that* $Q' \neq Q''$ *it holds that* $Q' \cap Q'' = \emptyset$.
- *For every candidate separator* S *it holds that* $\cup_Q$ *s.t.* $_{(S,Q)\in\mathbb{L}} Q = X_{\text{-}S}$.

Then Alg. 2.3 returns either a failure or an AKU junction tree of treewidth k *over* X *consistent with* $\mathbb{L}$.

Observe that if $\mathbb{Q}_S$ is the result of calling LTCI for candidate separator S, then the list

$$\mathbb{L} = \bigcup_{(S,Q) \text{ s.t. } |S|=k, Q\in\mathbb{Q}_S} \{(Q, S)\}$$

satisfies the conditions of Lemma 23. Indeed, given a separator S, one can verify that throughout the execution of Alg. 2.2 it holds that the elements of $\mathbb{Q}_S$ are mutually exclusive and cover all of $X_{\text{-}S}$. Because LTCI is only called for every separator S once, partitions Q corresponding to the same S remain mutually exclusive in $\mathbb{L}$.

Greedy construction of decompositions

Note that Alg. 2.3 depends on the ability to check whether (S, Q) is $\mathbb{L}$-decomposable (line 2). Unfortunately, this problem, in general, is provably hard. Suppose we fix the variable x in the notation of Def. 22. We can form a set $\mathbb{L}' \subseteq \mathbb{L}$ of all pairs (S', Q') that can possibly participate in a decomposition of (S, Q) with clique Sx by checking the necessary conditions:

Algorithm 2.4: GetSubtree

Input: Decomposable component (S, Q)

$x \leftarrow Q \setminus \cup_{(S',Q') \in \mathbb{D}(S,Q)} Q'$
$\mathbb{T} = \emptyset, \mathbb{C} = \{Sx\}, \mathbb{S} = \emptyset$
for *every* $(S', Q') \in \mathbb{D}(S, Q)$ **do**
    if $S' \notin \mathbb{S}$ **then**
        Add S' to $\mathbb{S}$, $(Sx - S')$ to $\mathbb{T}$
    $\{C', (\mathbb{T}', \mathbb{C}', \mathbb{S}')\} \leftarrow$ GetSubtree$((S', Q'))$
    $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbb{C}', \ \mathbb{S} \leftarrow \mathbb{S} \cup \mathbb{S}', \ \mathbb{T} \leftarrow \mathbb{T} \cup \mathbb{T}' \cup (S' - C')$
return $\{Sx, (\mathbb{T}, \mathbb{C}, \mathbb{S})\}$

Algorithm 2.5: FindDecompositionGreedy

Input: Pair (S, Q), list $\mathbb{L}$ of pairs (S', Q'), already known decompositions $\mathbb{D}(S', Q')$

$\mathbb{L}' \leftarrow \{(S', Q') \in \mathbb{L} \mid \mathbb{D}(S', Q')$ is already known$\}$
if $|Q| = 1$ **then**
    $\mathbb{D}(S, Q) = \emptyset$, mark (S, Q) as $\mathbb{L}$-decomposable
else
    for $x \in Q$ **do**
        $\mathbb{D}(S, Q) = \emptyset$
        for $(S', Q') \in \mathbb{L}'$ **s.t.** **do**
            if $S' \subset Sx$ **AND** $Q' \subseteq Q \setminus \left(x \cup \bigcup_{(S'',Q'') \in \mathbb{D}(S,Q)} Q'' \right)$ **then**
                add (S', Q') to $\mathbb{D}(S, Q)$
        if $\bigcup_{(S',Q') \in \mathbb{D}(S,Q)} Q' = Q \setminus x$ **then**
            mark (S, Q) as $\mathbb{L}$-decomposable
            $\mathbb{D}$ is a valid decomposition of (S, Q); **return** $\mathbb{D}(S, Q)$
did not find a decomposition for (S, Q) in $\mathbb{L}$; **return** failure

- (S', Q') is $\mathbb{L}$-decomposable.
- $S' \subset S \cup \{x\}$.
- $Q' \subseteq Q \setminus \{x\}$.

Denote $\mathbb{Q}' \equiv \cup_{(S',Q') \in \mathbb{L}'} \{Q'\}$. Then finding the decomposition of (S, Q) is equivalent to finding a subset $\mathbb{Q}'' \subseteq \mathbb{Q}'$ such that the elements of $\mathbb{Q}''$ do not intersect and cover all of $Q_{\text{-}x}$. The latter problem is known as *exact cover* and is, in general, NP-complete (Karp, 1972). However, by placing additional restrictions on the structure of $\mathbb{L}$, one can identify tractable subclasses of the problem. For example, Arnborg et al. (1987) provide a method to check for decomposability in a restricted setting: their approach requires that $\mathbb{Q}'$ is such that

$$\forall Q'_1, Q'_2 \in \mathbb{Q}' \text{ s.t. } Q'_1 \cap Q'_2 \neq \emptyset \text{ it holds that } Q'_1 \subseteq Q'_2 \text{ or } Q'_2 \subseteq Q'_1. \tag{2.9}$$

Algorithm 2.6: Efficient approach to structure learning

Input: X, oracle $I(\cdot, \cdot \mid \cdot)$, treewidth k, threshold δ, $\mathbb{L} = \emptyset$

for $S \subset X$ *s.t.* $|S| = k$ **do**
  for $Q \in$ *LTCI*$(X, S, I, \delta, k+2)$ **do**
    $\mathbb{L} \leftarrow \mathbb{L} \cup (S, Q)$

return *FindConsistentTreeDPGreedy*$(\mathbb{L})$

Provided that property (2.9) holds, whenever a suitable $\mathbb{Q}''$ exists, it can always be constructed by taking $\mathbb{Q}'$ and removing all sets $Q'_i \in \mathbb{Q}'$ that have a superset $Q'_j \in \mathbb{Q}', Q'_i \subseteq Q_j$. Because Arnborg et al. (1987) used a graph-theoretical separation oracle *with a fixed graph* to find components (S, Q), they were able to guarantee property (2.9).

Unfortunately, to use the separation oracle from Arnborg et al. (1987) one needs a partitioning oracle that for every candidate separator S returns partitioning components (S, Q) consistent with *the same one* graph shared across *all possible* S. Such an oracle would guarantee property (2.9). However, the separation oracle based on conditional mutual information, which we use, may return partitionings not consistent with any single graph for different separators and the property (2.9) does not necessarily hold. Therefore, we need a more general method to find decompositions than Arnborg et al. (1987). The previous work on learning limited-treewidth models has not addressed this problem. In particular, Narasimhan and Bilmes (2004), in a setting similar to ours, rely on the method of Arnborg et al. (1987), which is not guaranteed to work.

To keep complexity polynomial, we use a simple greedy approach (Alg. 2.5): given a candidate pair (S, Q), for every $x \in Q$, starting with an empty candidate decomposition $\mathbb{D}$, add $(S', Q') \in \mathbb{L}$ to $\mathbb{D}$ if parts 1,3 and 4 of Def. 22 hold for (S_i, Q_i). If eventually Def. 22 holds, return the decomposition $\mathbb{D}$, otherwise return that no decomposition exists. We call the resulting procedure *FindDecompositionGreedy*, and Alg. 2.3 with greedy decomposition check, correspondingly, *FindConsistentTreeDPGreedy*.

Lemma 24. *For separator size k, time complexity of FindConsistentTreeDPGreedy is equal to $O(n^{k+2}k)$.*

Combining Alg. 2.2 and *FindConsistentTreeDPGreedy*, we arrive at Alg. 2.6, a polynomial-time structure learning algorithm. Overall complexity of Alg. 2.6 is dominated by Alg. 2.2:

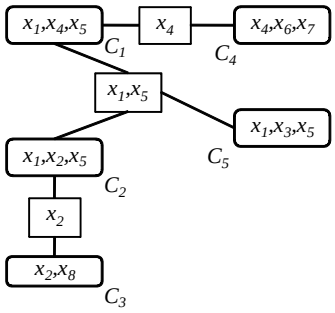
Proposition 25. *For separator size k, time complexity of Alg. 2.6 is $O\left(n^{2k+2}\left(k^3 J^{MI}_{2k+2} + n\right)\right)$.*

2.2.5 Putting it together: quality guarantees for the case of infinite samples

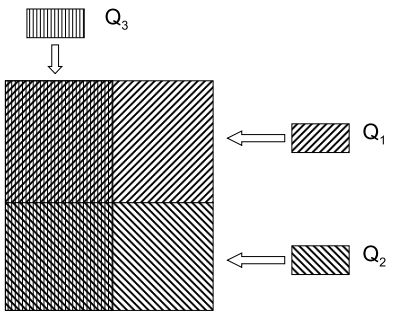
In general, *FindDecompositionGreedy* may fail to find a decomposition even when one exists. For example, suppose for a component (S, Q) and a fixed x there are 3 candidate subcomponents (S_1, Q_1), (S_2, Q_2), $(S_3, Q_3) \in \mathbb{L}$ such that $S_1, S_2, S_3 \subset S \cup \{x\}$ and

$$Q_1 \cap Q_2 = \emptyset, \;\; Q_1 \cup Q_2 = Q \setminus \{x\}, \;\; Q_3 \cap Q_1 \neq \emptyset, \;\; Q_3 \cap Q_2 \neq \emptyset.$$

The relationship between Q_1, Q_2 and Q_3 is depicted in Fig. 2.5b. One can see that a decomposition $\mathbb{D}(S, Q) = \{(S_1, Q_1), (S_2, Q_2)\}$ exists. However, suppose Alg. 2.5 first adds (S_3, Q_3) to the candidate decomposition. Then neither (S_1, Q_1) nor (S_2, Q_2) can be added afterwards, so the existing decomposition will be missed. Therefore, *FindConsistentTreeDPGreedy* may miss a consistent with $\mathbb{L}$ junction



(a) An example of a nonmaximal junction tree.



(b) A potentially problematic instance of the exact cover problem.

Figure 2.5

tree, and consequently fail to output a junction tree even if the distribution $P(X)$ is k-JT ε-representable. However, there is a class of distributions for which Alg. 2.6 is guaranteed to find a junction tree.

Intuitively, *FindConsistentTreeDPGreedy* will always find a junction tree if

1. $\mathbb{L}$ contains the component $(S, X_{S \to C})$ for every directly connected pair of $S \in \mathbb{S}$ and $C \in \mathbb{C}$ from the "true" $(\mathbb{T}, \mathbb{C}, \mathbb{S})$.
2. There are no components (S', Q') in $\mathbb{L}$ that can interfere with the "true" decomposition of $X_{S \to C}$ in the way that (S_3, Q_3) does in the above example.

Indeed, if both requirements hold, then the greedy decomposition check will succeed for every $X_{S \to C}$, and $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ will be found, or some other junction tree $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ will be found before $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. In both cases, Alg. 2.6 will return a junction tree. One way to ensure that the above properties hold is to require $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ to be sufficiently *strongly connected:*

Definition 26. A junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ for $P(X)$ is α**-strongly connected** if and only if for every $S \in \mathbb{S}$, $C \in \mathbb{C}$ it holds that

$$\forall U \subset C_{\text{-}S} \text{ it holds that } I\left(U, C_{\text{-}SU} \mid S\right) > \alpha. \tag{2.10}$$

Intuitively, in a strongly connected junction tree instantiating a separator S makes variables on the different sides of S weakly dependent, but the variables within every clique remain strongly dependent: there is no way to split any clique into two weakly dependent parts.[6] Note that the clique C and separator S in Def. 26 do *not* have to be directly connected in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$; property (2.10) has to hold for *every pair* of C and S. Also, observe that Def. 26 is stronger than similar definition of strong connectivity by Narasimhan and Bilmes (2004): instead of requiring high conditional mutual information within every clique (2.10), Narasimhan and Bilmes (2004) only require high conditional mutual information within full connected components $X_{S \to C}$. However, both for our structure learning approach and for that of Narasimhan and Bilmes (2004), strong connectivity in the sense of Def. 26 is in fact needed to guarantee that a junction tree will be found.

If there exists a maximal junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ for a distribution $P(X)$ that both approximates $P(X)$ well and is, at the same time, sufficiently strongly connected, then LTCI is guaranteed to find the correct

[6] It is still possible for a *subset* of a clique to consist of two weakly dependent parts, this point is discussed in detail in Section 2.2.5.

partitionings $\mathbb{Q}_S = \{X_{S \to C}\}$ for every separator $S \in \mathbb{S}$, and Alg. 2.6 is guaranteed to find a good tractable approximation for $P(X)$:

Theorem 27. *If there exists a maximal AKU ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ of treewidth k for $P(X)$ such that $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is $(k+2)\varepsilon$-strongly connected, then Alg. 2.6, called with $\delta = \varepsilon$, will output a $n(k+1)\varepsilon$-JT for $P(X)$.*

Notice that the combination of maximality and strong connectivity required by Theorem 27 places significant restrictions on the space of learnable junction trees. Essentially, the conditions of Theorem 27 require the order (number of variables involved) of probabilistic dependencies to be the same throughout the true model. For example, the junction tree in Figure 2.1c is maximal, but a similar one in Figure 2.5a is not (because clique C_3 and separators $\{x_2\}$ and $\{x_4\}$ are non-maximal). Therefore, even when both JTs are strongly connected, Theorem 27 only guarantees that a good approximation to the JT in Figure 2.1c will be learned, but makes no guarantees about the JT in Fig. 2.5a. Extending the guarantees of Theorem 27 to nonmaximal junction trees is an important direction for future work.

Even though the theoretical guarantees of Theorem 27 are limited to the maximal junction trees, in Section 2.4.1 we demonstrate empirically that a variant of Alg. 2.6 can be successfully applied to learning a wide range of non-maximal distributions. Our empirical results suggest the feasibility of generalizing our theoretical guarantees beyond the case of maximal JTs.

Learning models of larger than optimal treewidth

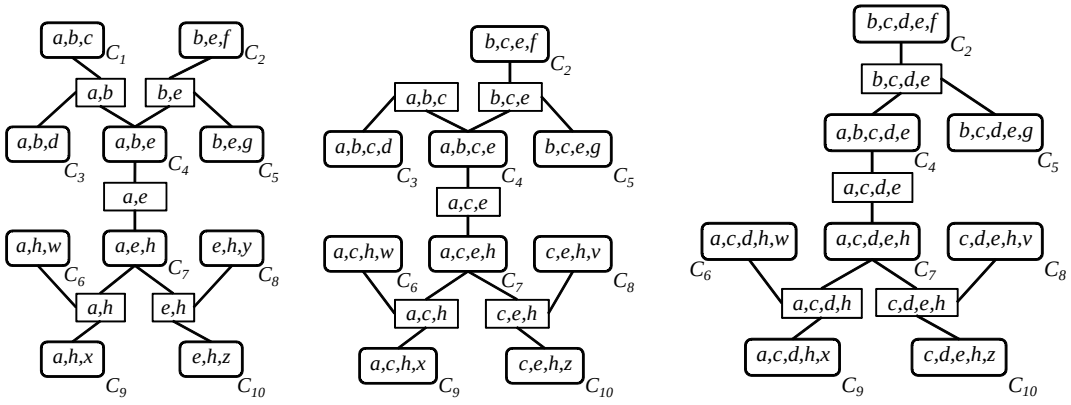
Directly applying Theorem 27 to guarantee the success of Alg. 2.6 requires knowledge of the treewidth k of the "true" ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. However, in practice, the "true" treewidth of the distribution is usually unknown. On the other hand, it is known that for every $m \leq k$, any distribution that factors according to a (not necessary maximal) junction tree of treewidth m also factors according to some junction tree of treewidth k. Moreover, it is straightforward to construct such a JT of treewidth k as follows. Denote $C^m \in \mathbb{C}$ to be a clique of size $m+1$ in the original $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ of treewidth m. Choose an arbitrary subset $U \subseteq X_{\text{-}C^m}$ such that $|U| = k - m$ and add U to every clique and separator of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Denote the resulting junction tree to be $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$. One can see that $\mathbb{C}'$ contains the clique $C^m U$ of size $k+1$ and no cliques of size larger than $k+1$, so the treewidth of $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ is k. Let us show that $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ is also an ε-junction tree. Take any $S' \in \mathbb{S}'$ and $C' \in \mathbb{C}'$ directly connected to S'. Denote S and C to be the counterparts of S' and C' in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. It holds that

$$I\left(X_{S' \to C'}, X_{C' \to S'} \mid S'\right) \equiv I\left(X_{S \to C} \setminus U_{\text{-}S}, X_{C \to S} \setminus U_{\text{-}S} \mid U_{\text{-}S} S\right) \leq I\left(X_{S \to C}, X_{C \to S} \mid S\right) = \varepsilon,$$

where the inequality follows from the chain rule (A.1) and monotonicity (A.3) of conditional mutual information. Therefore, whenever $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an ε-JT, $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ is also an ε-JT.

For example, in Figure 2.6, we show the results of increasing the treewidth of a junction tree from Fig 2.6a using $U = \{c\}$ (Fig 2.6b) and $U = \{c, d\}$ (Fig 2.6c).

Given that "fattening" the model does not decrease the approximation quality, a natural approach to dealing with unknown treewidth is to learn models with the highest treewidth that is feasible given the available data and computational resources. Therefore, it is desirable to have a guarantee that Alg. 2.6, called with treewidth parameter k, will find a high-quality structure whenever the true distribution factors according to some strongly connected maximal junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ of treewidth $m \leq k$. Unfortunately, a fattening of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is not necessarily strongly connected, even when $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ itself is (see Appendix A.1 for



(a) An Almond-Kong JT. Rounded rectangles are cliques, regular rectangles are separators.

(b) An junction tree obtained from that in (a) by adding c to every clique and separator.

(c) An junction tree obtained from that in (a) by adding c and d to every clique and separator.

Figure 2.6: Increasing treewidth of junction trees while preserving approximation quality.

an example). Therefore, the guarantee of Theorem 27 does not apply to fattened JTs directly. However, for the special case of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ that can be fattened to treewidth k using *all of the variables* of several leaf subtrees *attached to the same separator* as the extra variables set X, we provide such a learnability guarantee:

Theorem 28. *Suppose there exists a maximal AKU ε-junction tree $(\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)$ of treewidth $m \leq k$ for $P(X)$ such that*

1. *$(\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)$ is $(k+3)\varepsilon$-strongly connected.*
2. *There exists a separator $S^{m*} \in \mathbb{S}^m$ and cliques $C_1^m, \ldots, C_j^m$ directly connected to S^{m*} such that $\sum_{i=1}^{j} X_{S^{m*} \to C_i^m} = (k - m)$.*

Then Alg. 2.6, called with $\delta = \varepsilon$ and treewidth parameter k, will output a $n(k+1)\varepsilon$-JT for $P(X)$.

Note that the strong connectivity requirement in condition 1 of Theorem 28 depends on the treewidth k of the model we are trying to learn, and not on the treewidth m of the "true" junction tree. Also observe that the condition 2 of Theorem 28 is not vacuous. For example, the junction tree in Fig. 2.6a can be fattened according to the requirements of Theorem 28 by one variable using separator $S^{m*} = \{a, b\}$ and clique $C_1^m = C_1$, or by two variables using separator $S^{m*} = \{a, b\}$ and cliques $C_1^m = C_1$ and $C_2^m = C_3$, but not by three variables. However, for any maximal AKU junction tree, it is always possible to increase the treewidth by one or two variables according to the requirements of Theorem 28. Therefore, as long as the original JT is sufficiently strongly connected, learning a JT with treewidth higher by one or two is safe in the sense of graceful degradation of guarantees: increasing treewidth by one results in relaxing the quality guarantees on the result by $n\varepsilon$. Moreover, depending on the structure of the true junction tree, larger increases of treewidth are often possible.

Setting $k = m$ in Theorem 28 yields a very similar statement to Theorem 27, because the requirement of having cliques $C_1^m, \ldots, C_j^m$ such that $\sum_{i=1}^{j} X_{S^{m*} \to C_i^m} = k - m$ is satisfied trivially by an empty set of cliques. The only difference is in the required level of strong connectivity of the "true" junction tree:

$(k+2)\varepsilon$ for Theorem 27 versus $(k+3)\varepsilon$ for Theorem 28. The extra ε of strong connectivity is necessary for $k > m$.

To summarize, Theorem 28 provides a graceful quality degradation guarantee on the learned structures with respect to the treewidth parameter of the algorithm and shows that it is often possible to learn high-quality junction trees even when the treewidth of the "ground truth" structure is unknown. In particular, if the target distribution $P(X)$ factorizes exactly according to a thin junction tree, learning a higher treewidth JT often leads to no result quality decrease at all[7], because $\varepsilon = 0$. Although Theorem 28 does not extend the learnability guarantees beyond the set of maximal junction trees, it does allow for learning the models for distributions of different "true" treewidth values m using the same result treewidth k. In other words, the requirement for all the dependencies *within the same model* to have the same order m is not relaxed, but the order of dependencies *across the different true models* can be different.

Strong connectivity is a weaker property than a perfect map

The reader may find the definition of strongly connected junction tree similar to that of a *perfect map* (P-map); the latter property is also often called *faithfulness* of a distribution to a structure (c.f., Spirtes et al., 2001). Recall the P-map definition (generalized to use conditional mutual information):
Definition 29. A junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an **α-perfect map** for $P(X)$ iff for every non-intersecting $A, B, S \subset X$ such that in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ the set S does not separate A from B, it holds that $I(A, B \mid S) > \alpha$.

Intuitively, this means that the distribution $P(X)$ possesses only the (approximate) conditional independencies that are encoded in its perfect map $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ and no others. It is easy to see that every α-perfect map is also α-strongly connected by taking C, X, S from Def. 26 and setting in Def. 29 $A = U, B = C_{\text{-}SU}$. Importantly, the reverse does not hold: α-P-map is a stronger property than α-strong connectivity. In particular, if $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a P-map, then property (2.10) holds not only for cliques C, but also for all subsets of C down to pairwise dependencies.

In contrast, α-strong connectivity does not require anything of the lower-order dependencies within a clique. In fact there may be none at all. For example, if the variables of a junction tree in Fig. 2.1c are binary and the conditional distribution $P(x_1x_2x_7 \mid x_4x_5)$ is such that $x_7 = XOR(x_1, x_2)$ and x_1 and x_2 are independently uniformly distributed, then pairwise conditional mutual informations, such as $I(x_1, x_2 \mid x_4x_5)$ are all equal to 0, meaning exact conditional independence. At the same time, $I(x_1, x_2x_7 \mid x_4x_5) = 1$. Thus the clique $C = (x_1x_2x_7)$ and separator $S = (x_4x_5)$ do not satisfy the definition of an α-perfect map, for any $\alpha \geq 0$, but satisfy the definition of an α-strongly connected junction tree for $0 \leq \alpha < 1$.

To summarize, the guarantees of Theorems 27 and 28 are more widely applicable than just the (quite narrow) class of distributions with perfect maps. This is in contrast to many other constraint-based algorithms, such as that of Spirtes et al. (2001), that require the underlying distribution to have a P-map.

[7]Here, we assume perfect mutual information measurements and do not consider the question of sample complexity. Insufficient data can lead to worse overfitting with larger treewidth models, because of the larger number of required parameters. The question of sample complexity is addressed in the next section.

2.2.6 Sample complexity and PAC learnability guarantees

So far we have assumed that a mutual information oracle $I(\cdot,\cdot \mid \cdot)$ exists for the distribution $P(X)$ and can be efficiently queried. In real life, however, one only has data (that is, samples from $P(X)$) to work with. Fortunately, it is possible to estimate $I(A, B \mid S)$ from data with accuracy $\pm\Delta$ with probability $1-\gamma$, using number of samples and computation time polynomial in $\frac{1}{\Delta}$ and $\log\frac{1}{\gamma}$ (but exponential in $|A|+|B|+|S|$). In this section, we will show that, using straightforward estimation of $I(\cdot,\cdot \mid \cdot)$ from data, our structure learning approach has polynomial sample complexity. To simplify discussion, we will concentrate on the sample complexity in the setting of Theorem 27, that is, assume that the treewidth of the "true" junction tree is known a priori. Similar guarantees can be extended to larger treewidth values of Theorem 28 in a straightforward manner.

Recall from (2.3) that conditional mutual information can be expressed as a difference of entropies. The following result can be then adapted directly for estimating $I(\cdot,\cdot \mid \cdot)$:

Theorem 30. *(Höffgen, 1993). For every $\Delta,\gamma > 0$, the entropy of a probability distribution over m discrete variables with domain size r can be estimated with accuracy Δ with probability at least $(1-\gamma)$ using $f(m,r,\Delta,\gamma) \equiv O\left(\frac{r^{2m}}{\Delta^2}\log^2\left(\frac{r^m}{\Delta}\right)\log\left(\frac{r^m}{\gamma}\right)\right)$ samples from P and the same amount of time.*

Notice that the complexity of this estimator is polynomial in $\frac{1}{\Delta}$ and $\log\frac{1}{\gamma}$, but exponential in the number of variables m. However, since our approach only needs estimates of $I(A, B \mid S)$ for $|A|+|B|+|S| \leq m = 2k+2$, in our algorithm m is a constant. Therefore, employing the estimator from Thm. 30 keeps the complexity of our approach polynomial.

Replacing the ideal exact estimator of $I(\cdot,\cdot \mid \cdot)$ with a probabilistic one affects the performance guarantees of our approach: it is now only possible to guarantee success with high probability $1-\gamma$. Also, it is necessary to allow for possible inaccuracies of the estimate within $\pm\Delta$ range. Taking these two factors into account, after working out technical details we get a variant of Theorem 27:

Theorem 31. *If there exists a maximal $(k+2)(\varepsilon+2\Delta)$-strongly connected AKU ε-junction tree of treewidth k for $P(X)$, then Alg. 2.6, called with $\delta = \varepsilon+\Delta$ and $\hat{I}(\cdot,\cdot \mid \cdot)$ based on Thm. 30, using $u \equiv f(2k+2, r, \Delta, \frac{\gamma}{n^{2k+2}})$ samples and $O(n^{2k+2}(uk^3+n))$ time, will find a $(k+1)n(\varepsilon+2\Delta)$-junction tree for $P(X)$ with probability at least $(1-\gamma)$.*

A particular implication of Thm. 31 is that if $P(X)$ is exactly representable by a maximal junction tree $(\mathbb{T},\mathbb{C},\mathbb{S})$ of treewidth k (that is, if $\varepsilon = 0$), and $(\mathbb{T},\mathbb{C},\mathbb{S})$ is strongly connected, it is possible to achieve an arbitrarily good approximation with probability arbitrarily close to one, given the amount of time and data polynomial in $\frac{1}{\Delta}$ and $\log\frac{1}{\gamma}$. In other words, the class of strongly connected maximal junction trees is *probably approximately correctly (PAC) learnable:*[8]

Corollary 32. *If a maximal α-strongly connected AKU junction tree of treewidth k for $P(X)$ with $\alpha > 0$ exists, then for any $\beta \in (0, \alpha n^2\frac{k+1}{k+2}]$, Alg. 2.6, using threshold value $\delta = \frac{\beta}{2(k+1)n^2}$, $O\left(\frac{n^4k^2r^{4k+4}}{\beta^2}\log^2\frac{n^2kr^k}{\beta}\log\frac{nr^k}{\gamma}\right)$ samples from $P(X)$ and $O\left(\frac{n^{2k+6}k^5r^{4k+4}}{\beta^2}\log^2\frac{n^2kr^k}{\beta}\log\frac{nr^k}{\gamma}\right)$ computation time, will learn, with probability at least $1-\gamma$, a junction tree $(\mathbb{T}',\mathbb{C}',\mathbb{S}')$ such that $KL(P, P_{(\mathbb{T}',\mathbb{C}',\mathbb{S}')}) \leq \beta$.*

[8] A class $\mathbb{P}$ of distributions is PAC learnable if for any $P \in \mathbb{P}, \delta > 0, \gamma > 0$ there exists a learning algorithm that will output $P' : KL(P,P') < \delta$ with probability $1-\gamma$ in time polynomial in $\frac{1}{\delta}$ and $\log\frac{1}{\gamma}$.

2.3 Scaling up

Our structure learning algorithm (Alg. 2.6) requires the value of threshold δ as part of the input. To get the tightest possible quality guarantees on the resulting structure, we need to choose the smallest δ for which Alg. 2.6 finds a junction tree. *A priori*, this value is not known, so we need a procedure to choose the optimal δ.

A possible way to select δ is binary search. The only input necessary for binary search is the range in which to search for the value δ. Because $I(\cdot, \cdot \mid \cdot) \geq 0$, it is natural to select $\delta = 0$ as the lower bound of the range. The upper bound can also be chosen independently of the distribution $P(X)$, by exploiting the fact that for discrete random variables with domain size r, for *any distribution* $P(X)$, variable x and set S it holds that $I(x, X_{\text{-}Sx} \mid S) \leq \log r$. Therefore, for every candidate separator S, LTCI with threshold $\delta \geq \log r$ will output partitioning $\mathbb{Q}_S$ consisting only of singleton sets. Consequently, for any $\delta \geq \log r$ Alg. 2.6 is guaranteed to find a junction tree (with all cliques connected to the same separator). Therefore, we can restrict binary search to range $\delta \in [0, \log r]$.

Unfortunately, one problem remains with threshold selection using binary search: there is no guarantee that the smallest possible threshold, or the thresholds prescribed by Thm. 27 or Thm. 31, will be found. The reason for it is that neither Thm. 27, nor Thm. 31 guarantee that a junction tree will be found for *all* values of δ *above* a certain threshold. Instead, they only guarantee success for a certain single value of δ. More generally, the success of Alg. 2.6 is *not monotonic* in δ. Therefore, Alg. 2.6 with threshold selection using binary search can give *arbitrary bad* results even if sufficiently strongly connected junction tree exists for the true distribution: the guarantees of Thm. 31 and Corollary 32 do not hold for Alg. 2.6 with δ unknown a priori.

2.3.1 Finding the optimal threshold

Fortunately, it is possible to find the optimal value of δ efficiently. In this section, we will demonstrate that although threshold δ can take any nonnegative real value, for every distribution $P(X)$ it is sufficient to only look at a polynomial number of different threshold values to obtain the full range of the outcomes of structure learning.

Observe that there is only one place where δ is used: on line 3 of Alg. 2.2 that partitions $X_{\text{-}S}$ given a candidate separator S. If for every candidate S the partitioning results $\mathbb{Q}_{\text{-}S}$ are the same for two different values of δ, then these two threshold values are indistinguishable from the point of view of the rest of the Alg. 2.6. For the remainder of the chapter, we return to the assumption that an exact mutual information oracle exists. It is straightforward to replace the idealistic exact oracle with the realistic estimator from Thm. 30, and to extend theoretical results in the same way as Thm. 31 extends Thm. 27.

The partitioning results $\mathbb{Q}_S$ depend only on the outcomes of the comparisons between δ and $\min_{U \subset W} I(U, W_{\text{-}U} \mid S)$ for all S of size k and W of size at most $k+2$. Therefore, if we were to start with $\delta = 0$ and continuously increase δ, the only values at which the partitionings $\mathbb{Q}_S$ can possibly change are exactly

$$\left\{ \min_{U \subset W} I(U, W_{\text{-}U} \mid S) \mid S \subset X, W \subseteq X_{\text{-}S}, |S| = k, |W| \leq k+2 \right\}. \tag{2.11}$$

There are $O\binom{n}{2k+2} = O(n^{2k+2})$ (a polynomial number) possible combinations of W and S. Therefore, it is possible to first compute all the threshold values (2.11) at which the outcome of the partitioning

Algorithm 2.7: Efficient structure learning with optimal threshold selection

Input: X, conditional mutual information oracle $I(\cdot, \cdot \mid \cdot)$, treewidth k

1 $\mathbb{W} \leftarrow \emptyset$
2 **for** *every* $S \subset X, W \subset X_{\text{-}S}$ *s.t.* $|S| = k, |W| \leq k+2$ **do**
3 $\quad \delta \leftarrow \min_{U \subset W} I(U, W_{\text{-}U} \mid S)$
4 $\quad$ add $(\delta; S)$ to $\mathbb{W}$
5 **for** *every* $S \subset X$ *s.t.* $|S| = k$ **do**
6 $\quad \mathbb{Q}_S \leftarrow \{X_{\text{-}S}\}$
7 **for** $(\delta; S) \in \mathbb{W}$ *in the order of increasing* δ **do**
8 $\quad \mathbb{Q}_S \leftarrow LTCI(X, S, I, \delta, k+2)$
9 $\quad (\mathbb{T}, \mathbb{C}, \mathbb{S}) \leftarrow FindConsistentTreeDPGreedy(\cup_S \mathbb{Q}_S)$
10 $\quad$ **if** $(\mathbb{T}, \mathbb{C}, \mathbb{S}) \neq$ *failure* **then**
11 $\quad\quad$ **return** $(\mathbb{T}, \mathbb{C}, \mathbb{S})$

algorithm (Alg. 2.2) may change, and then to try every one of those values, in the increasing order, as an input to Alg. 2.6. This approach is guaranteed to find the optimal threshold at the cost of a factor of $O(n^{2k+2})$ increase in computation time as compared to Alg. 2.6, so the overall time complexity will remain polynomial.

Although the optimal threshold δ can be found in polynomial time, a major inefficiency remains in the approach described above: for every value of δ, the outcome of the comparisons on line 3 of Alg. 2.2 changes for only one pair of X and S (excluding degenerate cases). However, re-running Alg. 2.6 for every value of δ will lead to performing $O(n^{2k+2})$ such comparisons, a major waste of computation time. Fortunately, this problem is easy to fix: simply record $\mathbb{Q}_S$ between invocations of Alg. 2.6 for different values of δ and only recompute those $\mathbb{Q}_S$ that may actually differ for the new and old values of δ. This approach is summarized in Alg. 2.7. The complexity of Alg. 2.7 can be calculated as follows:

Stage	*Complexity*
Compute, cache, sort all necessary mutual informations	$O(n^{2k+2}(J^{MI}_{2k+2}k^3 + k \log n))$
Do for every $\delta = \min_{U \subset W} I(U, W_{\text{-}U} \mid S)$	$O(n^{2k+2})$
Recompute the corresponding $\mathbb{Q}_S$ using LTCI.	$O(n^{k+3})$
Run $GetConsistentTreeGreedy$	$O(n^{k+2})$
Total	$O\left(n^{2k+2}J^{MI}_{2k+2}k^3 + n^{3k+5}\right)$

Summarizing the properties of Alg. 2.7, we get:

Lemma 33. *Alg. 2.7 has time complexity* $O\left(n^{2k+2}J^{MI}_{2k+2}k^3 + n^{3k+5}\right)$ *. If there exists an AKU* ε*-junction tree* $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ *for* $P(X)$ *such that* $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ *is* $(k+2)\varepsilon$*-strongly connected, then Alg. 2.7 will output a* $\hat{\varepsilon}$*-junction tree for* $P(X)$ *with* $\hat{\varepsilon} \leq n(k+1)\varepsilon$*.*

2.3.2 Redundant edges: only looking at dependencies that matter

The complexity of Alg. 2.7 can be improved even further. In this section, we will show that for the vast majority of the values of δ used by Alg. 2.7 the partitioning $\mathbb{Q}_S$ right before a call to LTCI on line 8 and right after the call to LTCI will be the same. For such values of δ, one can skip calling $FindConsistentTree$,

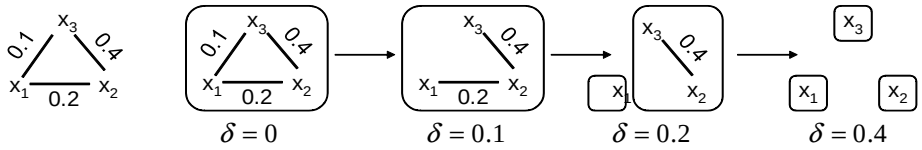


Figure 2.7: An example of evolution of partitioning $\mathbb{Q}_S$.

because the input to $FindConsistentTree$ will not have changed since the previous call. We will also show how to identify such redundant values of δ efficiently, without having to call LTCI.

It will be useful for the remainder of this section to think of partitions $Q \in \mathbb{Q}_S$ as of connected components of a hypergraph over variables $X_{\text{-}S}$ with hyperedges $\mathbb{W}_S$. Every hyperedge $W \in \mathbb{W}_S$ of the hypergraph is annotated with its *strength:*

$$\text{strength}_S(W) \equiv \min_{U \subset W} I\left(U, W_{\text{-}U} \mid S\right) \geq 0.$$

Let $\mathbb{W}_S[\delta]$ be the set of all hyperedges $W \subset X$ with strength greater than δ and such that $|W| \leq q$:

$$\mathbb{W}_S[\delta] = \{W \mid W \subseteq X_{\text{-}S}, |W| \leq q, \text{strength}_S(W) > \delta\}. \tag{2.12}$$

Observe that for every pair δ_1, δ_2 such that $\delta_1 \geq \delta_2$ it holds that $\mathbb{W}_S[\delta_1] \subseteq \mathbb{W}_S[\delta_2]$. In particular, for every $\delta \geq 0$ it holds that $\mathbb{W}_S[\delta] \subseteq \mathbb{W}_S[0]$. To simplify the discussion, we will assume for the remainder of this section that no two hyperedges of nonzero strength (that is, no two edges from $\mathbb{W}_S[0]$ have the same strength). This uniqueness assumption, however, can be relaxed without affecting any of the results of this section.

Let us set the initial value of δ to $\delta = 0$ and consider the evolution of connected components $\mathbb{Q}_S$ of a graph with nodes $X_{\text{-}S}$ and hyperedges $\mathbb{W}_S[\delta]$ as δ increases. As the threshold grows, the hyperedges with the least strength disappear, potentially splitting a connected component $Q \in \mathbb{Q}_S$ into several smaller ones. Fig. 2.7 shows an example of the evolution of $\mathbb{Q}_S$ for $X_{\text{-}S} = \{x_1 x_2 x_3\}$ if only pairwise dependencies are taken into account.

Observe in Fig. 2.7 that the removal of the hyperedge $x_1 - x_3$ did not affect the connected components. We will call such hyperedges **redundant.** Analogously, when the removal of an edge (with edge removals ordered by the increasing strength) changes the connected components $\mathbb{Q}_S$ for separator S, we will call such edge **non-redundant** for that separator. For example, the edge $x_1 - x_2$ in Fig. 2.7 is a non-redundant edge. Denote $\mathbb{NRD}_S$ to be the set of non-redundant edges for separator S.

Notice that there are at most $|X_{\text{-}S}| - 1$ non-redundant edges: as δ increases, every removal of a non-redundant edge increases the number of connected components in $\mathbb{Q}_S$ by at least one, and after all edges are removed, $\mathbb{Q}_S$ consists of $|X_{\text{-}S}|$ singleton sets. Now, observe that typically a candidate separator S does not partition the variables of $X_{\text{-}S}$ into several conditionally independent sets (in the extreme case, for a distribution faithful to a given junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, only $S \in \mathbb{S}$ involve non-trivial conditional independencies). Therefore, usually $\mathbb{W}_S[0]$ contains $O\binom{n}{k+2} = O(n^{k+2})$ hyperedges. Since there are at most $|X_{\text{-}S}| = O(n)$ non-redundant edges, the vast majority of hyperedges from $\mathbb{W}_S[0]$ typically are redundant. In other words, when the threshold δ crosses the value $\delta = \text{strength}_S(W)$ for some S and W, then $\mathbb{W}_S[\delta]$ will necessarily change, but in most cases all connected components $\mathbb{Q}_S$ will remain the same. Because the rest of the structure learning algorithm only depends on $\mathbb{Q}_S$, we would like to identify the non-redundant hyperedges and skip the runs of LTCI and dynamic programming if the partitionings $\mathbb{Q}_S$ do not change.

Algorithm 2.8: Discover the non-redundant hyperedges

Input: X, mutual information oracle $I(\cdot,\cdot \mid \cdot)$, separator S, max hyperedge size q

1 $\mathbb{W}_S \leftarrow \emptyset$
2 **for** *every* $W \subset X_{\text{-}S}$ **s.t.** $|W| \leq q$ **AND** $\text{strength}_S(W) > 0$ **do**
3 add W to $\mathbb{W}_S$
  // Below, $\text{strength}_S(W)$ *is computed using oracle* $I(\cdot,\cdot \mid \cdot)$ *and Queyranne's algorithm*
4 **for** *every* $Y \in \mathbb{W}_S$ **s.t.** $\text{strength}_S(Y) \leq \text{strength}_S(W)$, *including* $Y = W$, **do**
5 **if** Y *is redundant w.r.t.* S *and* $\mathbb{W}_S$ *as per Def. 34* **then**
6 remove Y from $\mathbb{W}_S$
7 **return** $\mathbb{W}_S$

Intuitively, an edge W is redundant whenever there exist stronger edges that connect all the variables of W (and probably also some other variables). Formally, we have

Definition 34. A **hyperedge** W **is redundant** w.r.t. separator S and set of hyperedges $\mathbb{W}$ iff either $\text{strength}_S(W) = 0$ or there exists a set of hyperedges $\mathbb{R}(W \mid S, \mathbb{W}) \equiv \{W_1, \ldots, W_m\} \subseteq \mathbb{W}$, which we call a **redundant set** for W w.r.t. S and $\mathbb{W}$, s.t.

- $W \subseteq \cup_{i=1}^{m} W_i$,
- for every $W_i \in \mathbb{R}(W \mid S, \mathbb{W})$ it holds that $\text{strength}_S(W_i) > \text{strength}_S(W)$,
- All the nodes $w \in W$ belong to the same connected component of a graph with hyperedges $\mathbb{R}(W \mid S, \mathbb{W})$.

We will mostly discuss redundancy w.r.t. S and $\mathbb{W} \equiv \mathbb{W}_S[0]$. To lighten notation, in the remainder of the chapter we will only explicitly specify $\mathbb{W}$ if $\mathbb{W} \not\equiv \mathbb{W}_S[0]$.

Observe that redundancy is monotonic in the set $\mathbb{W}$: for any $\mathbb{W}_1, \mathbb{W}_2$ s.t. $\mathbb{W}_1 \subseteq \mathbb{W}_2$, it holds that

$$W \text{ is redundant w.r.t. } S \text{ and } \mathbb{W}_1 \Rightarrow W \text{ is redundant w.r.t. } S \text{ and } \mathbb{W}_2. \tag{2.13}$$

Using the monotonicity of hyperedge redundancy, one can construct an algorithm (Alg. 2.8) to find all non-redundant edges of size at most q for a particular separator S (denoted $\mathbb{N}_S$). Throughout its execution, Alg. 2.8 maintains a set $\mathbb{W}_S$ of candidate non-redundant edges. For every edge W not seen previously, Alg. 2.8 tries to determine (a) whether knowing W helps one conclude that any of the edges in $\mathbb{W}_S$ are redundant and (b) whether W itself is redundant (lines 4–5). The redundancy check (line 5) is performed only on edges with strength at most $\text{strength}_S(W)$ (line 4), including W itself. By definition, W cannot be in a redundant set of any edge stronger than itself, so not checking stronger edges for redundancy saves computation without affecting the result. One can prove that Alg. 2.8 returns exactly the set $\mathbb{N}_S$ of non-redundant hyperedges:

Lemma 35. *Alg. 2.8 has time complexity* $O\left(n^q q^3 J^{MI}_{q+k} + n^{q+2} q\right)$ *(the first component is the complexity of computing* $\text{strength}_S(\cdot)$ *of the hyperedges, the second—of checking Def. 34 on line 5). Alg. 2.8 outputs* $\mathbb{W}_S \equiv \mathbb{N}_S$, *where* $\mathbb{N}_S$ *id the set of non-redundant hyperedges for separator* S.

Using Alg. 2.8, we can modify the algorithm that finds the optimal threshold (Alg. 2.7) to skip the attempts to find partitioning $\mathbb{Q}_S$ using LTCI when it is known that only redundant edges disappear from $\mathbb{W}_S$. The new algorithm (Alg. 2.9) is much more efficient:

Algorithm 2.9: Efficient structure learning exploiting non-redundant hyperedges

Input: X, conditional mutual information oracle $I(\cdot, \cdot \mid \cdot)$, treewidth k

$\mathbb{N} \leftarrow \emptyset$
for *every* $S \subset X$ *s.t.* $|S| = k$ **do**
  $\mathbb{N}_S \leftarrow$Alg. 2.8(X, I, S, $k+2$)
  for every $W \in \mathbb{N}_S$, add $(S; W)$ to $\mathbb{N}$
for *every* $(S; W) \in \mathbb{N}$ *in the order of increasing* $\mathrm{strength}_S(W)$ **do**
  // $\mathrm{strength}_S(W)$ is computed using $I(\cdot, \cdot \mid \cdot)$ within Alg. 2.8 and cached
  $\delta = \mathrm{strength}_S(W)$
  recompute $\mathbb{Q}_S$ using hyperedges from $\mathbb{N}_S$ with strength above δ
  $(\mathbb{T}, \mathbb{C}, \mathbb{S}) \leftarrow FindConsistentTreeDPGreedy(\cup_S \mathbb{Q}_S)$
  if $(\mathbb{T}, \mathbb{C}, \mathbb{S}) \neq$ *failure* **then**
    return $(\mathbb{T}, \mathbb{C}, \mathbb{S})$

Stage	*Complexity*
For every candidate separator...	$O(n^k)$
Run Alg. 2.8	$O(n^{k+2} J^{MI}_{2k+2} k^3 + k n^{k+4})$
For every non-redundant edge...	$O(n^{k+1})$
Recompute $\mathbb{Q}_S$	$O(n)$
Run $GetConsistentTreeGreedy$	$O(n^{k+2})$
Total	$O(n^{2k+2} J^{MI}_{2k+2} k^3 + k n^{2k+4})$

Lemma 36. *Alg. 2.9 has time complexity $O\left(n^{2k+2} J^{MI}_{2k+2} k^3 + k n^{2k+4}\right)$. If there exists an AKU ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ for $P(X)$ such that $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is $(k+2)\varepsilon$-strongly connected, then Alg. 2.9 will output a $\hat{\varepsilon}$-junction tree for $P(X)$ with $\hat{\varepsilon} \leq n(k+1)\varepsilon$.*

Comparing the complexity of Alg. 2.9 (Lemma 36) with that of Alg. 2.7 (Lemma 33), one can see that the time complexity component related to mutual information computation , $O\left(n^{2k+2} J^{MI}_{2k+2} k^3\right)$, is the same for both algorithms, but exploiting edge redundancy helps bring down the complexity of constructing the actual structure by a factor of n^{k+1} : $O(k n^{2k+4})$ for Alg. 2.9 versus $O(n^{3k+5})$ for Alg. 2.7.

2.3.3 Lazy evaluation of mutual information

Let us now look at structure learning from slightly different perspective: what if we had a procedure to make (educated) guesses $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ of the structure of a good junction tree. Then we could use the following approach:

- Get a new guess $(\mathbb{T}, \mathbb{C}, \mathbb{S})$.
- Use Corollary 17 to evaluate the quality of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$.
- If $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a high-quality structure (that is, an $\hat{\varepsilon}$-junction tree for some small $\hat{\varepsilon}$), stop and output $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Otherwise repeat.

The advantage of the above approach is that evaluating the quality of one junction tree using Corollary 17 is relatively cheap: $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ has $|X| - k = O(n)$ separators. Using the complexity discussion of Sec-

Algorithm 2.10: Structure learning with lazy evaluation of conditional mutual information

Input: X, conditional mutual information oracle $I(\cdot, \cdot \mid \cdot)$, treewidth k

1 **for** *every* $S \subset X$ *s.t.* $|S| = k$ **do**
2 $\mathbb{W}_S \leftarrow$Alg. 2.8(X, I, S, 2)
3 $\mathbb{N} \equiv \cup_S \cup_{W \in \mathbb{W}_S} (S; X)$ *// When* $\mathbb{W}_S$ *change,* $\mathbb{N}$ *is updated correspondingly*
4 **for** *every* $(S; W) \in \mathbb{N}$ *in the order of increasing* $\mathrm{strength}_S(X)$ **do**
5 $\delta = \mathrm{strength}_S(W)$
6 recompute $\mathbb{Q}_S$ using hyperedges from $\mathbb{W}_S$ with strength above δ
7 $(\mathbb{T}, \mathbb{C}, \mathbb{S}) \leftarrow FindConsistentTreeDPGreedy(\cup_S \mathbb{Q}_S)$
8 **if** $(\mathbb{T}, \mathbb{C}, \mathbb{S}) \neq$ *failure* **then**
  // Note that CheckJTQuality *routine below may modify* $\mathbb{W}_S$ *and consequently* $\mathbb{N}$
9 **if** *CheckJTQuality(*$(\mathbb{T}, \mathbb{C}, \mathbb{S})$, $\{\mathbb{W}_S \mid S \in \mathbb{S}\}$, k, δ*)* $\equiv$ *success* **then**
10 **return** $(\mathbb{T}, \mathbb{C}, \mathbb{S})$

tion 2.2.2, observe that the complexity of evaluating one junction tree using Corollary 17 is thus

$$O(n) \times O\left(n^{k+1} J^{MI}_{2k+1}\right) = O\left(n^{k+2} J^{MI}_{2k+1}\right),$$

which is a factor of n^k better than Alg. 2.9.

Of course, the performance of this "guess-evaluate-repeat" approach depends on how good the guessed structures are, and what is the complexity of the guessing procedure. In this section, we show how to modify Alg. 2.9 in the spirit of "guess-evaluate-repeat" approach. Although this modification has fewer theoretical guarantees, it performed well in practice.

Observe that for many separators, namely those not included in the resulting junction tree, it is often an overkill to find the exact set of non-redundant edges. Consider again the evolution of $\mathbb{Q}_S$ in Fig. 2.7. Suppose S is not actually used in the resulting junction tree and Alg. 2.9 finds a JT with $\delta = 0.05$. For this value of threshold, it is sufficient to examine any pair of pairwise dependencies in $X_{\text{-}S}$, for example $x_1 - x_2$ and $x_1 - x_3$, even though $x_1 - x_3$ is a redundant edge, to conclude that $\mathbb{Q}_S$ consists of a single connected component $\{x_1 x_2 x_3\}$. Generalizing, we conjecture that for many candidate separators it may be possible to only compute mutual information for low-order dependencies (pairwise, triplets, etc) and to *never need to compute mutual information for larger subsets*.

Let us describe a heuristic that aims at exploiting this phenomenon. The key idea is to *interleave partitioning* of sets $X_{\text{-}S}$ *with dynamic programming.* Observe that throughout its execution Alg. 2.8 maintains a set of candidate non-redundant hyperedges $\mathbb{W}_S$. Therefore, we can interrupt Alg. 2.8 at any time and use current $\mathbb{W}_S$ instead of the true non-redundant hyperedges $\mathbb{NRD}_S$. Stopping early helps avoid evaluating the strength of higher-order hyperedges. This idea is summarized in Alg. 2.10, where lines 1–7 correspond to the "guess" phase and line 9 (Alg. 2.11) corresponds to the "evaluate" phase.

The problem with stopping Alg. 2.8 early, of course, is the risk of missing the high-order dependencies and losing the guarantee that the connected components are weakly dependent (α-weakness in terms of Lemma 21). To fix this problem, we need to check all the higher-order dependencies for the separators that are involved in the junction tree that $FindConsistentTree$ *actually finds* (see line 9 of Alg. 2.10 and Alg. 2.11). If a higher-order dependency is discovered that involves more than one connected component

Algorithm 2.11: *CheckJTQuality*—check for strong dependencies contradicting the JT structure; modify candidate set of non-redundant hyperedges if needed.

Input: $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, candidate non-redundant edges $\mathbb{W}_S$ for every $S \in \mathbb{S}$, treewidth k, edge strength threshold δ

```
1 for every S ∈ 𝕊 do
2     for every W ⊂ X_{-S} s.t. |W| ≤ k + 1 do
3         if strength_S(W) > δ then
4             add W to 𝕎_S, remove from 𝕎_S the edges redundant w.r.t. S and 𝕎_S
5         if ℚ_S has changed then
6             return failure
7 return success
```

$Q_1, \ldots, Q_m \in \mathbb{Q}_S$, the junction tree found is rejected (line 6 of Alg. 2.11) and the process repeats with an updated set of candidate relevant hyperedges for $\mathbb{Q}_S$ (c.f. line 9 of Alg. 2.10). This after-the-fact check ensures the *a posteriori* quality guarantee of the resulting structure:

Theorem 37. *The worst case complexity of Alg. 2.10 is* $O\left(n^{2k+1} J^{MI}_{2k+1} k^3 + n^{3k+3}\right)$. *Alg. 2.10 always returns a junction tree* $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. *If* δ^* *is the value of* δ *when Alg. 2.10 returns* $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, *then* $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ *is an* $n(\varepsilon + k\delta^*)$*-junction tree for* $P(X)$.

Although there are no formal guarantees of complexity improvements for the lazy heuristic, we found it to work quite well in practice. Moreover, because only a small number of dependencies is examined before the calls to dynamic programming, Alg. 2.10 finds the first candidate junction trees very fast, making it an almost *any-time* algorithm.

2.4 Experiments

To gain more insight about the behavior of our approach, we have coded a proof of concept implementation of Algorithms 2.9 and 2.10 with slight modifications for efficiency and applied them to several synthetic and real-life datasets.

2.4.1 Synthetic data

To compare the worst-case sample complexity bounds of Theorem 31 and Corollary 32 with the average-case behavior, we have generated α-strongly connected junction trees of treewidth $k = 2$ with varying strong connectivity $\alpha \in \{0.002, 0.005, 0.01, 0.02, 0.05\}$, number of variables $n \in 10 \ldots 30$, and variable cardinality $R \in \{2, 3, 4, 5\}$. We then applied Alg. 2.9 to recover the structure of the junction trees. Because the true tractable models are known in this case, we are able to calculate exactly both the KL divergence $KL(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')})$ from the true distribution to the projection on the learned structure $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$, and the similarity of the true and learned structures in a purely graphical sense.

The random junction trees were generated as follows:

1. Start with a single clique $\mathbb{C} = \{C\}$ of size $k + 1$. Set degree$(C) \leftarrow 0$.

2. While $X \neq \cup_{C \in \mathbb{C}} C$ do:
   (a) randomly select a clique $C \in \mathbb{C}$ such that degree$(C) < r = 5$.
   (b) select a variable $x \in C$ uniformly at random, add separator $S = C_{\text{-}x}$ to $\mathbb{S}$ and edge $(C - S)$ to T.
   (c) set degree$(C) \leftarrow$ degree$(C) + 1$.
   (d) select any variable $y \in X_{\text{-}\cup_{C \in \mathbb{C}} C}$, add clique $C' = Sy$ to $\mathbb{C}$ and edge $(C' - S)$ to T.
   (e) set degree$(C') \leftarrow 1$.

The degree of every clique has been restricted in step 2a to be at most $r = 5$ to avoid the shallow star-like junction trees, which are relatively easy to learn. For every combination of parameters, 10 different JTs were generated. For every junction tree, we sampled sets of $100 \ldots 200000$ samples from the corresponding distribution to be used as training data.

Impact of the number of variables and strong connectivity

In this section, we concentrate on the dependence of sample complexity on the strong connectivity α and the number of variables n. First, we consider a purely graph-theoretical notion of quality of the learned structures. We will say that the structure has been *exactly recovered* if and only if the result $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ of the structure learning algorithm is *exactly the same* as the ground truth junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ from which the data has been sampled.

Observe that for every distribution P that factorizes exactly according to $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, the condition $(\mathbb{T}', \mathbb{C}', \mathbb{S}') \equiv (\mathbb{T}, \mathbb{C}, \mathbb{S})$ is at least as strong as $KL(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')}) \leq \beta$ for every $\beta \geq 0$. Therefore, the probability of exactly recovering a structure is a lower bound on the probability of recovering a structure within β in KL divergence of the optimum. Here, we validate the sample complexity bounds of Corollary 32 by empirically demonstrating that the sample complexity of exactly recovering a structure is within those bounds.

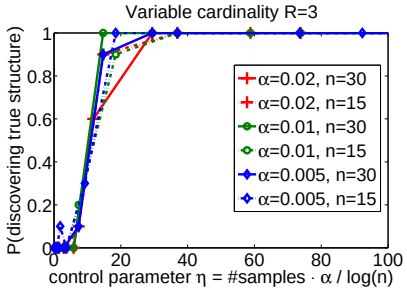
For distributions that factorize exactly according to an α-strongly connected junction tree, for fixed variable cardinality R and treewidth k, Corollary 32 guarantees that, for any $\beta \in (0, \alpha n^2 \frac{k+1}{k+2}]$, using $O\left(\frac{n^4}{\beta^2} \log^2 \frac{n^2}{\beta} \log \frac{n}{\gamma}\right)$ samples, a junction tree $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ such that $KL(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')}) \leq \beta$ will be found with probability at least $1 - \gamma$. Sample complexity of Corollary 32 is decreasing in KL upper bound β. Therefore, the hardest case from the perspective of exactly recovering the ground truth structure is achieved with the maximal $\beta = \alpha n^2 \frac{k+1}{k+2}$, and the smallest number of samples $O\left(\frac{1}{\alpha^2} \log^2 \frac{1}{\alpha} \log \frac{n}{\gamma}\right)$. For a fixed failure probability γ, we get a required sample size s of

$$s = O\left(\frac{1}{\alpha^2} \log^2 \frac{1}{\alpha} \log n\right). \tag{2.14}$$

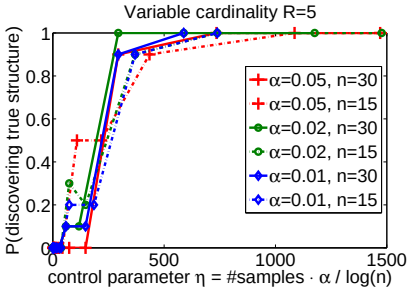
Denote

$$\eta'(s, \alpha, n) = \frac{s\alpha^2}{\log^2 \frac{1}{\alpha} \log n} \tag{2.15}$$

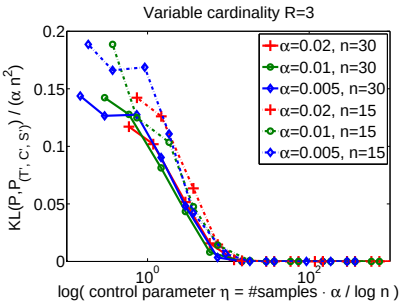
to be the *effective sample size.* The asymptotic dependence (2.14) leads to the conjecture that the effective sample size $\eta'(s, \alpha, n)$ is the parameter that controls the success of Alg. 2.9. In other words, the conjecture is that the probability of Alg. 2.9 successfully learns a high-quality junction tree depends on the parameters



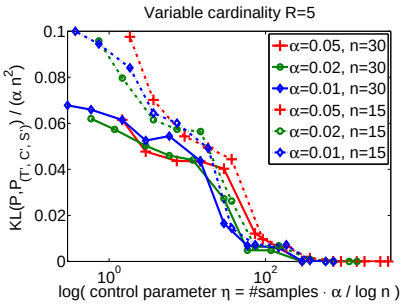
(a) Empirical share of recovered true JTs vs. effective sample size.



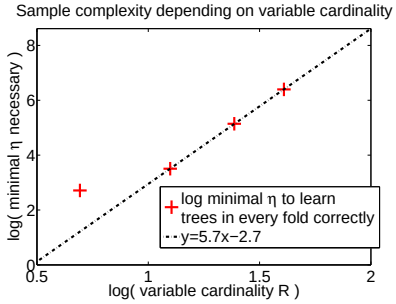
(b) Empirical share of recovered true JTs vs. effective sample size.



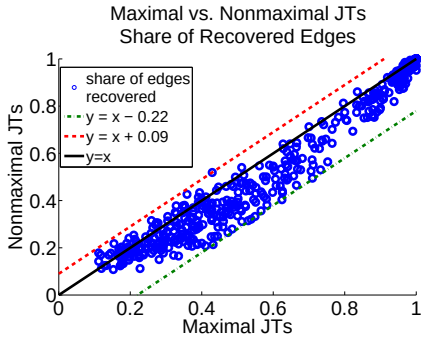
(c) Normalized KL divergence from ground truth vs. effective sample size.



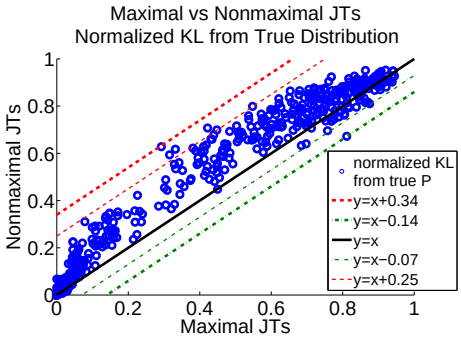
(d) Normalized KL divergence from ground truth vs. effective sample size.



(e) log-effective sample size needed to recover all ground truth JTs vs. variable cardinality.



(f) Comparison results of Alg. 2.9 on max and nonmax JTs - recovered edges.



(g) Comparison results of Alg. 2.9 on max and nonmax JTs - normalized KL.

Figure 2.8: Results on synthetic data.

s, α, n only through the value of $\eta'(s, \alpha, n)$. In our experiments, however, we have found that a different effective sample size function, namely

$$\eta(s, \alpha, n) = \frac{s\alpha}{\log n}, \tag{2.16}$$

describes the success probability much better than η'. Observe that η decreases slower than η' when $\alpha \to 0$ (linearly versus almost quadratically), which suggests that the average case sample complexity of our approach is smaller than the worst-case bounds of Theorem 31 and Corollary 32 suggest.

In Fig. 2.8a and 2.8b, we plot, as a function of the effective sample size η, the empirical share of cases where the structure of the ground truth junction trees was successfully exactly recovered. For every combination of parameter settings, the share of successes is averaged over 10 random junction trees. One can see that for a fixed variable cardinality R ($R = 3$ in Fig. 2.8a and $R = 5$ in Fig. 2.8b), the dependence of share of exactly recovered structures on η for different variable numbers n and strong connectivity values α is very similar. Such similarity suggests that η is the practically important effective sample size in the average case.

So far in this section we have treated γ as the failure probability of discovering the ground truth structure. However, Theorem 31 and Corollary 32 only make statements about discovering a structure within a certain KL divergence from the ground truth, and not necessarily the ground truth itself. We would like to compare the behavior of $KL(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')})$, where $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ is the junction tree learned by Alg. 2.9, as a function of effective sample size for different parameter settings. To make a meaningful comparison, we first need to scale the KL divergences for different parameter settings so that they are comparable. Observe that Corollary 32 can be equivalently restated as "for any $\beta' \in (0, \frac{k+1}{k+2}]$, a junction tree $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ such that $\frac{1}{\alpha n^2} KL(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')}) \leq \beta'$ will be found with high probability, given enough samples and computation time". In this alternative formulation, the range of β' does not depend on the strong connectivity α and number of variables n, which suggests that normalizing KL divergence by αn^2 will provide a measure of model quality that is directly comparable across different values of strong connectivity and number of variables. Let us define

$$KL_{\text{norm}}(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')}) \equiv \frac{KL(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')})}{\alpha n^2}.$$

In Fig. 2.8c and Fig. 2.8d we plot $KL_{\text{norm}}(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')})$ for variable cardinality $R = 3$ and $R = 5$ correspondingly. For each cardinality, we vary the strong connectivity of the original distribution and the number of variables in the model. Notice that the values of the KL_{norm} cover approximately the same range for different values of α and n, suggesting that scaling the absolute KL by $\frac{1}{\alpha n^2}$ yields a useful parameter-independent structure quality measure. One can see that the behavior of the KL_{norm} as a function of the effective sample size $\eta(s, \alpha, n)$ is very similar for different combination of model parameters, again suggesting that $O(\alpha^{-1} \log n)$ is the average case sample complexity of our approach in practice.

Impact of variable cardinality

Let us now consider the dependence of sample complexity on the variable cardinality. From Corollary 32, assuming $\beta = n^2\alpha$, we have the required sample size s of

$$s = O\left(\frac{R^{4k+4}}{\alpha^2} \log^2 \frac{1}{\alpha} \log \frac{n}{\gamma}\right),$$

and, for a fixed failure probability γ, we get $s = O\left(\frac{R^{4k+4}}{\alpha^2}\log^2\frac{1}{\alpha}\log n\right)$. Equivalently, we get

$$\frac{s\alpha^2}{\log^2\frac{1}{\alpha}\log n} = O(R^{4k+4}) \Leftrightarrow \eta'(\alpha, n, s) = O(R^{4k+4}), \tag{2.17}$$

where $\eta'(s, \alpha, n)$ is the effective sample size defined in (2.15). In the previous section, we have discussed that using $\eta(\alpha, n, s) = s\alpha\log^{-1} n$ provides a better way to control for the number of variables and sample size. We thus conjecture that the average-case sample complexity for a fixed failure probability γ has the form

$$\log\eta(\alpha, n, s) = \theta\log R + O(1).$$

Corollary 32 and Equation 2.17 show that $\theta \leq 4k + 4$.

For a given strong connectivity α and number of variables n, define $s^*(\alpha, n, R)$ to be the smallest number of samples for which all 10 junction trees in our data generated with strong connectivity α, and n variables of cardinality R each were learned successfully. To test our conjecture, for variable cardinalities $2, \ldots, 5$, and different values of α (ranging from 0.002 to 0.05) and n (ranging from 10 to 30), we have computed the average $\log\eta(\alpha, n, s^*(\alpha, n, R))$, and plotted it against $\log R$ in Fig. 2.8e. One can see that for $R = 3, 4, 5$ the dependence can be characterized with a very high accuracy by a straight line with $\theta = 5.7$. Observe that $\theta = 5.7 < 12 = 4k + 4$, so the average case sample complexity is, as in the case of dependence on α and n, grows slower than the worst-case bounds suggest.

Performance on non-maximal junction trees

The theoretical guarantees of our approach are only applicable to maximal junction trees, that is JTs where every clique is of size $k + 1$ and every separator is of size k. In practice, however, most interesting distributions are described by non-maximal trees. Fortunately, nothing prevents one from running Alg. 2.9 or Alg. 2.10 even when the assumption of the ground truth JT being maximal is violated. In this section, we investigate the empirical performance of Alg. 2.9 on nonmaximal junction trees. We have generated random strongly connected JTs of treewidth 2 with strong connectivity $\alpha \in [0.002, 0.05]$, number of variables $n \in 10 \ldots 30$ and variable cardinality R ranging from 2 to 5. For every combination of parameters 10 JTs were generated. The generation procedure was the same as described in Section 2.4.1, except that on step 2b with probability 0.5 we select two variables x and y to be replaced instead of one. The resulting separator $C_{\text{-}xy}$ then has size 1. Thus, in our nonmaximal junction trees half of the separators on average were of size 1 instead of 2. All the cliques were of size 3.

To single out the effect that non-maximality of the true junction trees has on the results of Alg. 2.9, we compared the results of learning the nonmaximal distributions with results for maximal JTs generated with the same combination of (α, n, R). Unfortunately, the metrics used previously in this section (probability of successfully learning the true structure and KL divergence) are not directly applicable. Alg. 2.9 always returns maximal structures, so the probability of learning a non-maximal one is always 0. Because nonmaximal JTs describe a distributions with fewer dependencies, and thus are easier to approximate, the KL divergences from the ground truth have to be scaled to make the values for maximal JTs comparable with those for nonmaximal JTs.

As a graph-theoretical quality measure of the learned structures, in this section we use the share of the recovered *pairwise edges* of the ground truth structure. The set of pairwise edges for $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is defined as all pairs of variables that belong to the same clique: $E(\mathbb{T}, \mathbb{C}, \mathbb{S}) = \{(x - y) \mid \exists C \in \mathbb{C} \text{ s.t. } x, y \in C\}$.

For the ground truth junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ and a learned JT $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ the share of the recovered pairwise edges is thus

$$r((\mathbb{T}', \mathbb{C}', \mathbb{S}'), (\mathbb{T}, \mathbb{C}, \mathbb{S})) = \frac{|E(\mathbb{T}', \mathbb{C}', \mathbb{S}') \cap E(\mathbb{T}, \mathbb{C}, \mathbb{S})|}{|E(\mathbb{T}, \mathbb{C}, \mathbb{S})|} \in [0, 1].$$

The attractive properties of $r((\mathbb{T}', \mathbb{C}', \mathbb{S}'), (\mathbb{T}, \mathbb{C}, \mathbb{S}))$ are as follows:

1. When $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ and $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ are both maximal JTs of the same treewidth, then $r((\mathbb{T}', \mathbb{C}', \mathbb{S}'), (\mathbb{T}, \mathbb{C}, \mathbb{S})) = 1$ if and only if $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ and $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ are the same.
2. Whenever $r((\mathbb{T}', \mathbb{C}', \mathbb{S}'), (\mathbb{T}, \mathbb{C}, \mathbb{S})) = 1$, for every distribution P that factorizes according to $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, it holds that $P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')} = P$, regardless of whether $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ and $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ are maximal.

In Fig. 2.8f, for every combination of parameters α, n, R and sample size s, we have plotted the average share of recovered pairwise edges by Alg. 2.9 for nonmaximal α-strongly connected JTs over n variables of cardinality R using s samples (we will call this quantity r_{nonmax}) versus the same quantity, but for maximal JTs (which we will call r_{max}). Every point on the plot thus represents a fixed combination of α, n, R, s. One can see that on average Alg. 2.9 recovers fewer true pairwise edges if the ground truth distribution factors according to a nonmaximal JT, so the nonmaximality does have an adverse effect on performance. On the other hand, that adverse effect is limited: $r_{\text{nonmax}} \in [r_{\text{max}} - 0.22, r_{\text{max}} + 0.09]$ for all combinations of parameters.

To directly assess the quality of approximating the ground truth distribution, in Fig. 2.8g we plot the averaged normalized KL distances

$$KL_{\text{norm-MF}}(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')}) \equiv \frac{KL(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')})}{KL(P, P_{\text{mean field}})},$$

where $P_{\text{mean field}}(X) = \prod_{x \in X} P(x)$ is the *mean field approximation*, which assumes that all the variables are independent. The normalization is necessary to make the KL divergences directly comparable across different parameter values and both maximal and nonmaximal ground truth junction trees. As in Fig. 2.8f, every point in the plot corresponds to a fixed combination of parameters α, n, R and sample size s. Similarly to Fig. 2.8f, in Fig. 2.8g one can see that the performance of Alg. 2.9 in terms of approximation quality suffers from nonmaximality of the ground truth JTs, but the quality decrease is limited: $KL_{\text{norm-MF}}$ for nonmaximal junction trees is within $[-0.14, 0.34]$ of that for maximal JTs for all parameter combinations, and within $[-0.07, 0.14]$ for 99% of all parameter combinations.

To summarize, even though the theoretical guarantees of Theorem 31 and Corollary 32 do not extend to the case of non-maximal junction trees, we have demonstrated empirically that our approach can still be successfully applied to such distribution, with only a moderate performance decrease as compared to the case of maximal JTs.

2.4.2 Real-life data

Here, we test the performance of Alg. 2.10 the following three real-life datasets:

dataset	variables number	data type	citation
TEMPERATURE	54	real-world	Deshpande et al. (2004)
TRAFFIC	32	real-world	Krause and Guestrin (2005)
ALARM	37	synthetic	Beinlich et al. (1988)

Notice we placed the ALARM data in the "real-life" datasets, even though the data has been sampled from a known Bayesian network, because the ALARM network has not only graphical structure, but also the parameter values that are very different from the typical randomly-generated graphical models such as those of Section 2.4.1.

We compared the implementation of Alg. 2.10, which we call **LPAC-JT** (from **L**azy **PAC**-learning of **J**unction **T**rees) in this section, with the following baselines:

- **Chow-Liu** algorithm introduced by Chow and Liu (1968), an efficient algorithm that learns the most likely trees (that is, junction trees of treewidth 1).
- **Order-based search (OBS)** by Teyssier and Koller (2005). For OBS, the maximum number of parents in the Bayesian network was always set to the same value as junction tree treewidth for LPAC-JT.
- **Karger-Srebro** algorithm (Karger and Srebro, 2001) with the same model treewidth as LPAC-JT.
- **Elidan-Gould** algorithm (Elidan and Gould, 2008) with the same model treewidth as LPAC-JT.
- **Local search** directly in the space of junction trees. This algorithm greedily applies the search step that yields the highest increase in the training likelihood of the model among the candidate steps, until a local optimum is found. The candidate steps were as follows:
  - *Leaf relocation.* Choose a leaf clique C, the separator S connected directly to C, and another clique C' from the current junction tree.
    - We are working with maximal junction trees, so $|C_{\text{-}S}| = 1$. Denote $C_{\text{-}S} \equiv x$.
    - Remove C from the structure.
    - Choose a subset $S' \subset C'$ of size k.
    - Add a new separator S' (unless such a separator already exists) and a new clique $C'' \equiv S' \cup x$ to the junction tree.
    - Connect S' to C' and C''.
  - *Variable propagation.* Replace a connected subcomponent of the current junction tree of form $A_1xy - S_1x - A_2xz$ (clique-separator-clique), with a subcomponent $A_1zy - S_1z - A_2xz$, provided that the running intersection property will hold in the resulting structure (RIP may cease to hold in the context of a larger component of a junction tree, for example if we replace $A_3xv - S_2x - A_1xy - S_1x - A_2xz$ with $A_3xv - S_2x - A_1zy - S_1z - A_2xz$).

  The starting points for the local search were all the *star-shaped* junction trees (the JTs with one separator S of size k and every clique connected directly to S). There are $\binom{n}{k} = O(n^k)$ possible separators S and therefore $O(n^k)$ star-shaped junction trees.
- **LPAC-JT + Local search.** This approach uses the same search steps as the local search described above, but instead of using all star-shaped junction trees as starting points, it runs Alg. 2.10 and uses the candidate junction trees constructed on line 7 of Alg. 2.10 as starting points for local search.

All experiments were run on a Pentium D 3.4 GHz. Because most of the algorithms have local nature, the runtimes were capped to 10 hours to obtain the complete picture of the search behavior. The necessary entropies were computed in advance.

2.4.3 Structure quality comparison

In this section, we compare the quality, in terms of the log-likelihood on test data, of the structures learned by the algorithms on different datasets.

ALARM. This discrete-valued data was sampled from a known ALARM Bayesian network (Beinlich et al., 1988) intended for monitoring patients in intensive care. The treewidth of ALARM network is 4, but, because of computational concerns, we learned models of treewidth 3 (at most 3 parents per variable for OBS algorithm). Fig. 2.9a shows the per-point log-likelihood of learned models on test data depending on the amount of training data.

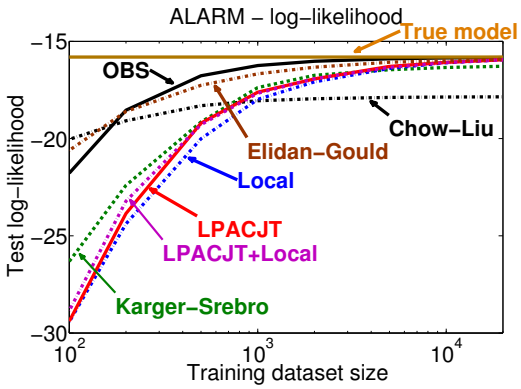
We see that on small training datasets LPAC-JT finds models of similar quality to a basic hill-climbing approach, but worse than the OBS and Elidan-Gould. For large enough datasets all variable-treewidth algorithms achieve the maximal possible likelihood of the model. The superior performance of OBS and Elidan-Gould algorithms in the small sample size regime persists for all the datasets and can be explained by the fact that out of the methods being compared, OBS and Elidan-Gould, are the only one to use regularization and favor sparser structures. All other variable-treewidth algorithms use unregularized likelihood as a quality criterion. Moreover, LPAC-JT and local search operate in the space of maximal junction trees and do not consider sparser structures at all. Chow-Liu algorithm uses a much more restricted model space than other algorithms, which can also be thought of as a way of regularization. Such implicit regularization leads to superior performance of Chow-Liu algorithm for small training set sizes. As the results for large enough training sets show, none of the algorithms supporting arbitrary model treewidth values suffer from insufficient expressive power of their respective model spaces. On the other hand, Chow-Liu algorithm performs much worse, since it is limited to models with treewidth 1, which do not have enough expressive power to capture all the dependencies of the distribution.

TEMPERATURE. This data consists of temperature readings from a network of 54 sensors taken over the course of 2 months (Deshpande et al., 2004). The readings were averaged over a 2-minute window and discretized into 4 bins. For this data, we learned models of treewidth 2. The sensors in the network were arranged in a 8-shaped pattern, allowing for complex dependencies between the readings of different sensors. The log-likelihoods of the learned models are shown in Fig. 2.9b.

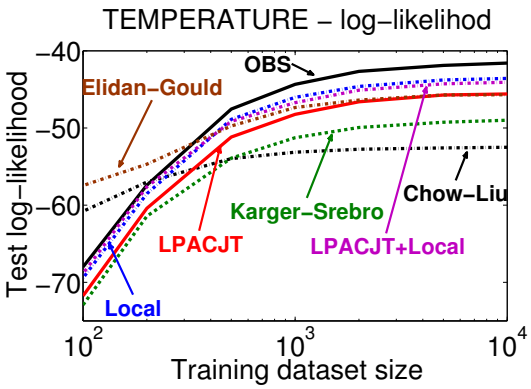
One can see that on this data OBS outperforms junction tree-based methods, suggesting a difference in expressive power from using a larger model space (limited in-degree Bayesian networks). In turn, LPAC-JT, Elidan-Gould algorithm and local search dominate Karger-Srebro algorithm, which operates with windmills instead of general junction trees. Windmills are essentially junction trees of small diameter, so we conjecture that the difference in the results is due to the difficulties of windmills in capturing long chains of indirect dependencies. Chow-Liu algorithm, as expected, provides very good results on the smallest training sets, but loses to other methods by a significant margin if there is enough data available.

TRAFFIC. This data, a part of a much larger dataset from Krause and Guestrin (2005), contains traffic flow information measured every 5 minutes in 32 locations in San Francisco Bay area for 1 month. The values were discretized into 4 bins and we learned models of treewidth 3. The resulting models log-likelihoods are shown in Fig. 2.9c.

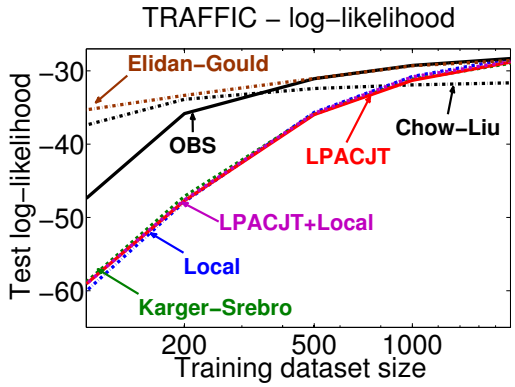
For traffic data, all non-regularized algorithms, including LPAC-JT, give results of essentially the same quality. Moreover, given enough data, all algorithms except Chow-Liu achieve the same model likelihood, suggesting that the difference in model spaces is not an issue for this dataset. Also, Chow-Liu algorithm performs very well on traffic data, which means that most of the strong dependencies between the variables



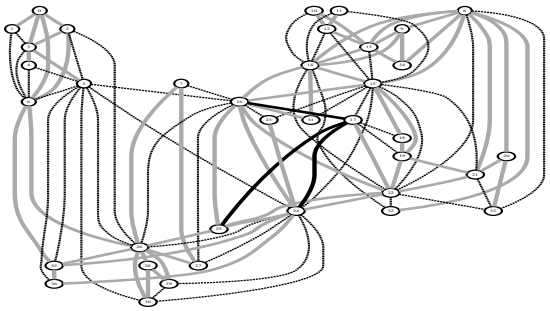
(a) ALARM log-likelihood



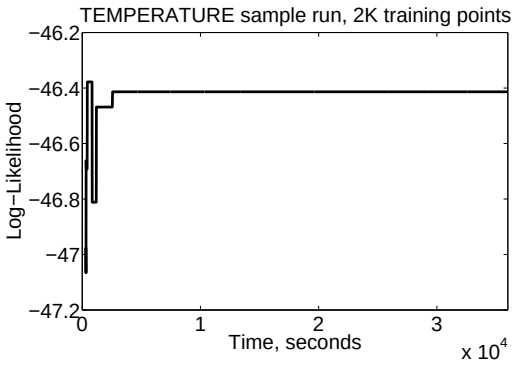
(b) TEMPERATURE log likelihood



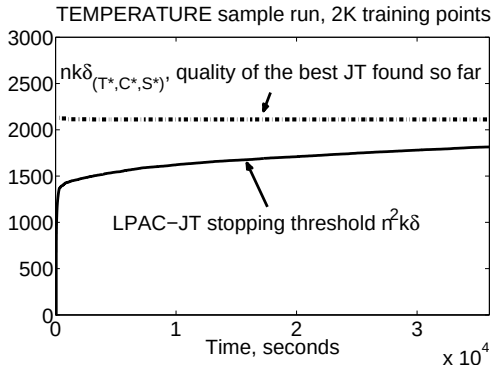
(c) TRAFFIC log likelihood



(d) ALARM structure



(e) TEMPERATURE sample run



(f) Example evolution of LPAC-JT threshold $n^2k\delta$ and the quality $n^2k\delta_{(\mathbb{T}^*,\mathbb{C}^*,\mathbb{S}^*)}$ of the best JT found so far for Alg. 2.10

Figure 2.9: (a),(b),(c): comparison of log-likelihoods of learned models.
(d): an example structure learned by LPAC-JT (nodes denote variables, edges connect variables that belong to the same clique, solid gray edges belong to both true and learned models, thin dashed black—only to the learned model, solid black—only to the true one).
(e): an example evolution of the test set likelihood of the best found model.
(f): an example evolution of the computable part of $KL(\text{empirical}, \text{model})$ and of the stopping threshold of Alg. 2.10 for that component $n^2k\delta$ (see the end of Section 2.4.4 for explanation).

are pairwise and exact tree likelihood maximization by the Chow-Liu algorithm offsets the reduction in the expressive power compared to the other algorithms. Elidan-Gould algorithm demonstrates that with proper regularization it is possible to simultaneously match the performance of Chow-Liu for small sample sizes and higher treewidth methods for larger sample sizes.

To summarize, we have shown that, given enough samples, LPAC-JT is competitive with other approaches, achieving the maximum model likelihood for 2 out of the 3 datasets, which confirms the viability of our approach.

2.4.4 Empirical properties overview

Besides the likelihood of the final structure, other properties of structure learning algorithms, such as runtime or tendency to miss important edges in the learned model, are of large practical interest. In this section, we briefly review some of these properties of LPAC-JT.

Comparison with the true model

It is important for the learning algorithms to include in the learned models all the the edges that are necessary to explain the significant dependencies in the data and at the same time to not include spurious edges. Such behavior is not only desirable from the model likelihood perspective, but also makes the models easier to interpret.

To investigate the behavior of LPAC-JT, we compared the learned models for the ALARM dataset with the known true graphical model (triangulated Bayesian network) from which the data was sampled. A typical comparison is in Fig. 2.9d, where the edges that LPAC-JT missed are thick solid black and the spurious edges (those added by LPAC-JT, but missing from the true model) are thinner dotted black. Observe that LPAC-JT captured almost all of the true edges, missing only 3 of them. At the same time, our algorithm introduced several spurious edges, because it always constructs a maximal junction tree and the true model is not maximal. Spurious edges make the learned model more difficult to interpret, but do not hurt density estimation on large datasets: given enough data, the learned models achieve the same test likelihood as the true model. For small datasets, however, dense connectivity significantly increases overfitting (by making the number of model parameters too large to be learned reliably). Thus, generalizing LPAC-JT to sparser, non-maximal junction trees is a crucial future research step.

Runtime

Even though LPAC-JT never reached the stopping criterion within the allocated 10 hours for any of the experimental runs, it typically found candidate models of very good likelihood very early on in the process. In an example run shown in Fig. 2.9e (showing the test log-likelihood of the best structure found so far versus time), the first candidate structure was found within 5 minutes. That first candidate structure already had likelihood within 2% of the eventual result. The eventual result was found within 1 hour. The following table shows that the run shown in Fig. 2.9e is a typical one for real-life datasets:

Dataset	ALARM	TEMPERATURE	TRAFFIC
Mean time to first structure	1037 sec.	59 sec.	126 sec.
Mean $\frac{\text{LLH(final)}-\text{LLH(first)}}{\lvert\text{LLH(final)}\rvert}$	-5.4×10^{-2}	3.3×10^{-3}	-1.9×10^{-2}
Mean time to final result	16721 sec.	630 sec.	3247 sec.

One can see that for real-life datasets the first structure was found, on average, in less than 15 minutes, and achieved, on average, the test likelihood just 3.3% worse than the eventual result for TEMPERATURE data and 1.9% *better* than the eventual result for TRAFFIC data. Similarly, on ALARM dataset, the first structure was on average 5.4% better than the final result (although on this artificial data it took LPAC-JT significantly longer to find a first structure). Therefore, we believe that in practice LPAC-JT can be stopped much earlier than the current stopping criterion prescribes, almost without degradation in result quality. Also the uniform stopping criterion (for every separator S from $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ and for every small $W \subset X_{\text{-}S}$ it must hold that $\text{strength}_S(W) < \delta$) looks to be overly restrictive. A different criterion, better relating to the model likelihood may improve the runtime and should be a focus of future work.

Theoretical guarantees

In addition to the log-likelihood of the candidate models that LPAC-JT finds, another important performance indicator is the upper bound on KL divergence from the empirical distribution $\widetilde{P}$ to its projection on a candidate junction tree, $KL(\widetilde{P}, \widetilde{P}_{(\mathbb{T},\mathbb{C},\mathbb{S})})$, that follows from Lemma 14 and Corollary 17. Denote $\delta_{(\mathbb{T},\mathbb{C},\mathbb{S})}$ to be the strength of the strongest conditional dependency not captured by the junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$:

$$\delta_{(\mathbb{T},\mathbb{C},\mathbb{S})} \equiv \max_{S\in\mathbb{S}, C\in\mathbb{C}, X_1\in X_{S\to C}, X_2\in X_{C\to S}, |X_1|+|X_2|\leq k+1} \text{strength}_S(X_1 \cup X_2)$$

Observe that Alg. 2.11 returns success iff $\delta_{(\mathbb{T},\mathbb{C},\mathbb{S})} \leq \delta$. For k-JT ε-representable distribution $\widetilde{P}$, from Lemma 14 we have the upper bound

$$KL(\widetilde{P}, \widetilde{P}_{(\mathbb{T},\mathbb{C},\mathbb{S})}) \leq n^2\varepsilon + n^2 k\delta_{(\mathbb{T},\mathbb{C},\mathbb{S})} \tag{2.18}$$

Unfortunately, the upper bound (2.18) depends on an unknown parameter ε and thus cannot be evaluated in the experiments. However, for any junction tree it is possible to evaluate the second component of upper bound (2.18), namely $n^2 k\delta_{(\mathbb{T},\mathbb{C},\mathbb{S})}$, that gives the value of (2.18) up to a distribution-dependent constant. Observe that the stopping criterion of Alg. 2.10 can be equivalently stated as $n^2 k\delta_{(\mathbb{T},\mathbb{C},\mathbb{S})} \leq n^2 k\delta$.

As execution of Alg. 2.10 progresses, the stopping threshold $n^2 k\delta$ increases. Simultaneously, as more candidate junction trees are discovered (c.f. line 7), the quantity $n^2 k\delta_{(\mathbb{T}^*,\mathbb{S}^*,\mathbb{C}^*)}$, computed for the best candidate JT $(\mathbb{T}^*, \mathbb{S}^*, \mathbb{C}^*)$ discovered so far, decreases. Empirical rates of change of both quantities are of interest: for example, if $n^2 k\delta_{(\mathbb{T}^*,\mathbb{S}^*,\mathbb{C}^*)}$ decreases quickly, one can conclude that Alg. 2.10 improves significantly on the candidate junction trees throughout the execution.

An example evolution of $n^2 k\delta_{(\mathbb{T}^*,\mathbb{S}^*,\mathbb{C}^*)}$ (dashed line) and of the threshold $n^2 k\delta$ (solid line) over time for one run on the TEMPERATURE data is shown in Fig. 2.9f. Similarly to the evolution of log-likelihood in Fig. 2.9e, the upper bound component $n^2 k\delta_{(\mathbb{T}^*,\mathbb{S}^*,\mathbb{C}^*)}$ does not improve significantly after the first structure is found very early in the process. Fig. 2.9f suggests that the the stopping threshold will eventually be crossed mostly because of the threshold itself increasing, not because of improvements in the quality of the results, which is another argument in favor of stopping LPAC-JT early.

Table 2.1: Representative examples of prior work.

class	model	guarantees(true P)	samples	time	representative reference
score	all	global	any	exp	Singh and Moore (2005)
score	tract./all	local	any	poly†/exp†	Della Pietra et al. (1997)
score	comp.	local	any	poly†	Teyssier and Koller (2005)
score	tract.	const. factor	any	poly	Karger and Srebro (2001)
score	tract.	local	any	poly†	Bach and Jordan (2002)
score	tract.	local	any	poly†	Lowd and Domingos (2008)
score	tract.	local	any	poly†	Elidan and Gould (2008)
score	tree	global	any	$O(n^2)$	Chow and Liu (1968)
constr.	comp.	global(comp.)	∞	poly(tests)	Spirtes et al. (2001)
constr.	comp.	PAC(comp.)/ graceful(positive)	poly	poly	Abbeel et al. (2006)
constr.	tract.	PAC(k-JT)	exp‡	exp‡	Narasimh., Bilmes (2004)
constr.	tract.	PAC(k-JT)/gracef.(ε-k-JT)	poly	poly	this chapter

tract. (tractable) means that the result is guaranteed to be of limited treewidth.
comp. (compact)—limited connectivity of the graph, but not necessarily low treewidth.
Guarantees column shows whether the result is a local or global optimum, a constant factor approximation of the optimum, whether there are PAC guarantees or graceful degradation guarantees in terms of KL divergence. For guarantees that hold only for a restricted class of input distributions, the restriction is given in parentheses.
In *time* complexity column, † superscript means per-iteration complexity of local search approaches,
$poly$ — $O(n^{O(k)})$,
$exp^\ddagger$ — exponential in general, but $poly$ for special cases.
$poly(tests)$ — polynomial complexity with an additional requirement of access to an exact conditional independence oracle (such an oracle is impossible to construct using any finite number of samples).

2.5 Related work

To place our work in context, we first review the existing results on complexity of structure learning for graphical models, restricting the discussion to the approaches that deal with *fully observed data,* that is data where for every datapoint the value of every variable is known.

As we discussed in section 1.1.1, in most settings, learning the optimal (i.e., most likely) PGM structure is provably hard: NP-hard for polytrees (Dasgupta, 1999), NP-complete junction trees of treewidth greater than 1 (Karger and Srebro, 2001) and general directed models with limited in-degree (Chickering, 1996). The exception to the negative results is the case of tree-structured models, which can be learned in $O(n^2 \log n)$ time (Chow and Liu, 1968). Also, approximately optimal structures can be learned in polynomial time. For general limited-degree graphical models, PAC-learning is possible (Abbeel et al., 2006), while for junction trees one can learn in polynomial time a structure within a fixed factor of log-likelihood of the optimum (Karger and Srebro, 2001). In the remainder of this section, we discuss the existing structure learning algorithms and their relation to our approach. For a concise summary, we refer the reader to Table 2.1.

2.5.1 Constraint-based algorithms

As discussed in Section 2.2, our structure learning algorithm belongs to the class of *constraint-based* approaches. Given the training data, constraint-based algorithms attempt to enumerate the conditional independencies of the underlying distribution and then to construct a structure consistent with those independencies. The exact formulation of a constraint-based algorithm is essentially determined by its way of dealing with two fundamental problems:

1. Because of finite training data amount, the estimated independencies may differ from the true independencies of the underlying distribution.
2. Constructing a structure consistent with a given set of independencies is not trivial.

Learning general compact models

Spirtes et al. (2001) assumed that a perfect conditional independence oracle exists and concentrated on addressing the problems of constructing a suitable structure of a Bayesian network. For distributions *faithful* to a Bayesian network with at most k parents per variable, the algorithm of Spirtes et al. (2001), called PC, is guaranteed to discover the *minimal I-map*[9] for the distribution in question. PC has polynomial in n complexity, requiring $O(n^{k+1})$ independence tests that involve at most $k+2$ variables each. Unfortunately, the guarantees of Spirtes et al. (2001) only hold for perfect independence oracle (in other words, in the limit of infinite samples). Also, the PC algorithm does not have graceful degradation guarantees when a limited in-degree Bayesian network can only represent the true distribution approximately.

Closer to our theoretical guarantees is the work of Abbeel et al. (2006), who presented an algorithm for learning factor graphs (a variant of Markov networks) with polynomial time and sample complexity. Similar to our results, Abbeel et al. (2006) provided both a PAC-learnability result for the case when the true distribution can be represented by a factor graph exactly, and graceful degradation guarantees (in KL divergence) for distributions that can only be represented approximately by limited-size factor graphs. The main difference between our theoretical guarantees and those of Abbeel et al. (2006) is that our approach is guaranteed to return a limited-treewidth model that admits efficient exact inference, while the algorithm of Abbeel et al. (2006) may return an intractable model.

Learning tractable models

Narasimhan and Bilmes (2004) introduced an algorithm that is guaranteed to return a limited-treewidth junction tree and has a PAC learnability guarantee. However, their approach requires fixed-accuracy estimates of conditional mutual information values $I(A, B \mid C)$ for sets A and B of size $\Theta(n)$. Currently, the best known methods for mutual information estimation have exponential in n complexity on such queries. Therefore, the approach of Narasimhan and Bilmes (2004) has exponential complexity. Also, Narasimhan and Bilmes (2004) do not provide graceful degradation guarantees for their algorithm for distributions that are only approximately representable by limited-treewidth junction trees. In contrast, our approach has polynomial complexity, and in addition to PAC learnability result has graceful degradation guarantees.

[9] A minimal I-map is a graph $(\mathbb{T}, X)$ such that (a) every conditional independence encoded by $(\mathbb{T}, X)$ is present in the true distribution and (b) every subgraph of $(\mathbb{T}, X)$ encodes some spurious conditional independence absent from the true distribution

Although low-treewidth models are the most extensively studied class of tractable probabilistic graphical models, there has also been progress in learning *high-treewidth* tractable models. Tractable inference in high-treewidth models is possible in the presence of context-specific independence (CSI, Boutilier et al., 1996), which imposes additional equality constraints on the values of the parameters, making it possible to deal with groups of assignments simultaneously instead of individually. However, analogously to how compactness does not guarantee tractability, not every PGM with compact potentials encoding context-specific independence is tractable. To achieve tractability, the equality constraints for different potentials have to act together to induce a specific inner structure of all the potentials.

As the standard notion of PGM structure does not take context-specific independence into account, to learn tractable high-treewidth models it is more convenient to use PGM formalisms that explicitly represent CSI via the structure, such as *arithmetic circuits* (Darwiche, 2003) or *feature trees* (Dechter and Mateescu, 2007; Gogate et al., 2010). Building upon Lemma 16 and Alg. 2.2 of this thesis, Gogate et al. (2010) have shown that approximately optimal feature trees can be learned with similar quality guarantees to those we provide for junction trees. A notable advantage of the approach of Gogate et al. (2010) over ours is that by conditioning on binary features instead of all possible joint assignments of separators simultaneously, as we do, their approach achieves better computational efficiency and, in practice, sample complexity. Gogate et al. (2010) also described a greedy version of their approach, which does not have theoretical guarantees, but works much faster in practice, and demonstrated high empirical performance of their approach.

2.5.2 Score-based algorithms

Given the inherent difficulty of structure learning, many researchers resorted to variations of heuristic search for high-likelihood structures. A large class of *score-based* structure learning algorithms (for example, Bach and Jordan, 2002; Lee, Ganapathi, and Koller, 2006; Chickering and Meek, 2002; Teyssier and Koller, 2005; Della Pietra, Della Pietra, and Lafferty, 1997) work by associating a score (usually, a form of regularized likelihood) with every possible structure, and performing a hill-climbing search in the space of structures. For certain classes of structures (limited-treewidth junction trees, Bayes nets with limited in-degree), the structure score is *decomposable*, that is, can be represented as sum of components, each of which only depends on a small subset of variables, and can be computed efficiently. When the score is decomposable, local hill-climbing search can be quite efficient in practice (Teyssier and Koller, 2005; Deshpande et al., 2001; Malvestuto, 1991; Chickering and Meek, 2002). However, likelihood is not decomposable for general undirected models: computing the likelihood requires inference in the model, which is in general intractable even for compact models. Consequently, score-based algorithms for general undirected models evaluate the scores of candidate structures only approximately (for example, Lee et al., 2006; Della Pietra et al., 1997; Choi et al., 2005).

Learning general compact models

Most hill-climbing approaches work directly in the space of Bayesian networks or junction trees, but search space design is an important research direction in score-based structure learning. A well-designed search space lets one take larger and better informed local steps, which decreases the chances of getting trapped in local optima. For example, Teyssier and Koller (2005), for learning Bayesian networks structure, instead of directly searching in the space of directed acyclic graphs with limited in-degree, perform search in the space of topological orderings of variables induced by such graphs. Given an ordering of

the variables, for every variable it is possible to exhaustively enumerate all of $O(n^k)$ candidate sets of parents, evaluate their respective scores and pick the optimal set of parents. Because the log-likelihood score of a Bayesian network decomposes into a sum of the scores of the network *families* (a family is a variable and all of its parents), decisions about optimal parent sets for different variables can be made independently. Therefore, the complexity of evaluating the score of one particular variable ordering is $O(n \times n^k) = O(n^{k+1})$, which is polynomial in the number of variables, but exponential in the in-degree. Teyssier and Koller (2005) perform such exact maximization to score a concrete variable ordering and find the best structure. Intuitively, using a globally optimal procedure as a subroutine decreases the degree of "myopia" during the local search, leading to larger changes per step and better quality of those changes.

Recently, L_1 regularization of graphical model parameters has become a popular tool for learning the structure of exponential family graphical models. L_1 regularization tends to give *sparse* results, that is, the optimal parameters have many components that are exactly zero. When a model has exactly one parameter per edge, such as Ising model with binary variables, setting a parameter to zero corresponds to removing the corresponding edge from the graph. Thus L_1 regularization of parameters implicitly imposes sparsity of the graphical model structure. As Wainwright et al. (2006) showed, for Ising models with binary nodes, L_1-regularized logistic regression provides a consistent estimator of the structure using a polynomial number of samples. L_1-regularized regression can also be used to restrict the search space of any local search algorithm for structure learning (Lee et al., 2006). In practice, the choice of regularization parameter is an important problem. Schmidt et al. (2007) showed that optimal model scores can be only achieved for the regularization parameters that correspond to discontinuities of the regularization path and proposed a way to efficiently find the approximate discontinuity locations. Restricting attention to the parameters at the discontinuities of the regularization path is similar to the idea that we employ in this chapter, namely to restrict the values of threshold δ to the set of the actual strengths of hyperedges.

Because of the local nature of score-based approaches, relatively few of them have performance guarantees. For modest sized problems, it is possible to search the space of all structures exhaustively by reusing computation (Singh and Moore, 2005). Provided that the underlying distribution has a perfect map, Chickering and Meek (2002) have shown that in the infinite data limit a two pass greedy edge addition-removal algorithm will recover the true structure. Unfortunately, infinite data assumes infinite computational cost, so this guarantee, while interesting theoretically, is inapplicable in practice.

Learning tractable models

Approaches based on local search are also a popular choice for learning tractable structures. Here also designing well-behaved search spaces and high-quality local step selection procedures is an important research direction. For example, Meilă and Jordan (2001), aiming to learn the most likely mixtures of trees, use EM algorithm to iteratively optimize the structure and parameters for every mixture component and use Chow-Liu algorithm to find exactly optimal trees for every component of the mixture in the M step. Again, as is the case with the work of Teyssier and Koller (2005), here, exact maximization by Chow-Liu algorithm allows for larger search steps than local search directly in the space of structures.

To learn low-treewidth Bayesian networks, Elidan and Gould (2008) developed developed an algorithm for discovering *sets of edges* that, when added all together to a given Bayesian network, increase the treewidth of the network by at most 1. Moreover, Elidan and Gould (2008) also show that, *given a fixed ordering over the variables,* a locally optimal edge set can be found efficiently, immediately leading to a

greedy structure learning algorithm. Here, again, the building blocks of larger local search steps (adding sets of edges instead of a single edge at a time), optimal decisions as subroutines and a higher-level search space representation (topological orderings over variables) combine to yield an approach that works very well in practice, as section 2.4.2 demonstrates. Theoretically, however, the approach of Elidan and Gould (2008) is still a local search at its core, so only local quality guarantees are provided.

A number of approaches go beyond the local perspective and provide global approximation guarantees. The algorithm of Chow and Liu (1968) is guaranteed to learn the most likely structure. Moreover, Dasgupta (1999) showed that the same most likely tree also provides a constant-factor approximation of the optimal solution to the *polytree* learning problem. More precisely, the Chow-Liu tree has likelihood at most a factor $(\frac{7}{2} + \frac{1}{2}\frac{U}{L})$ worse than the optimal polytree, where U is the maximum entropy of a single variable for the given distribution, and L is the minimum entropy of a single variable.

Close to our problem setting is the work of Karger and Srebro (2001). They consider a subclass of junction trees, which they call *windmills* and use the difference in log-likelihoods of the learned model and the model where all variables are independent as a quality criterion:

$$Quality(\text{learned}) = (LLH(\text{learned}) - LLH(\text{all independent})). \tag{2.19}$$

Karger and Srebro (2001) show that, using a linear programming relaxation and a special rounding scheme, one can always find a windmill of treewidth k that achieves at least a constant fraction of the quality of the MLE junction tree of treewidth k :

$$Quality(\text{learned windmill of width } k) \geq \frac{1}{8^k k!^2} Quality(\text{MLE junction tree of width } k). \tag{2.20}$$

The main difference of the theoretical guarantees of Karger and Srebro (2001) and our work is in their behavior for k-JT representable distributions. As the amount of data grows, the approximation factor in the guarantee of Karger and Srebro (2001) does not improve. Our algorithm, in contrast, according to Thm. 31, will find a δ-junction tree with $\delta \to 0$ as the number of samples increases, so an arbitrary quality of approximation can be achieved given enough data.

High-treewidth tractable models can also be learned via local search. Lowd and Domingos (2008) show that for arithmetic circuits (Darwiche, 2003), inference complexity can be used directly as a regularization term in a local search for the optimal structure, leading to an algorithm with good performance in practice, but few global guarantees.

To summarize, score-based algorithms provide state of the art performance in practice, but many practical approaches are based on local search and lack theoretical quality guarantees. Although our approach is not a score-based one, it is possible to incorporate some ideas from the score-based methods into our algorithm. In particular, it is possible to define a decomposable score over the sub-junction trees. Then, in the greedy decomposition search (Alg. 2.5) one can prioritize the candidate subcomponents that are being added to the decomposition on line 7 by their score (or score normalized by subcomponent size). Although not a primary objective in our algorithm, such prioritization by the score may increase the score (and thus likelihood) of the result.

2.5.3 Bayesian model averaging

Sometimes, instead of finding a single best fitting structure, one is interested in the probability of the true structure having a particular *structural feature,* such as an edge between two variables or existence

of a directed path from variable x to y. Computing the probabilities of such features is especially useful when one aims to recover the underlying causal structure of the domain, for example, when studying gene interaction. A point estimate, such as an MLE structure, would lead to many over-confident conclusions about the domain interactions structure.

Approaches that place a prior probability on every possible structure and compute the posterior probability of a given feature given the data form the class of *Bayesian model averaging* algorithms. For n variables, there are $2^{\binom{n}{2}}$ undirected graphs and $O\left(n!2^{\binom{n}{2}}C^{-n}\right)$ directed acyclic graphs (for some constant C), so exact Bayesian model averaging is in general intractable. Instead of trying to compute the posterior exactly, the standard approach (Madigan and York, 1995) is to use Metropolis-Hastings sampling with a proposal that locally modifies a structure by adding or removing an edge. One can also sample from the space of topological orderings of the variables and compute the posterior given the ordering analytically (Friedman and Koller, 2003) to get a faster mixing Markov chain. Using dynamic programming (Koivisto and Sood, 2004), it is possible to compute the posterior probability of a given edge exactly using $O(n2^n)$ time. One can also use the results of dynamic programming to guide the Metropolis Hastings sampling (Eaton and Murphy, 2007).

Bayesian averaging algorithms have an attractive property of taking all possible structures into account instead of concentrating on a single one, which is especially useful when the structure is not just a part of distribution representation, but an object of the primary interest. Bayesian averaging approaches can also be modified to perform density estimation. The main drawbacks of Bayesian averaging are high computational complexity (for example, exact averaging algorithms have complexity exponential in the number of variables) and the fact that Bayesian averaging does not result in a model that admits efficient inference (essentially, Bayesian averaging itself *is* the inference procedure).

2.6 Discussion and future work

In this chapter, we have described three main contributions:

1. A theoretical result (Lemma 16) that allows one to bound conditional mutual information of arbitrarily large sets of random variables in polynomial time.

2. A polynomial time algorithm (Alg. 2.6 and Alg. 2.9) for learning fixed-treewidth junction trees with PAC learnability guarantees for distributions exactly representable with strongly connected maximal junction trees and graceful degradation guarantees for distributions approximately representable with such junction trees.

3. A more practical development of algorithm Alg. 2.9, namely Alg. 2.10, which only considers higher-order conditional dependencies that arise in candidate junction trees constructed based on low-order dependencies. In practice such an approach significantly improves computational efficiency by "pruning" a large share of high-order conditional dependencies, although the speedup is not guaranteed theoretically and the result quality guarantees of Alg. 2.9 are lost with the lazy approach.

Here, we outline some of the directions for future work aimed at making the algorithms of this chapter more practical. Broadly, we will focus on two goals: generalizing to **junction trees with non-uniform properties** and designing **computationally efficient heuristics**.

The algorithms presented in this chapter rely on two assumptions that are rather restrictive. First, all the junction trees in question are assumed to be *maximal,* that is have *uniform clique and separator size* throughout the structure. Second, the global conditional mutual information threshold δ used to partition the variables $X_{\text{-}S}$ into weakly dependent components by LTCI enforces *uniform dependence strength* for every candidate separator S.

2.6.1 Generalizing to junction trees with non-uniform clique size

Generalizing the approach of this chapter to deal with non-maximal junction trees, including structures containing cliques and separators of different sizes will be useful in both small and large sample regimes:

- In the **small-sample regime,** it is important for the learning algorithm to be able to output non-maximal structures to prevent overfitting in learning parameters. The more densely connected a structure is, the more parameters it requires to be learned, and the more prone to overfitting it is. As the experimental results of Section 2.4.2 demonstrate, when the training data is scarce, learning sparser structures is crucial to achieve good approximations.
- In the **large-sample regime,** the parameters overfitting is not a factor, as can be seen from the results of Section 2.4.2 for large sample sizes. Much more important is the issue of learnability of non-maximal junction trees. Even learning a maximal fattening (in the sense of Section 2.2.5) of a non-maximal strongly connected ground truth JT would yield a high-quality approximation in the large sample limit. Current guarantees of Theorems 27 and 28, however, only extend to maximal junction trees. Most real-life distributions have nonmaximal structure of dependencies, so modifying our algorithms and analysis to guarantee the learnability of nonmaximal models would drastically extend the applicability of the guarantees in applications.

Out of two major components of our approach, namely estimating the strength of conditional dependencies via Lemma 16 and construction of junction trees based on dynamic programming, the former can be used for non-maximal junction trees without any modifications. Because the size of the conditioning set Y in Lemma 16 does not depend on the treewidth of a "true" ε-junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, one can use Lemma 16 for candidate separators of arbitrary size, including separators of different sizes for the same distribution.

Modifying the second component of our approach, namely dynamic programming-based structure construction, to allow for simultaneous handling of cliques and separators of different sizes would be much more challenging. Although it is rather straightforward to modify the *FindDecompositionGreedy* procedure (Alg. 2.5) to try out the available non-maximal separators and cliques during the greedy construction of a subtree over a component (S, Q), extending the learnability guarantees of Thm. 27 to non-maximal junction trees is likely to be difficult. Recall that requiring the "true" ε-junction trees for the ground truth distribution to be strongly connected made it possible to guarantee that components (S, Q) for different separators would not interfere with each other during the greedy construction of a supercomponent decomposition. When components with separators of different sizes are available for constructing a decomposition, we have an additional potential source of interference: components (S_1, Q_1) and (S_2, Q_2) where $S_1 \subset S_2$ and Q_1 and Q_2 overlap, which our theoretical results do not allow for. Possible ways to address this issue:

- Show that if there exists a strongly connected *non-maximal* true junction tree, the greedy approach will still succeed, i.e. the additional component interference is guaranteed not to happen.

- Show that for strongly connected *non-maximal* junction trees $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ it is possible to filter out the separators S' that are supersets of the true separators $S \in \mathbb{S}$ before the candidate components list $\mathbb{L}$ is formed, probably using independence testing or mutual information estimation to exclude extraneous variables and find the minimal separators.
- Replace the greedy approach with a different technique based on exact cover algorithms, for example from Knuth (2000), and identify assumptions under which the new approach will successfully discover the necessary decompositions even with some inter-component interference.

Finally, when learning nonmaximal JTs in the small-sample regime, it is also important to **bias** the algorithm **towards sparser structures**. For example, one can score the candidate components (S, Q) in Alg. 2.3 by regularized likelihood computed using the decomposition $\mathbb{D}(S, Q)$ with a regularization term depending on the number of parameters needed for $\mathbb{D}(S, Q)$. Trying higher-scoring components first in the greedy decomposition construction would then bias the learning towards sparser junction trees.

2.6.2 Generalizing to junction trees with non-uniform dependence strengths

A robust method for learning non-maximal junction trees would also present a way, although a somewhat crude one, to deal with **non-uniform** strength of **dependencies** throughout the model and retain the global conditional dependence strength threshold δ. One would simply set δ high enough to recover correct partitionings $\mathbb{Q}_S$ for separators $S \in \mathbb{S}$ such that $I\left(X_{S \to C}, X_{C \to S} \mid S\right)$ is the largest. Although setting δ high enough would lead to overly fine partitionings for other separators $S' \in \mathbb{S}$, it should be still possible in principle to recover a subtree of smaller treewidth involving S' and related variables. Essentially, with a robust enough approach for constructing non-maximal junction trees one could trade off non-uniformity in the strength of dependencies for non-uniformity in the clique and separator sizes. Moreover, after a good non-maximal tree has been discovered, one could selectively merge some of the non-maximal cliques to obtain better approximation quality.

A more straightforward possible approach to learn maximal junction trees with non-uniform dependence strengths is based on the following observation: whenever there is a non-redundant hyperedge W such that increasing the conditional mutual information threshold δ above $\mathrm{strength}_S(W)$ splits a component (S, Q) into (S, Q_1) and (S, Q_2), it means that it is *possible, but not required* for Q_1 and Q_2 to be on different sides of S. Therefore, one can keep both the connected component (S, Q) and its smaller subcomponents (S, Q_1) and (S, Q_2) in the list $\mathbb{L}$ that is used as input for *FindConsistentTreeDPGreedy*. The issue with this approach is that there is an exponential number of combinations of small subcomponents into larger ones: suppose for a certain value of threshold δ the partitioning of $X_{\text{-}S}$ found by LTCI is $\mathbb{Q}_S = \{Q_1, Q_2, Q_3\}$. Then, in addition to the elementary components $(S, Q_1), \ldots, (S, Q_3)$, all potential merges, namely (S, Q_1Q_2), (S, Q_1Q_3) and (S, Q_2Q_3) are potentially useful. In addition to exponential growth in the number of components per separator, including all possible merge results into $\mathbb{L}$ will inevitably violate property (2.9) that forms the basis of the learnability guarantees. It follows that some compromise is needed between including al possible components for a given separator S and conditional mutual information threshold δ and the approach of this chapter of only including the smallest possible components.

One possible way to select components (S, Q) for consideration by *FindConsistentTreeDPGreedy* is to use only the maximal connected components of hypergraps with edges $\mathbb{W}[\delta']$ for $\delta' \in [0, \delta]$ (c.f. Equation 2.12). For example, in the setting of Fig. 2.7, we would use components (S, x_1), (S, x_2x_3), (S, x_2) and (S, x_3), but not (S, x_1x_3), because $\{x_1, x_3\}$ has never been the *maximal* connected component of a

hypergraph with the strengths of hyperedges above certain threshold (even though $\{x_1, x_3\}$ was a non-maximal connected component for $\delta < 0.2$). Restricting the attention to maximal connected components of $\mathbb{W}[\delta']$ makes property (2.9) hold for components *corresponding to the same separator*. However, important open questions remain: (a) whether additional strong connectivity or other assumptions are needed to ensure property (2.9) for components with different separators and (b) whether including such a subset of potential components into consideration will result in more broad learnability guarantees.

2.6.3 Faster heuristics

Two factors contribute to relatively large runtimes of our approach compared to the state of the art: expensive separator scoring procedure and a large number of separators and corresponding partitionings that need to be processed both before the structure construction can begin and in the process of constructing candidate structures.

First, computing the quality of an element of the structure (in our case, a separator S and a partitioning $\mathbb{Q}_S$) requires estimating distributions $P(SW)$ with $|S| + |W| = 2k + 2$ to get a model of treewidth k, while score-based methods such as those of Elidan and Gould (2008) or Teyssier and Koller (2005) rely on scoring only *cliques* of candidate models and require estimating $P(C)$ only for $|C| = k + 1$. Because representing (an estimating) a probability distribution over m discrete variables with cardinality r has time and space complexity of $O(r^m)$, our approach has a factor of r^{k+1} disadvantage in scoring complexity.

Second, because score-based approaches do not attempt to learn the "true" structure and aim instead for a high-quality local optimum, they are able to make certain choices at the start of a run that drastically prune the space of structures that can be "seen" by the algorithm, and do not spend *any* resources evaluating structures that are inconsistent with the initial choices. For example, a local search algorithm only needs to look at structures that are reachable from the starting point via a series of hill-climbing local steps (the search space may be the space of actual structures, or the space of topological orderings over variables as in Teyssier and Koller (2005), but the general point stands). In contrast to such aggressive implicit pruning, our approach has to spend at least some computation on every possible separator.

Of the two above issues, the latter (more local perspective and implicit pruning) is much easier to address: as algorithms 2.10 and 2.11 demonstrate, it is straightforward to use Lemma 16 to evaluate the quality of candidate structures regardless of the exact way of generating those candidates. Moreover, it is also straightforward to make such quality evaluation incremental for structures that are only a local step away from an already evaluated structure (i.e., only evaluate separators S and partitionings $\mathbb{Q}_S$ that have actually changed as compared to the starting point). An alternative approach would be to explicitly restrict the set of candidate separators, for example by using a greedy heuristic similar to Gogate et al. (2010) to find the separators that are connected the strongest internally (i.e., all possible ways to split S result in subsets with high mutual information).

Although there are no obvious ways to reduce the dimensionality of probability distributions $P(SW)$ needed to apply Lemma 16 it may be possible to show that under certain assumptions, for example, similar to strong connectivity, it is sufficient to only look at pairwise conditional dependencies (i.e., have $W = 2$ instead of $k + 2$) to recover high-quality partitionings $\mathbb{Q}_S$. If one were to check only pairwise conditional dependencies, the total dimensionality of distributions that need to be estimated would be only $k + 2$. Correspondingly, the time complexity of evaluating any individual dependence strength would be $O(r^{k+2})$, which is only a factor of r slower than evaluating the likelihood of a single clique.

Chapter 3

Learning Evidence-Specific Structures for Rich Tractable CRFs

The flexibility of generative probabilistic graphical models in choosing the sets of evidence and query variable at test time is a useful feature required by many applications. However, there is also a large number of problem domains, where the set E of the evidence variables is known in advance. For example, in **computer vision** problems the colors and brightness levels of the pixels are always known, and the higher-level semantic information about the scene, such as identities and locations of the objects in an image, has to be inferred (Ladicky et al., 2009; Kumar and Hebert, 2005). In **natural language processing**, the words and their order in the document are known, and information such as part of speech tags for every word (Lafferty et al., 2001), a parse tree for a sentence (Taskar et al., 2004), or the topic of the document (Craven et al., 1998) needs to be inferred.

In applications where the set of evidence variables E is fixed and does not change at test time, the flexibility of generative models in terms of handling arbitrary evidence sets at test time is not needed. Moreover, generative learning is fundamentally unable to exploit the extra information about the identities of the evidence variables. Instead, *discriminative* learning aims to find accurate models that directly represent the conditional distribution $P(Q \mid E)$. Although discriminative models tend to be more prone to overfitting, they typically provide better accuracy that generative models, if given enough training data (Ng and Jordan, 2001). Discriminative PGMs have been very successful in practice (e.g., Ladicky et al., 2009; Lafferty et al., 2001; Vail et al., 2007).

So far, learning probabilistic graphical models in a discriminative setting was approached mainly from the perspective of parameter learning (Lafferty et al., 2001; Schmidt et al., 2008; Sutton and McCallum, 2007). Model structure has been typically assumed to be fixed, and the aspect of inference complexity in the resulting models has been largely ignored. In this chapter, we propose a novel approach for learning tractable discriminative models with *evidence-specific structure*.

The main difference of our approach from existing algorithms is in explicit dependence of the model structure on the particular *assignment* $\boldsymbol{E}$ of the evidence variables E. The extra flexibility of the dependence of the low-treewidth structure on the evidence values $\boldsymbol{E}$ lets one improve the representative power of the resulting model beyond what is possible with any fixed tractable structure. At the same time, our approach retains the advantages of efficient exact inference at test time. As a result (c.f. section 3.6), tractable discriminative models with evidence-specific structure achieve the same accuracy as high-treewidth models

on real-life datasets, while being an order of magnitude faster at test time (and also at train time for some settings).

Importantly, our approach directly builds on generative structure learning, making it straightforward to adapt many of the existing or future generative structure learning algorithms for discriminative structure learning. Moreover, because our approach affects only one step in the graphical model workflow of Fig. 1.3b, namely structure selection, it follows that existing algorithms for all the remaining steps of the workflow can be used without any changes. For example, both evidence selection via L_1 regularization (Andrew and Gao, 2007; Vail and Veloso, 2008) and fast approximate discriminative parameter learning by optimizing the pseudolikelihood (Besag, 1974) can be seamlessly integrated with our evidence-specific model structures. Finally, as we will show in detail in this chapter, evidence-specific structure learning relies on the same input data as the standard discriminative graphical models. As a result, our approach is attractive from the point of view of a practitioner for two reasons. First, there is *no loss of generality:* whenever a standard discriminative probabilistic graphical model is applicable, so are our models with evidence-specific structures. Second, for any existing application-specific "pipeline" *switching to evidence-specific model structures is almost effortless:* one only needs to replace the existing structure learning component with our approach and retain the rest of the legacy workflow.

Before proceeding to the algorithmic aspects of our approach, we review the particular PGM formalism used in this chapter, namely log-linear models, and the key differences between generative and discriminative learning.

3.1 Log-linear models, generative and discriminative weights learning

As we have discussed in section 1.1.1, the problem of learning a probabilistic graphical model, which is equivalent to selecting factors ψ_α to define a factorized distribution

$$P(X) = \frac{1}{Z} \prod_\alpha \psi_\alpha(X_\alpha),$$

is typically decomposed into *structure learning* - selecting the scopes X_α of the factors, and *parameters learning* - choosing the actual values of ψ_α given the structure. The motivation behind such a decomposition is three-fold:

1. **Discrete versus continuous optimization.** There are few good approaches for optimization problems that have both discrete (structure) and continuous (parameters) facets. One widely used exception is optimizing parameters with a sparsity-inducing regularization term such as L_1 penalty (c.f., Schmidt et al., 2007). On the other hand, purely discrete and purely continuous optimization are well-studied fields, with a wealth of existing techniques that can be transferred to the PGM setting. For example, to avoid getting trapped in local minima when searching for an optimal structures, Teyssier and Koller (2005) used tabu lists, which is a standard trick in the search literature and not specific to graphical models. Moreover, in practice, learning the parameters given a fixed structure is *relatively* easy. As a consequence, it is convenient to use parameter learning as a subroutine for evaluating the quality of candidate PGM structures.

2. **Computational complexity.** In very high-dimensional settings (for example, when every variable represents a pixel in a high-resolution image), many structure learning approaches are prohibitively expensive, because there are too many candidate edges ($O(|X|^2)$) to be considered, even without

looking at higher-order cliques. Moreover, in *relational* settings (Friedman et al., 1999; Richardson and Domingos, 2006; Taskar et al., 2002; Getoor and Taskar, 2007), where every variable represents a property of a unique object and an edge represents an instance of a relation that links different properties, every variable only occurs once in a dataset, and traditional structure learning is altogether impossible. At the same time, there is often domain knowledge or common sense that make it possible to define a model structure that closely reflects the nature of the domain. For example, webpages that are connected via a hyperlink are more likely to have related content than the ones that are not, pixels that are next to each other are more likely to belong to the same object, and so on. For such applications, restricting oneself to learning only the parameters of the model and constructing the structure using domain knowledge is more useful in practice than attempting to explore a huge space of possible structures.

3. **Interpretability.** This is a different aspect of relying on domain knowledge for defining a model structure. Often, practitioners are not graphical model experts, but have extensive domain knowledge about the structure of direct dependencies for the distribution in question. For example, in a power plant monitoring scenario, one can deduce the possible direct dependencies from the information on which systems or machines are directly connected to each other. In such applications, practitioners need to continue to operate with the existing domain-specific notions and either are unable to interpret the learned model or have little confidence in the output of a model that does not reflect the known underlying processes of the system in question. In such cases it is also useful to restrict the learning only to model parameters.

Weighted features and log-linear models

One can see from paragraphs 2 and 3 that it is often useful to be able to let the user guide the selection of the model structure. However, instead of explicitly specifying the factor scopes X_α, it has proven much more convenient to define the models structure via defining *features* $f_\alpha : X_\alpha \to \mathcal{R}$ and have the potential ψ_α be an exponent of a weighted feature:

$$\psi_\alpha(X_\alpha) \equiv \exp\{w_\alpha f_\alpha(X_\alpha)\}. \tag{3.1}$$

The weight w_α determines the relative scale of the feature f_α.

Defining the model structure via the choice of features is at least as expressive as via the choice of factor scopes: choosing the factor scope X_α is equivalent to introducing a separate feature $f_\alpha^{(\mathbf{X}_\alpha)}(X_\alpha) = \mathcal{I}(X_\alpha = \mathbf{X}_\alpha)$, with its own separate weight $w_\alpha^{(\mathbf{X}_\alpha)}$, for every possible assignment $\mathbf{X}_\alpha$ of X_α. Importantly, the reverse does not hold: explicitly specifying the features introduces ways to make use of certain types of domain knowledge that are impossible to express via just the factor scopes. Consider an example feature, expressing the intuition that nearby pixels in an image are more likely to belong to the same segment if they have similar color:

$$f(x_i, x_j, \text{color}_i, \text{color}_j, \text{loc}_i, \text{loc}_j) \equiv \mathcal{I}(x_i = x_j \wedge |\text{color}_i - \text{color}_j| < \delta_{\text{color}} \wedge |\text{loc}_i - \text{loc}_j| < \delta_{\text{loc}}). \tag{3.2}$$

Feature (3.2) illustrates two important properties that are impossible to achieve by describing the model structure only via factor scopes:

1. **Parameters dimensionality and sample complexity.** Feature (3.2) enforces additional constraints on the *values* of the corresponding factor. In (3.2), the constraint being enforced is that the factor

ψ_α can only have values 1 or e^{w_α} for all possible assignments to $\{x_i, x_j, \text{color}_i, \text{color}_j, \text{loc}_i, \text{loc}_j\}$. Restricting the set of values of a factor is not the only possible type of a constraint. For example $f(x_i, x_j) = |x_i - x_j|$ may take infinitely many different values for real x_i and x_j. What is important is that *the same weight* w_α is used *for every assignment to* X_α. Because the values of a feature are fixed, and only feature weights need to be learned, significantly reducing the weights dimensionality of a model dramatically reduces overfitting.

2. **Inference efficiency.** Feature (3.2) is zero for all assignments except for some special subset (pixels that are close by both in color space and in physical location). Boutilier et al. (1996) have shown that whenever it is known that the feature f_α (and hence the corresponding factor ψ_α) has the same value for a group of assignments to X_α, the efficiency of inference can be significantly improved by dealing with that group of assignments as a single entity instead of iterating over every individual assignment $\mathbf{X}_\alpha$.

   Moreover, for features similar to (3.2), it often holds that the values of some of the variables are *observed at test time* and in many instances $f_\alpha = 0$ for *every assignment to the unknown part* of X_α. For example, in (3.2) the colors and locations of the pixels can be directly observed from the input image. Whenever from the available information about the values of some of the variables in X one can guarantee that $f_\alpha = 0$, it follows that the factor ψ_α can be dropped from the model without affecting the distribution $P(X)$, thereby simplifying the model structure and reducing inference complexity. In the approach of this chapter, we will heavily rely on the possibility of dropping constant features from the model to dramatically speed up inference without affecting the resulting distribution.

The resulting class of factorized models with factors of form (3.1) is called the *log-linear* models (because the function under the exponentiation is *linear in feature weights* w_α):

$$P(X \mid w) = \frac{1}{Z(w)} \exp\left\{\sum_\alpha w_\alpha f_\alpha(X_\alpha)\right\}, \tag{3.3}$$

where $Z(w)$ is the normalization constant. The exponentiation in (3.1) and (3.3) brings about significant computational and theoretical benefits. Here, we will only rely on some basic properties of log-linear model and refer the reader to Wainwright and Jordan (2008) for thorough theoretical treatment.

Generative parameter learning

In the log-linear model (3.3), the features are assumed to be fixed by the user, and the parameter learning problem is equivalent to learning *optimal feature weights.* There exist different notions of optimality, leading to different optimization objectives. In the most basic case, when either the nature of the questions about the distribution $P(X)$ is not known in advance, or it is necessary to have a single model for many different queries, one aims to optimize the quality of approximation of the empirical distribution of the data $\mathcal{D}$ by the factorized model (3.3). The standard measure of approximation quality is log-likelihood, leading to the objective

$$\begin{aligned} LLH(\mathcal{D} \mid w) &\equiv \sum_{\boldsymbol{X} \in \mathcal{D}} \log\left(\tfrac{1}{Z(w)} \exp\left\{\sum_\alpha w_\alpha f_\alpha(\mathbf{X}_\alpha)\right\}\right) \\ &= \sum_{\boldsymbol{X} \in \mathcal{D}} \log\left(\sum_\alpha w_\alpha f_\alpha(\mathbf{X}_\alpha) - \log Z(w)\right). \end{aligned} \tag{3.4}$$

Maximizing the log-likelihood $LLH(\mathcal{D} \mid w)$ is equivalent to minimizing the KL divergence from the empirical distribution $P_{\mathcal{D}}(X)$:

$$\begin{aligned} KL(P_{\mathcal{D}}(X)||P(X \mid w)) &= \sum_{\boldsymbol{X}} P_{\mathcal{D}}(\boldsymbol{X}) \log \frac{P_{\mathcal{D}}(\boldsymbol{X})}{P(\boldsymbol{X} \mid w)} \\ &= -H(P_{\mathcal{D}}) - \sum_{\boldsymbol{X}} P_{\mathcal{D}}(\boldsymbol{X}) \log P(\boldsymbol{X} \mid w) \\ &= -H(P_{\mathcal{D}}) - \frac{1}{|\mathcal{D}|} LLH(\mathcal{D} \mid w). \end{aligned}$$

Observe that the entropy $H(P_{\mathcal{D}})$ of the of the empirical distribution does not depend on the model parameters w, so feature weights w can only affect KL divergence via changing the log-likelihood.

Log-likelihood (3.4) is an attractive optimization objective not only because of the direct connection to the KL divergence from the empirical distribution, but also because of its very convenient computational properties in the case of log-linear models. Namely, log-likelihood (3.4) is *concave* in w :

Definition 38 ((Boyd and Vandenberghe, 2004), Def. 3.1.1)**.** A function $F(w)$ is concave if **dom** F (the set of points on which F is defined) is a convex set and for every $\mathbf{w_1}, \mathbf{w_2} \in$ **dom** F and $\theta \in [0, 1]$, it holds that

$$F(\theta \mathbf{w_1} + (1 - \theta)\mathbf{w_2}) \geq \theta F(\mathbf{w_1}) + (1 - \theta) F(\mathbf{w_2}).$$

Fact 39 (e.g., (Getoor and Taskar, 2007))**.** Log-likelihood (3.4) is concave as a function of w.

An important property of concave functions is that any local maximum $\mathbf{w}^*$ is also a global maximum (see e.g. (Boyd and Vandenberghe, 2004) for the detailed discussion). Moreover, the log-likelihood (3.4) is continuously differentiable:

Fact 40 (e.g., (Getoor and Taskar, 2007))**.** For the log-likelihood (3.4), it holds that

$$\frac{\partial LLH(\mathcal{D} \mid w)}{\partial w_\alpha} = \sum_{\boldsymbol{X} \in \mathcal{D}} \left(f_\alpha(\mathbf{X}_\alpha) - E_{P(X_\alpha \mid w)}[f_\alpha(X_\alpha)] \right). \tag{3.5}$$

Maximizing a concave continuously differentiable objective (equivalently, minimizing a convex objective) is a well-studied problem that lies at the basis of *convex optimization* (Boyd and Vandenberghe, 2004). Many highly efficient algorithms, such as L-BFGS (Liu and Nocedal, 1989) or conjugate gradient (Fletcher and Reeves, 1964) have been developed can reliably maximize a concave continuously differentiable objective $F(w)$ and require only the ability to compute the values of $F(w)$ and the gradient $\nabla F(w)$. Moreover, state of the art convex optimization techniques converge in just a few iterations, and correspondingly require only a few objective and gradient evaluations, even when the dimensionality of parameters w is very high. Therefore, one can efficiently find the optimal optimal parameters $\mathbf{w}^*$ maximizing (3.4) by plugging (3.4) and the gradient (3.5) into an off-the-shelf convex optimization algorithm.

The only obstacle, but a very significant one, in directly applying standard convex optimization techniques to learning the graphical model parameters $\mathbf{w}^*$ that maximize the log-likelihood (3.4) is the need to compute the expected feature values $E_{P(X_\alpha \mid w)}[f_\alpha(X_\alpha)]$ in the gradient expression (3.5). Computing expected features requires *inference* in the graphical model corresponding to the factorized distribution (3.3) with the structure induced by the features f_α : the set of edges $\mathbb{T}$ is such that $(i - j) \in \mathbb{T} \Leftrightarrow \exists f_\alpha$ s.t. $x_i, x_j \in X_\alpha$. As we have discussed in section 1.1.1, inference in graphical models is intractable in general, except for special cases such as low-treewidth models. Therefore, in practice one needs to choose one of the two options:

1. Low treewidth, exact parameters. Restrict the set of features so that the resulting graphical model has structure with low treewidth. As a result, inference in the graphical model (3.3) and computing feature expectations $E_{P(X_\alpha|w)}[f_\alpha(X_\alpha)]$, the log-likelihood gradient (3.5) and the log-likelihood (3.4) itself can be done exactly. Consequently, convex optimization techniques can be used to efficiently find optimal feature weights $\mathbf{w}^*$.

2. High treewidth, approximate parameters. Keep a more expressive high-treewidth model and use approximate inference approaches such as belief propagation (Pearl, 1988) or Gibbs sampling (Geman and Geman, 1984) to compute the log-likelihood (3.4) and gradient (3.5). Because the objective and gradient can only be computed approximately, the optimization with respect to feature weights w can also be only done approximately. Moreover, approximate computation will typically break the concavity of the log-likelihood objective, and gradient-based optimization techniques would only find a local optimum with respect to w.

In chapter 2, we have demonstrated that low-treewidth structures, despite their smaller expressive power, can be competitive with high-treewidth model in overall approximation accuracy because of better parameter estimation and inference accuracy. In this chapter, we will develop a way to further improve the expressive power of low-treewidth models, by adjusting the structure depending on the observed evidence, without sacrificing the advantages of exact inference and parameter learning.

Discriminative parameter learning

The advantage of log-likelihood (3.4) as maximization objective for learning the optimal feature weights w for a log-linear model (3.3) is that it provides a single model with a good approximation for answering a wide range of queries about the distribution $P(X)$. However, often one has *a priori* information about the types of queries of interest that the factorized model (3.3) will be used to answer. When information about test time queries is available, the wide applicability of a maximum likelihood model becomes not an advantage, but a drawback. Intuitively, because (3.3) only approximates the true distribution, optimizing the model to have good accuracy on irrelevant queries "consumes the approximation power" that could be used to improve accuracy on the queries that are actually important for the end user.

Consider the kind of information about test time queries that is available particularly often: the knowledge of which variables will have their values known at test time. Typically, the set of variable whose values can be directly observed at test time is determined by the application and is *the same from query to query.* For example, in computer vision problems the colors and brightness levels of individual pixels of the images can be observed directly, while the variables encoding the characteristics about the objects present in the scene need to be inferred using the graphical model. In automated medical diagnosis systems, variables such as patient's temperature and heart rate can be measured directly, while the most likely diagnosis needs to be inferred. Importantly, the inverse problems, such as inferring the most likely image given the objects in the scene, or the patient's most likely temperature given the diagnosis are typically not relevant to the end user (and even when they are relevant, usually a different model needs to be constructed, because the same model cannot produce results of acceptable quality for both problems). Similar settings arise in natural text processing (Lafferty et al., 2001), heart motion abnormality detection (Schmidt et al., 2007), and other applications. To formalize this intuition, the random variables describing the application domain can be partitioned into two sets:

1. The variables whose values can be measured directly at test time. We will call these variable the **evidence** and denote E.

2. The variables whose values are unknown at test time and need to be inferred *given the evidence.* We will call these variable the **query** and denote X.

Given the partitioning of all the variables of the model into query X and evidence E, it follows that one is interested in the conditional distribution $P(X \mid E)$. To approximate the conditional distribution $P(X \mid E)$, it is possible to employ a generative approach: define a generative log-linear model for $P(X, E \mid w)$ as

$$P(X, E \mid w) = \frac{1}{Z(w)} \exp\left\{\sum_{\alpha} w_{\alpha} f_{\alpha}(X_{\alpha}, E_{\alpha})\right\}, \tag{3.6}$$

learn the parameters $\mathbf{w}^*$ my maximizing the log-likelihood

$$LLH(\mathcal{D} \mid w) = \sum_{(\boldsymbol{X},\boldsymbol{E})\in\mathcal{D}} \log\left(\sum_{\alpha} w_{\alpha} f_{\alpha}(\mathbf{X}_{\alpha}, \boldsymbol{E}_{\alpha}) - \log Z(w)\right). \tag{3.7}$$

with gradient

$$\frac{\partial LLH(\mathcal{D} \mid w)}{\partial w_{\alpha}} = \sum_{(\boldsymbol{X},\boldsymbol{E})\in\mathcal{D}} \left(f_{\alpha}(\mathbf{X}_{\alpha}, \boldsymbol{E}_{\alpha}) - E_{P(X_{\alpha},E_{\alpha}|w)}[f_{\alpha}(X_{\alpha}, E_{\alpha})]\right).$$

and use the conditional probability formula

$$P(X \mid E, w) \equiv \frac{P(X, E \mid w)}{P(E \mid w)} = \frac{P(X, E \mid w)}{\sum_{\boldsymbol{X}} P(\boldsymbol{X}, E \mid w)} \tag{3.8}$$

to compute the conditional distribution. However, as one can see from (3.8), such a generative approach involves modeling the evidence prior $P(E)$ as an intermediate step. A more direct way to approximate the conditional distribution $P(X \mid E)$ is to model it directly as

$$P(X \mid E, w) = \frac{1}{Z(E, w)} \exp\left\{\sum_{\alpha} w_{\alpha} f_{\alpha}(X_{\alpha}, E_{\alpha})\right\}, \tag{3.9}$$

and learn the parameters that maximize the *conditional* log-likelihood

$$CLLH(\mathcal{D} \mid w) = \sum_{(\boldsymbol{X},\boldsymbol{E})\in\mathcal{D}} \log P(\boldsymbol{X} \mid \boldsymbol{E}, w) = \sum_{(\boldsymbol{X},\boldsymbol{E})\in\mathcal{D}} \left(\sum_{\alpha} w_{\alpha} f_{\alpha}(\mathbf{X}_{\alpha}, \boldsymbol{E}_{\alpha}) - \log Z(\boldsymbol{E}, w)\right). \tag{3.10}$$

Conditional log-likelihood (3.10) has similar properties to the log-likelihood 3.4:
Fact 41 (e.g., (Getoor and Taskar, 2007))**.** Conditional log-likelihood (3.10), is concave in w. Moreover,

$$\frac{\partial \log P(\boldsymbol{X} \mid \boldsymbol{E}, w)}{\partial w_{\alpha'}} = f_{\alpha}(\mathbf{X}_{\alpha}, \boldsymbol{E}_{\alpha}) - \mathbb{E}_{P(X_{\alpha}|\boldsymbol{E}_{\alpha},w)}\left[f_{\alpha}(X_{\alpha}, \boldsymbol{E}_{\alpha})\right], \tag{3.11}$$

where $\mathbb{E}_P$ denotes expectation with respect to a distribution P.

It follows that the same convex optimization techniques that can be used to optimize log-likelihood (3.4) and (3.7) can be also used to efficiently optimize the conditional log-likelihood (3.10). Learning parameters by optimizing conditional log-likelihood is called *discriminative learning* (in contrast to generative learning using (3.7) as an objective; see Ng and Jordan (2001) for a detailed comparison of the two approaches). The structured log-linear model (3.9) with parameters w learned by optimizing (3.10) is called

a **conditional random field** (CRFs). Conditional random fields were introduced by Lafferty et al. (2001) and have been successfully applied to domains from natural text processing (Lafferty et al., 2001) to computer vision (Saxena et al., 2008) to activity recognition (Vail et al., 2007).

Although conditional random fields (3.9) and generative log-linear models (3.6) have very similar form, modeling the conditional distribution $P(X \mid E, w)$ directly leads to several key distinctions between the two approaches:

1. **Inference complexity.** At *test time,* inference complexity of both approaches is determined by the treewidth of the graph over variables X induced by the features f_α. The same inference complexity follows from the fact that in both generative and discriminative approaches the concrete assignment $\boldsymbol{E}$ to evidence variables E is known and can be directly plugged into feature $f_\alpha(X_\alpha, E_\alpha)$, resulting in a log-linear model over X with features $f_\alpha^{\boldsymbol{E}}(X_\alpha) \equiv f_\alpha(X_\alpha, \boldsymbol{E}_\alpha)$.

   At *training time,* however, inference complexity of the two approaches is very different. Given a fixed set of weights w, for a generative model it is sufficient to perform inference once to compute the log-likelihood $LLH(\mathcal{D} \mid w)$ and its gradient, because neither the normalization constant $Z(w)$ in (3.6) nor the feature expectations $E_{P(X_\alpha, E_\alpha|w)}[f_\alpha(X_\alpha, E_\alpha)]$ in (3.1) depend on the data. In contrast, conditional random fields require inference to be done for *every datapoint* $(\boldsymbol{X}, \boldsymbol{E}) \in \mathcal{D}$, because both the normalization constant $Z(E, w)$ in (3.9) and conditional feature expectations $\mathbb{E}_{P(X_\alpha|\boldsymbol{E}_\alpha,w)}[f_\alpha(X_\alpha, \boldsymbol{E}_\alpha)]$ in (3.11) depend on the particular values $\boldsymbol{E}$ of evidence. However, the complexity of an individual inference problem is also different for generative models and CRFs: generative models require inference in a PGM over variables (X, E), while CRFs only require inference in a model over variables X, because the values of the evidence $\boldsymbol{E}$ are fixed.

   To summarize, if exact inference is used during training, generative models require solving one inference problem of per iteration, with complexity exponential in the treewidth of a graph over (X, E) induced by the features $f_\alpha(X_\alpha, E_\alpha)$. Conditional random fields, on the other hand require solving $|\mathcal{D}|$ inference problems per iteration, but every problem only has complexity exponential in the treewidth of a graph over X induced by the features $f_\alpha^{\boldsymbol{E}}(X_\alpha) \equiv f_\alpha(X_\alpha, \boldsymbol{E}_\alpha)$.

2. **Representational complexity.** There are two key aspects related to the representation. First, given the same set of features $f_\alpha(X_\alpha, E_\alpha)$, a discriminative model will typically yield a better approximation of the conditional distribution $P(X \mid E)$ than a generative one, because the discriminative model does not have to adjust the weights to also approximate $P(E)$ well. However, in the small sample regime, as Ng and Jordan (2001) have discussed, a generative model may perform better because of smaller sample complexity: although the asymptotic accuracy of the generative model is worse, than for a discriminative one, a generative model reaches its optimal accuracy using fewer samples.

   The second important difference between the discriminative and generative approaches in terms of representational power is closely connected with the training complexity of the two techniques. Recall that training a generative model requires inference in a graphical model over (X, E) with structure defined by features $f_\alpha(X_\alpha, E_\alpha)$, while for a discriminative model, one only needs to perform inference in a "projection" of a generative model on the unknown variables X via the features $f_\alpha^{\boldsymbol{E}}(X_\alpha)$. It immediately follows that in a discriminative model one can use features that *depend on evidence E in an arbitrarily complex manner,* in particular any feature f_α may depend on arbitrarily large number of evidence variables, and adding those features *will not increase either training or test complexity* as long as the structure of direct dependencies between the variables

of X does not change. In other words, discriminative models let one use much more expressive features compared to the generative models, without any significant computational penalty. The extra freedom in choosing features often greatly simplifies the feature design.

3. **Types of supported variables.** Closely related to the question of feature complexity is the question of which kinds of variables can be handled in practice by a particular model formalism. Inference approaches for graphical models, both exact and approximate, are typically formulated for the case of finite discrete variables (e.g., Yedidia et al., 2000) or for real-valued variables with very restricted types of potentials, such as Gaussians[1] (Weiss and Freeman, 2001b). As a result, in a generative setting, both query and evidence variables have to be discrete or Gaussian.

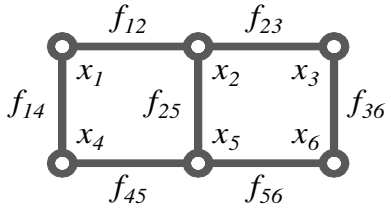
   In a discriminative setting, however, because one needs to deal only with a concrete known assignment $\boldsymbol{E}$ to the evidence variables E, and not with all possible assignments to E, it follows that *evidence variables E can be of any type,* including countably infinite discrete variables, arbitrary real-valued variables, etc. In fact, in a discriminative setting one can disregard completely the nature of the evidence variables, and consider instead the feature values $f_\alpha^{\boldsymbol{E}}(X_\alpha)$ to be the ultimate inputs to the model. As will be shown in this chapter, our evidence-specific discriminative structure learning approach also only takes into account the feature values $f_\alpha^{\boldsymbol{E}}(X_\alpha)$ corresponding to the observed evidence assignments $\boldsymbol{E}$. As a result, our approach can also handle arbitrary types of evidence variables, including countably infinite and arbitrary real-valued evidence, in a straightforward manner.

   The discriminative setting shares the restrictions on the type of query variables X with the generative setting. Because in a discriminative setting inference in the model over the query variables X is required, it follows that the query variables in both generative and discriminative settings, including our approach of models with evidence-specific structures, have to be finite discrete or Gaussian.

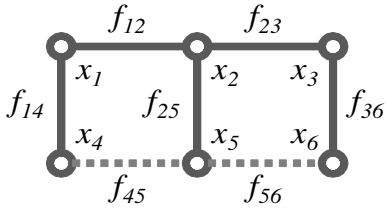
To summarize, above we have discussed how one can take advantage of the knowledge of the set of variables E that are guaranteed to be known at test time to (a) improve the approximation quality for the conditional distribution of interest $P(X \mid E)$ and (b) loosen the restrictions on the possible types of evidence variables E via discriminative parameter learning for log-linear models. There has also been work on learning the model structure in a discriminative way (see, for example, (Schmidt et al., 2007) and section 3.7 for more detailed discussion). However, existing structure learning approaches can only learn a fixed CRF structure. In other words, the information about the *identities* of the evidence variables E is taken into account during structure learning. In this chapter, we make a step further and propose an approach that takes into account not only the identities of the evidence variables, but also their *values* $\boldsymbol{E}$ to learn conditional random fields with *tractable structures specific to the concrete evidence assignment* that occurs at test time.

In chapter 2, we have shown that fixed-structure low-treewidth models can yield approximation quality competitive with high-treewidth models because the former admit exact inference and parameters learning. By adopting evidence-specific CRF structures, we further increase the representational power of low-treewidth models, without sacrificing the advantages of exact inference and optimal feature weights over high-treewidth CRFs.

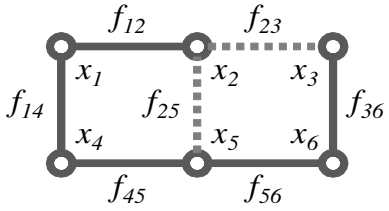
[1]There is also recent work on inference for real-valued variables in a more general setting (Song et al., 2011), but it is much less widespread in practice.



(a) The set of all features of a model forms a high-treewidth graph.



(b) For $E = \boldsymbol{E}^{(1)}$, features f_{45} and f_{56} are identically zero, resulting in a tree-structured effective model.



(c) For $E = \boldsymbol{E}^{(2)}$, features f_{25} and f_{23} are identically zero, resulting in an effective model that is also tree-structured, but the tree is different from that in Fig. (b).

Figure 3.1: An example of a high-treewidth conditional random fields with low-treewidth effective structure (c.f. Definition 42). Features that are identically zero given a particular evidence assignment are marked with dashed lines.

3.2 Evidence-specific structure for conditional random fields

Given the set $\mathcal{F}$ of features $f_\alpha(X_\alpha, E_\alpha)$ of a conditional random field (3.9), define the set $\mathbb{T}$ of edges of the CRF as

$$\mathbb{T} = \{(i-j) \mid \exists f_\alpha \in \mathcal{F} \text{ s.t. } x_i, x_j \in X_\alpha\}. \tag{3.12}$$

Observe that, given a particular evidence value $\boldsymbol{E}$, the set of edges $\mathbb{T}$ in the CRF formulation (3.9) actually can be viewed as a *supergraph* of the conditional model over X. An edge $(i-j) \in \mathbb{T}$ can be "disabled" in the following sense: if for $E = \boldsymbol{E}$ all the edge features involving $(i-j)$ are identically zero regardless of the values of x_r and x_s,

$$\forall f_\alpha \in \mathcal{F} \text{ s.t. } x_i, x_j \in X_\alpha \text{ and } \forall \mathbf{X}_\alpha \text{ it holds that } f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) = 0,$$

then

$$\sum_{f_\alpha \in \mathcal{F}} w_\alpha f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) \equiv \sum_{f_\alpha \in \mathcal{F} \text{ s.t. } \{x_i, x_j\} \not\subseteq X_\alpha} w_\alpha f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}),$$

and so *for evidence value* $\boldsymbol{E}$, the model (3.9) with edges $\mathbb{T}$ is equivalent to (3.9) with $(i-j)$ removed from $\mathbb{T}$. The following notion of *effective CRF structure,* captures the extra sparsity:

Definition 42. Given the CRF model (3.9) and evidence value $E = \boldsymbol{E}$, the effective conditional model features $\mathcal{F}(E = \boldsymbol{E})$ are those that are not identically zero:

$$\mathcal{F}(E = \boldsymbol{E}) \equiv \{f_\alpha \mid f_\alpha \in \mathcal{F} \text{ s.t. } \exists \mathbf{X}_\alpha \text{ s.t. } f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) \neq 0\}$$

and the effective structure $\mathbb{T}(E = \boldsymbol{E})$ is the set of edges corresponding to the effective features:

$$\mathbb{T}(E = \boldsymbol{E}) \equiv \mathbb{T}(\mathcal{F}(E = \boldsymbol{E})) = \{(i-j) \mid \exists f_\alpha \in \mathcal{F}(\boldsymbol{E}) \text{ s.t. } x_i, x_j \in X_\alpha\}. \tag{3.13}$$

Example. Consider the conditional random field with features shown in Fig. 3.1. To reduce clutter, we do not show graphically the dependence on the features on the evidence. Suppose the model contains pairwise features that together form a grid graph as shown in Fig. 3.1a. The treewidth of such a grid

graph is 2: edges $(x_2 - x_4)$ and $(x_2 - x_6)$ yield an example triangulation with maximum clique size of 3. However, suppose that for $E = \boldsymbol{E}^{(1)}$, features f_{45} and f_{56} are identically zero. Then those features can be removed from the model, resulting in a structure in Fig. 3.1b. One can see that the effective structure corresponding to evidence value $\boldsymbol{E}^{(1)}$ is a tree and has treewidth of 1. Similarly, for $E = \boldsymbol{E}^{(2)}$ Fig. 3.1c also shows an effective structure that is a tree, but *a different tree* from Fig. 3.1b. Suppose the training dataset $\mathcal{D}$ is such that for every datapoint $(\boldsymbol{X}, \boldsymbol{E})$ it holds that either $\boldsymbol{E} = \boldsymbol{E}^{(1)}$ or $\boldsymbol{E} = \boldsymbol{E}^{(2)}$. Then the discriminative weights learning can be done with inference cost of $|\mathcal{D}| \exp(O(1))$ instead of $|\mathcal{D}| \exp(O(2))$, where $\exp(O(1))$ is the complexity of exact inference in tree-structured models (i.e., with treewidth 1) and $\exp(O(2))$ is the complexity of exact inference in a grid-structured model with treewidth 2. The same reasoning applies to exact inference complexity at test time.

In general, the notion of effective structure is important, because it is the treewidth of effective structure that determines inference complexity given the particular evidence assignment $\boldsymbol{E}$. In particular, if $\mathbb{T}(E)$ has low treewidth for all values $\boldsymbol{E}$ of E, then inference and parameter learning using the effective structure are tractable, even if *a priori* structure $\mathbb{T}$ has high treewidth. Unfortunately, in practice the treewidth of $\mathbb{T}(E)$ is usually not much smaller than the treewidth of $\mathbb{T}$. Low-treewidth effective structures are rarely used, because treewidth is a *global* property of the graph (even *computing* treewidth is NP-complete Arnborg et al. (1987)), while feature design is a *local* process. In fact, it is the ability to learn optimal weights for a set of mutually correlated features without first understanding the inter-feature dependencies that is the key advantage of conditional random fields over other probabilistic graphical model formulations. Achieving low treewidth for the effective structures requires elaborate feature design, making model construction very difficult. Instead, in this work, we separate construction of low-treewidth effective structures from feature design and weight learning, to combine the advantages of exact inference and discriminative weights learning, high expressive power of high-treewidth models, and local feature design.

Observe that the CRF definition (3.9) can be written equivalently as

$$P(X \mid E, w) = \frac{1}{Z(E, w)} \exp\left\{\sum_{\alpha} w_\alpha \times (\mathcal{I}(f_\alpha \in \mathcal{F}(E)) \cdot f_\alpha(X_\alpha, E))\right\}. \tag{3.14}$$

Even though (3.9) and (3.14) are equivalent, in (3.14) the structure of the model is explicitly encoded as multiplicative component of the features. In addition to the feature values f, the set of effective features and the corresponding effective structure of the model are now controlled by the indicator functions $\mathcal{I}(\cdot)$. These indicator functions provide us with a way to control the treewidth of the effective structures independently of the features.

Traditionally, it has been assumed that the effective feature set $\mathcal{F}(E)$ is defined implicitly as in Def. 42: every feature that is not identically zero for the given evidence value $\boldsymbol{E}$ is included in the conditional model. However, such an assumption is not the only one possible. Here, we propose to add another level of "filtering" that, given the evidence assignment $\boldsymbol{E}$, would remove some of the nonzero features from $\mathcal{F}(E)$ so as to obtain a low-treewidth model. To achieve good approximation accuracy, such a filter cannot be arbitrary. Intuitively, we will aim to find a low-treewidth "backbone" of the most important features for evidence $\boldsymbol{E}$ and discard the rest.

More formally, we will assume that the effective feature set $\mathcal{F}(E)$ is determined by some algorithm that takes the value of E and parameters u as input. Denote $\mathcal{F}(E, u)$ to be the resulting effective features. Different approaches can be used to determine which features to retain in $\mathcal{F}(E, u)$, we will make the notion of the feature selecting algorithm more concrete shortly. For now, let us leave the algorithm $\mathcal{F}(\cdot, \cdot)$ a parameter of the model:

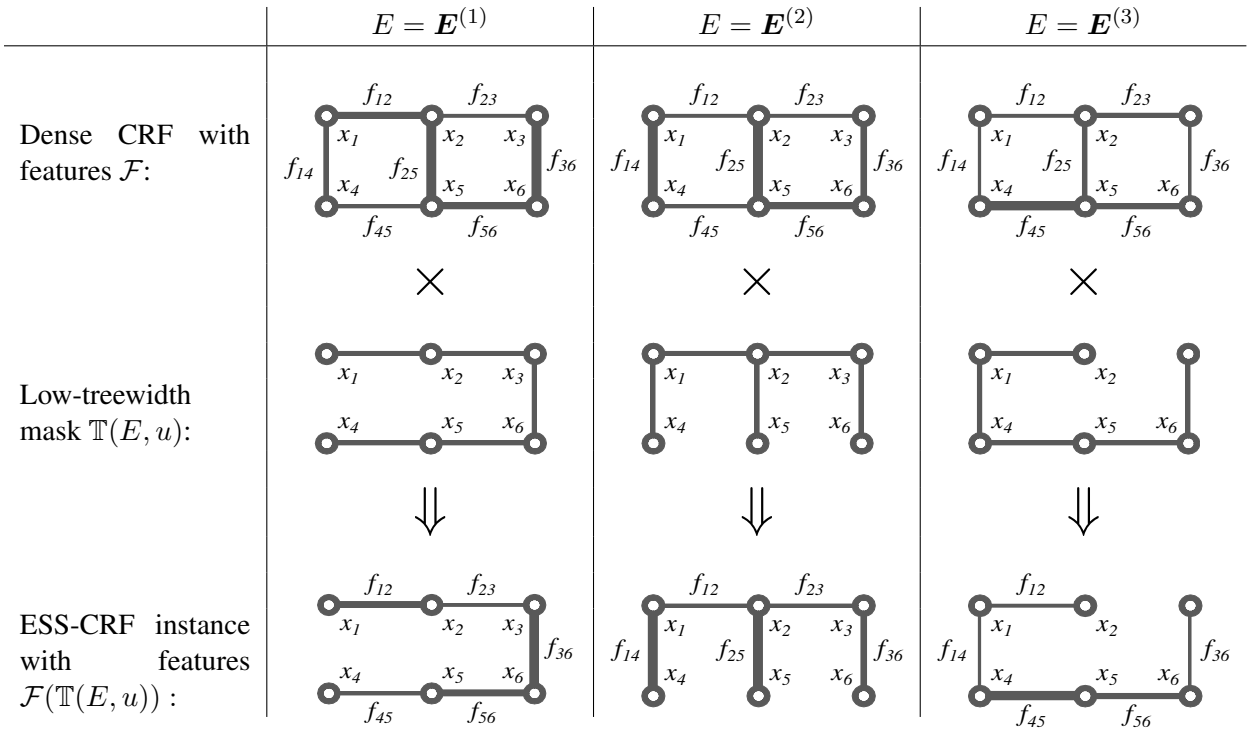


Figure 3.2: An example of ESS-CRF workflow (Alg. 3.2).

Definition 43. Given a set of query variables X, evidence variables E and feature selection algorithm $\mathcal{F}(E, u)$ parametrized by parameters u and feature weights w, a **conditional random field with evidence-specific structure (ESS-CRF)** defines a conditional distribution $P(X \mid E, w, u)$ as follows:

$$P(X \mid E, w, u) = \frac{1}{Z(E, w, u)} \exp\left\{ \sum_{f_\alpha \in \mathcal{F}} w_\alpha \times (\mathcal{I}(f_\alpha \in \mathcal{F}(E, u)) \cdot f_\alpha(X_\alpha, E)) \right\}. \tag{3.15}$$

ESS-CRFs have an important advantage over the traditional parametrization: in (3.15) the parameters u that determine the model structure are decoupled from the feature weights w. As a result, the problem of structure learning (i.e., optimizing u) can be decoupled from feature selection (choosing f) and feature weights learning (optimizing w). Such a decoupling makes it much easier to guarantee that the effective structure of the model has low treewidth by relegating all the necessary global computation to the feature selection algorithm $\mathcal{F}(E, u)$. For any fixed choice of a feature selection algorithm $\mathcal{F}(\cdot, \cdot)$ and structure parameters u, as long as $\mathbb{T}(\mathcal{F}(E, u))$ is guaranteed to have low treewidth for any evidence value, learning optimal feature weights w^* and inference at test time can be done exactly, because Fact 41 directly extends to feature weights w in ESS-CRFs:

Observation 44. Conditional log-likelihood $\log P(X \mid E, w, u)$ of ESS-CRFs (3.15) is concave in w. Also,

$$\frac{\partial \log P(\boldsymbol{X} \mid \boldsymbol{E}, w, u)}{\partial w_\alpha} = \mathcal{I}(f_\alpha \in \mathcal{F}(\boldsymbol{E}, u)) \left(f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) - \mathbb{E}_{P(X_\alpha \mid \boldsymbol{E}, w, u)} \left[f_\alpha(X_\alpha, \boldsymbol{E}) \right] \right). \tag{3.16}$$

To summarize, instead of the standard CRF workflow (Alg. 3.1), we propose ESS-CRFs (Alg. 3.2). Key to our proposal is the requirement that the feature selection algorithm $\mathcal{F}(E, u)$ **always** returns feature sets

Algorithm 3.1: Standard CRF approach

1 Define features $f_\alpha(X_\alpha, E)$, implicitly defining the **high-treewidth** CRF structure $\mathbb{T}$.
2 Optimize weights w to maximize conditional LLH (3.10) of the training data.
Use **approximate inference** to compute CLLH objective (3.10) and gradient (3.11).
3 **foreach** $\boldsymbol{E}$ *in test data* **do**
4 Use conditional model (3.9) to define the conditional distribution $P(X \mid \boldsymbol{E}, w)$.
Use **approximate inference** to compute the marginals or the most likely assignment to X.

Algorithm 3.2: CRF with evidence-specific structures approach

1 Define features $f_\alpha(X, E)$.
Choose feature selection alg. $\mathcal{F}(E, u)$ that is guaranteed to return feature sets with low treewidth of induced structures $\mathbb{T}(\mathcal{F}(E, u))$.
2 Define or learn from data parameters u for the feature selection algorithm $\mathcal{F}(\cdot, \cdot)$.
3 Optimize weights w to maximize conditional LLH $\log P(\boldsymbol{X} \mid \boldsymbol{E}, u, w)$ of the training data.
Use **exact inference** to compute CLLH objective (3.10) and gradient (3.11).
4 **foreach** $\boldsymbol{E}$ *in test data* **do**
5 Use conditional model (3.15) to define the conditional distribution $P(X \mid \boldsymbol{E}, w, u)$.
Use **exact inference** to compute the marginals or the most likely assignment to X.

with **low treewidth of induced structures** $\mathbb{T}(\mathcal{F}(E, u))$. Then, in contrast with the standard approach that has approximations (with little, if any, guarantees on the result quality) at every stage (lines 1,2,4), in our ESS-CRF approach only feature selection (line 1) involves an approximation.

Next, we will describe a general framework that allows how a wide range of existing structure learning algorithms to be adapted in a straightforward manner to perform feature selection $\mathcal{F}(E, u)$. We will demonstrate the general approach on a concrete example of Chow-Liu algorithm (Chow and Liu, 1968), a simple yet efficient algorithm that is guaranteed to learn the most likely tree structure in the generative case.

3.3 Learning tractable evidence-specific structures

Observe that selecting the most important features $\mathcal{F}(E, u)$ given the evidence is essentially a structure learning problem. Although in Def. 42 the evidence-specific structure set $\mathcal{F}(E, u)$ is the primary object and the evidence-specific structure $\mathbb{T}(E, u)$ is defined in terms of $\mathcal{F}(E, u)$, it is easy to reverse the causality. One can define the evidence-specific set of edges $\mathbb{T}(E, u)$ as the basis and obtain the corresponding set of features as

$$\mathcal{F}(\mathbb{T}) \equiv \{f_\alpha \mid f_\alpha \in \mathcal{F} \text{ and } \forall x_i, x_j \in X_\alpha \text{ it holds that } (i - j) \in \mathbb{T}\}. \tag{3.17}$$

A straightforward link with the properties in Def. 42 follows:
Observation 45. For any set of features $\mathcal{F}$ over (X, E) and set of edges $\mathbb{T}$ over X, for functions $\mathcal{F}(\mathbb{T})$ defined in (3.17) and $\mathbb{T}(\mathcal{F})$ defined in (3.13) it holds that

$$\forall \mathcal{F}' \subseteq \mathcal{F} \text{ it holds that } \mathcal{F}' \subseteq \mathcal{F}(\mathbb{T}(\mathcal{F}')). \tag{3.18}$$

Proof. From (3.13), we have

$$\forall f_\alpha(X_\alpha, E) \in \mathcal{F}' \; \forall x_i, x_j \in X_\alpha \text{ it holds that } (i-j) \in \mathbb{T}(\mathcal{F}'),$$

and therefore from (3.17) we have $f_\alpha \in \mathcal{F}(\mathbb{T})$. Also, it is not necessary for $\mathcal{F}'$ and $\mathcal{F}(\mathbb{T}(\mathcal{F}'))$ to be the same: there may be $f_\beta \in \mathcal{F} \setminus \mathcal{F}'$ such that $\forall x_i, x_j \in X_\beta \; \exists f_\alpha \in \mathcal{F}'$ s.t. $x_i, x_j \in X_\alpha$. Then $f_\beta \in \mathcal{F}(\mathbb{T}(\mathcal{F}'))$ and $\mathcal{F}' \subset \mathcal{F}(\mathbb{T}(\mathcal{F}'))$.□

From (3.17) and Observation (45), one can see that the problem of optimal evidence-specific feature selection $\mathcal{F}(E, u)$ can be formulated as a problem of learning the optimal evidence-specific structure $\mathbb{T}(E, u)$. As we have discussed in section 2.5, learning high quality structures for probabilistic graphical models is provably hard even in the generative case, that is, when $E = \emptyset$. It follows that (a) the evidence-specific structure selection problem is also hard and (b) it therefore desirable to take advantage on the existing research and state of the art structure learning approaches instead of designing specialized algorithms for evidence-specific feature selection from scratch. Thus, we adopt selecting the evidence-specific structure $\mathbb{T}(E, u)$ as the primary problem and will use (45) to select the corresponding features.

Fortunately, most algorithms for learning low-treewidth generative PGMs in the generative setting can be adapted to the problem of learning evidence-specific structure $\mathbb{T}(E, u)$ in a quite straightforward manner. Such an adaptation is possible because of a common property shared by most of the low-treewidth structure learning algorithms: the only information about the joint distribution $P(X)$ these approaches rely on is a set of marginal distributions $P(X_\gamma)$ for *small subsets* X_γ *of* X, where *small* means the size of the subsets is $|X_\gamma| = O(k)$ for treewidth k. For example, the approach of chapter 2 of this thesis only requires entropies for subsets of size $|X_\gamma| \leq 2k+2$ to compute conditional mutual information values, Karger and Srebro (2001) only require the entropies for candidate cliques with $|X_\gamma| \leq k+1$, approaches of Chow and Liu (1968) and Shahaf et al. (2009) only require the pairwise and single-variable entropies.

The problem of approximating conditional distributions $P(X_\gamma \mid E)$ for small $|X_\gamma|$ is relatively easy in the case of discrete variables, because the number of possible joint assignments $\mathbf{X}_\gamma$ to X_γ is also small ($r^{|X_\gamma|}$ for variable cardinality r) and therefore one can treat X_γ as a single variable x_γ with a state space equal to a Cartesian product of state spaces of individual variables $x_i \in X_\gamma$. Moreover, the problem of conditional density estimation $P(x_\gamma \mid E)$ for a single variable X_γ is one of the fundamental problems in machine learning, with a number of efficient high-quality solutions available (c.f., for example, Bishop, 2007; Härdle et al., 2004).

From the observations that (a) approaches for low-treewidth structure learning in the generative case rely only on low-dimensional marginals $P(X_\gamma)$ and (b) the problem of estimating conditional distributions $P(X_\gamma \mid E)$ is well-studied with a variety of high-quality approaches available, we arrive at a straightforward approach of evidence-specific structure learning. Namely, one first learns the conditional density estimators $\widehat{P}(X_\gamma \mid E)$ for every small subset $X_\gamma \subset X$ that may be considered by a structure learning algorithm. Then, given a particular assignment $\boldsymbol{E}$ of the evidence variables E, one computes the conditional density estimates $\widehat{P}(X_\gamma \mid \boldsymbol{E})$ and runs the generative structure learning algorithm with $\widehat{P}(X_\gamma \mid \boldsymbol{E})$ in place of the marginal distributions $P(X_\gamma)$. This general framework is summarized in Alg. 3.3 and 3.4.

Observe that for the purposes of constructing the low-dimensional conditional density estimators $\widehat{P}(X_\gamma \mid E)$ in Alg. 3.3 and 3.4, the evidence E is only available via feature values $\mathcal{F}_\gamma(X_\gamma, E)$, where $\mathcal{F}_\gamma$ is the subset of $\mathcal{F}$ that is "fully covered" by X_γ (c.f. lines 2, 4 and 5 of Alg. 3.3). The choice of only allowing the low-dimensional estimators to use the features included in the full ESS-CRF (3.15) is motivated not

Algorithm 3.3: Learning low-treewidth evidence-specific structures: training

Input: Training data $\mathcal{D}_{train}$, low-treewidth structure learning algorithm $\mathcal{A}_S$, conditional density estimation algorithm $\mathcal{A}_D$, feature set $\mathcal{F}$.

1 **foreach** X_γ *that may be used by* $\mathcal{A}_S$ **do**

2 $\quad \mathcal{F}_\gamma = \{f_\alpha \in \mathcal{F} \mid X_\alpha \subseteq X_\gamma\}$ *// The subset of features relevant to* X_γ

3 $\quad \mathcal{F}_\gamma(\cdot, E) = \{f_\alpha(\cdot, E) \mid f_\alpha \in \mathcal{F}_\gamma\}$

4 $\quad \mathcal{D}_\gamma = \cup_{(\mathbf{X}_\gamma, \boldsymbol{E}) \in \mathcal{D}_{train}} (\mathbf{X}_\gamma, \mathcal{F}_\gamma(\cdot, \boldsymbol{E}))$

5 $\quad u_\gamma \leftarrow$ parameters from training the estimator of $P(X_\gamma \mid \mathcal{F}_\gamma(\cdot, E))$ using $\mathcal{A}_D$ on dataset $\mathcal{D}_\gamma$

6 **return** conditional density parameters $u = \cup_\gamma \{u_\gamma\}$

Algorithm 3.4: Learning low-treewidth evidence-specific structures: test time

Input: Evidence assignment $\boldsymbol{E}$, low-treewidth structure learning algorithm $\mathcal{A}_S$, conditional density estimation algorithm $\mathcal{A}_D$, conditional density parameters u.

1 **foreach** X_γ *that may be used by* $\mathcal{A}_S$ **do**

2 $\quad \widehat{P}(X_\gamma \mid \boldsymbol{E}) \leftarrow \mathcal{A}_D(\mathcal{F}_\gamma(\boldsymbol{E}), u_\gamma)$ *//* $\widehat{P}(X_\gamma \mid \boldsymbol{E})$ *is only a function of* X_γ*, because* $\boldsymbol{E}$ *is fixed*

3 $\mathbb{T} \leftarrow \mathcal{A}_S(\widehat{P}(\cdot \mid \boldsymbol{E}))$ *// Use* $\widehat{P}(X_\gamma \mid \boldsymbol{E})$ *whenever* $\mathcal{A}_S$ *needs a marginal* $P(X_\gamma)$

4 **return** $\mathcal{F}(\mathbb{T})$ per the equation (3.17)

simply by the convenience concerns. Such a reuse of features also helps avoid including features with poor predictive power in the full model (3.15). If the features $\mathcal{F}_\gamma$ do not contain enough information to predict X_γ well, then typically

1. The entropy of the conditional distribution $P(X_\gamma \mid \mathcal{F}_\gamma(E))$ is large, and correspondingly likelihood-based clique scores for X_γ used by score-based structure learning approaches will be low.

2. Features $\mathcal{F}_\gamma$ are also not useful in predicting the full distribution $P(X)$, because they can only affect $P(X)$ via variables X_γ.

Therefore, in situations when in principle the evidence E contains enough information to predict the values of X_γ accurately, but features $\mathcal{F}\gamma$ are not informative, using a local conditional density estimator $\widehat{P}(X_\gamma \mid E)$ that uses a better set of features internally would lead to overestimation of the importance of $\mathcal{F}_\gamma$ in the model (3.15). Correspondingly, only using $\mathcal{F}_\gamma$ for estimating $\widehat{P}(X_\gamma \mid E)$ eliminates this source of error.

Example. The process of constructing evidence-specific conditional random fields during test time, summarized in Alg. 3.2, and with Alg. 3.4 as a feature selection procedure, is illustrated in Fig. 3.2. Every column corresponds to construction of an ESS-CRF instance for a certain assignment $\boldsymbol{E}$ of evidence variables. In the top row, the dense structure of a log-linear model with all the weighted features $\mathcal{F}$. Edge thickness encodes the strength of local dependence induced by the individual features. In the middle row are low-treewidth evidence-specific structures $\mathbb{T}(\boldsymbol{E}, u)$ obtained on line 3 of Alg. 3.4. Finally, in the bottom row are the resulting tractable evidence-specific models (3.15). One can see that the resulting models are obtained by using the tractable structures $\mathbb{T}(\boldsymbol{E}, u)$ as "masks" to choose the features $\mathcal{F}(\mathbb{T}(\boldsymbol{E}, u))$ per the equation (3.17). Notice that the thickness of the retained edges in the bottom row is the same as in the top row, which illustrates the fact that the feature selection procedure, encoded with indicator functions in (3.15), does not adjust the feature weights.

3.3.1 Evidence-specific Chow-Liu algorithm

Here, we illustrate the general approach of evidence-specific feature selection in Alg. 3.3 and 3.4 with a concrete example, where the conditional density estimation algorithm $\mathcal{A}_D$ is logistic regression (see, e.g., Bishop (2007) and the structure learning algorithm $\mathcal{A}_S$ is one of Chow and Liu (1968). As we discussed in section 1.1.1, the problem of learning optimal structure for low-treewidth models is intractable for most formulations, but the most likely trees (i.e., models of treewidth 1) can be learned efficiently using Chow-Liu algorithm.

For a fixed set of edges $\mathbb{T}$ that defines a tree over variables X, it follows from Lemma 8 that out of the distributions that factorize as $P(X) \equiv \frac{1}{Z}\prod_{(i,j)\in\mathbb{T}}\psi_{ij}(x_i,x_j)$, the one that maximizes the likelihood (equivalently, minimizes the KL divergence from the empirical distribution $P_{\mathcal{D}}(X)$) is the projection of $P_{\mathcal{D}}(X)$ on $\mathbb{T}$:

$$P_{\mathbb{T},\mathcal{D}}(X) = \prod_{x_i\in X} P_{\mathcal{D}}(x_i) \prod_{(i-j)\in\mathbb{T}} \frac{P_{\mathcal{D}}(x_i,x_j)}{P_{\mathcal{D}}(x_i)P_{\mathcal{D}}(x_j)} \tag{3.19}$$

and the corresponding log-likelihood is

$$\begin{aligned} LLH(\mathbb{T} \mid \mathcal{D}) &= |\mathcal{D}| \sum_{\boldsymbol{X}} P_{\mathcal{D}}(\boldsymbol{X}) \log\left(\prod_{x_i\in X} P_{\mathcal{D}}(\mathbf{x_i}) \prod_{(i-j)\in\mathbb{T}} \frac{P_{\mathcal{D}}(\mathbf{x_i},\mathbf{x_j})}{P_{\mathcal{D}}(\mathbf{x_i})P_{\mathcal{D}}(\mathbf{x_j})}\right) \\ &= |\mathcal{D}|\left(-\sum_{x_i\in X} H_{\mathcal{D}}(x_i) + \sum_{(i-j)\in\mathbb{T}} (H_{\mathcal{D}}(x_i)+H_{\mathcal{D}}(x_j)-H_{\mathcal{D}}(x_i,x_j))\right) \\ &= |\mathcal{D}|\left(-\sum_{x_i\in X} H_{\mathcal{D}}(x_i) + \sum_{(i-j)\in\mathbb{T}} I_{\mathcal{D}}(x_i,x_j)\right), \end{aligned} \tag{3.20}$$

where $I_{\mathcal{D}}(x_i,x_j)$ is the mutual information between x_i and x_j corresponding to the empirical distribution $P_{\mathcal{D}}(X)$. One can see that the tree structure $\mathbb{T}$ only affects the model likelihood via the component $\sum_{(i-j)\in\mathbb{T}} I_{\mathcal{D}}(x_i,x_j)$, and therefore to obtain the most likely structure, Chow-Liu algorithm finds the maximum spanning tree of a fully connected graph over X where an edge $(i-j)$ has weight $I_{\mathcal{D}}(x_i,x_j)$. Maximum spanning tree can be found in $O(|X|^2)$ using Prim's algorithm, resulting in an efficient structure learning algorithm.

Having fixed the low-treewidth structure learning algorithm $\mathcal{A}_S$ in Alg. 3.3 and 3.4 to be the Chow-Liu algorithm, we now only need to choose a concrete conditional density estimation approach $\mathcal{A}_D$. We use logistic regression, where the conditional distribution $P(x_i,x_j \mid E)$ is defined as

$$\widehat{P}(x_i,x_j \mid E, u_{ij}) = \frac{1}{Z(E,u_{ij})} \exp\left\{\sum_{f_\alpha\in\mathcal{F}_{ij}} u_{ij,\alpha} f_\alpha(X_\alpha, E)\right\}. \tag{3.21}$$

Observe that the set of features $\mathcal{F}_{ij}$ is a subset of the features $\mathcal{F}$ of the full ESS-CRF model (3.15) as defined on line 2 of Alg. 3.3. Essentially, a logistic regression model is a small CRF over only two variables. It follows that optimal weights $\boldsymbol{u}^*_{ij}$ for the logistic regression conditional density estimator can be found efficiently using the same standard convex optimization techniques, for example L-BFGS (Liu and Nocedal, 1989), that are used for parameter learning in structured log-linear models.

It is important to notice that the conditional density estimates $\widehat{P}(x_i, x_j \mid E, u_{ij})$ obtained via logistic regression (3.21) do not fully satisfy the assumptions behind the Chow-Liu algorithm. The problem is that two pairwise conditional distributions $\widehat{P}(x_i, x_j \mid E, u_{ij})$ and $\widehat{P}(x_i, x_k \mid E, u_{ik})$ may not agree on the single-variable conditional: in general it may be that

$$\sum_{\mathbf{x_j}} \widehat{P}(x_i, \mathbf{x_j} \mid \boldsymbol{E}, u_{ij}) \neq \sum_{\mathbf{x_k}} \widehat{P}(x_i, \mathbf{x_k} \mid \boldsymbol{E}, u_{ik}). \tag{3.22}$$

When the pairwise conditionals disagree on the single variable conditionals as in (3.22), replacing the marginal distributions in (3.19) with the corresponding approximate conditionals to get

$$\overline{P}(X \mid E) \propto \prod_{x_i \in X} \widehat{P}(x_i \mid E, u_i) \prod_{(i-j)\in\mathbb{T}} \frac{\widehat{P}(x_i, x_j \mid E, u_{ij})}{\widehat{P}(x_i \mid E, u_i)\widehat{P}(x_j \mid E, u_j)}, \tag{3.23}$$

we obtain an approximate distribution $\overline{P}(X \mid E)$ that in general does not agree with the approximations $\widehat{P}(\cdot \mid \cdot)$:

$$\overline{P}(x_i, x_j \mid E) \neq \widehat{P}(x_i, x_j \mid E, u_{ij}).$$

It follows in such a case that the log-likelihood (3.20) for $\overline{P}(X \mid E)$ does not decompose into a sum of mutual informations of local approximations $\widehat{P}(x_i, x_j \mid E, u_{ij})$, and the tree $\mathbb{T}$ may only be approximately optimal. We have found in practice, however, that even without compensating for such scoring errors, evidence-specific Chow-Liu algorithm finds high-quality structures.

Other approaches for estimating local conditionals $\widehat{P}(x_i, x_j \mid E, u_{ij})$, such as kernel density estimators (Härdle et al., 2004), may be guaranteed (depending on the available features) to return conditionals that agree on the single-variable marginals with each other. For such mutually consistent conditional estimates the score (3.20) can be computed exactly from the pairwise density estimates and extracting the maximum spanning tree from edges weighted by mutual information would return an optimal structure. Here, we chose logistic regression as the low-dimensional conditional density estimator because of its low complexity at the test time, compared to kernel-based approaches. The optimal choice of both the low-dimensional conditional density estimation approach and the low-treewidth structure learning algorithm would depend on the properties of the dataset in question and on the desired balance of accuracy versus efficiency of the concrete application.

The full ESS-CRF approach with evidence-specific Chow-Liu algorithm for feature selection is summarized in Alg. 3.5 (training) and Alg. 3.6 (testing). In section 3.6, we investigate the performance of Alg. 3.5 and Alg. 3.6 empirically and demonstrate that such an instance of ESS-CRF is able to match the approximation accuracy of high-treewidth models, while having much lower computational cost.

3.4 Relational CRFs with evidence-specific structure

Traditional (also called propositional) PGMs are not well suited for dealing with relational data, where every variable is an *entity* of some *type*, and entities are related to each other via different types of links. Usually, there are relatively few entity types and link types. For example, the webpages on the Internet are linked via hyperlinks, and social networks link people via friendship relationships. Relational data violates the *i.i.d.* data assumption of traditional PGMs, and huge dimensionality of relational datasets preclude learning meaningful propositional models. Instead, several formulations of *relational PGMs*

Algorithm 3.5: ESS-CRF with conditional Chow-Liu algorithm for feature selection: training

Input: Training data $\mathcal{D}$, set of features $\mathcal{F}$.

foreach $x_i, x_j \in X$ **do** *// Training the low-dimensional conditional density estimators*

  define $\widehat{P}(x_i, x_j \mid E, u_{ij}) = \frac{1}{Z(E,u_{ij})} \exp\left\{ \sum_{f_\alpha \in \mathcal{F}_{ij}} u_{ij,\alpha} f_\alpha(X_\alpha, E) \right\}$ *// Equation* (3.21)

  $\boldsymbol{u}^*_{ij} \leftarrow \arg\max_{u_{ij}} \sum_{(\boldsymbol{X},\boldsymbol{E})\in\mathcal{D}} \log \widehat{P}(\mathbf{x_i}, \mathbf{x_j} \mid \boldsymbol{E}, u_{ij})$ *// Convex problem, optimize with L-BFGS*

foreach $(\boldsymbol{X}, \boldsymbol{E}) \in \mathcal{D}$ **do** *// Selecting evidence-specific structures for the training data*

  foreach $x_i, x_j \in X$ **do**

    $\boldsymbol{v}_{ij} \leftarrow I_{\widehat{P}(x_i,x_j|\boldsymbol{E},\boldsymbol{u}^*_{ij})}(x_i, x_j)$

  $\mathbb{T}(\boldsymbol{E}, \boldsymbol{u}^*) \leftarrow \text{MaxSpanningTree}(\cup_{x_i,x_j\in X}\{\boldsymbol{v}_{ij}\})$

  $\mathcal{F}(\boldsymbol{E}, \boldsymbol{u}^*) = \{f_\alpha \mid f_\alpha \in \mathcal{F}, \forall x_i, x_j \in X_\alpha \text{ it holds that } (i-j) \in \mathbb{T}(\boldsymbol{E}, \boldsymbol{u}^*)\}$ *// Eqn.* (3.17)

// Below, use equation (3.15)*. Tractable inference, convex problem, optimize with L-BFGS*

$\mathbf{w}^* \leftarrow \arg\max_{w} \sum_{(\boldsymbol{X},\boldsymbol{E})\in\mathcal{D}} \frac{1}{Z(\boldsymbol{E},w,\boldsymbol{u}^*)} \exp\left\{ \sum_{f_\alpha \in \mathcal{F}(\boldsymbol{E},\boldsymbol{u}^*)} w_\alpha f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) \right\}$

return $(\mathbf{w}^*, \boldsymbol{u}^*)$

Algorithm 3.6: ESS-CRF with conditional Chow-Liu algorithm for feature selection: test time

Input: Evidence assignment $\boldsymbol{E}$, set of features $\mathcal{F}$, logistic regression parameters $\boldsymbol{u}$, ESS-CRF feature weights $\mathbf{w}$.

foreach $x_i, x_j \in X$ **do**

  $\boldsymbol{v}_{ij} \leftarrow I_{\widehat{P}(x_i,x_j|\boldsymbol{E},\boldsymbol{u}_{ij})}(x_i, x_j)$

$\mathbb{T}(\boldsymbol{E}, \boldsymbol{u}) \leftarrow \text{MaxSpanningTree}(\cup_{x_i,x_j\in X}\{\boldsymbol{v}_{ij}\})$

$\mathcal{F}(\boldsymbol{E}, \boldsymbol{u}) = \{f_\alpha \mid f_\alpha \in \mathcal{F}, \forall x_i, x_j \in X_\alpha \text{ it holds that } (i-j) \in \mathbb{T}(\boldsymbol{E}, \boldsymbol{u})\}$ *// Equation* (3.17)

return $P(X \mid \boldsymbol{E}, \mathbf{w}, \boldsymbol{u}) = \frac{1}{Z(\boldsymbol{E},\mathbf{w},\boldsymbol{u})} \exp\left\{ \sum_{f_\alpha \in \mathcal{F}(\boldsymbol{E},\boldsymbol{u})} \mathbf{w}_\alpha f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) \right\}$

have been proposed (Friedman et al., 1999; Richardson and Domingos, 2006; Taskar et al., 2002; Getoor and Taskar, 2007) to work with relational data, including relational CRFs. The key property of all these formulations is that the model is defined using a few *template potentials* defined on the abstract level of *variable types* and replicated as necessary for concrete entities.

More concretely, in relational CRFs every variable x_i is assigned a type m_i out of the set $\mathcal{M}$ of possible types. A binary relation $\boldsymbol{R} \in \mathbb{R}$, corresponding to a specific type of link between two variables, specifies the types of its input arguments, and a feature $f_{\boldsymbol{R}}(\cdot, E)$ with feature weight $w_{\boldsymbol{R}}$. We will say that variables write $X_\alpha \in \text{inst}(\boldsymbol{R}, X)$ and say that X_α forms a *grounding* or *instance* of a relation $\boldsymbol{R}$ if the types of X_α match the input types specified by $\boldsymbol{R}$. The conditional distribution $P(X \mid E)$ is then generalized from the propositional CRF (3.9) by copying the template potentials for every grounding of every relation:

$$P(X \mid E, \mathbb{R}, w) = \frac{1}{Z(E, w)} \exp\left\{ \sum_{\boldsymbol{R}\in\mathbb{R}} w_{\boldsymbol{R}} \cdot \left(\sum_{X_\alpha \in \text{inst}(\boldsymbol{R},X)} f_{\boldsymbol{R}}(X_\alpha, E) \right) \right\} \tag{3.24}$$

Observe that the only meaningful difference of the relational CRF (3.24) from the propositional formulation (3.9) is that the former shares the same parameters between different edges. In (3.24), parameter sharing is emphasized by taking the feature weight $w_{\boldsymbol{R}}$ outside of the sum of the features corresponding to all the groundings of $\boldsymbol{R}$. By accounting for parameter sharing, it is straightforward to adapt our ESS-CRF formulation to the relational setting. We define the relational ESS-CRF conditional distribution as

$$P(X \mid E, \mathbb{R}, w, u) = \frac{1}{Z(E, w, u)} \exp\left\{\sum_{\boldsymbol{R} \in \mathbb{R}} w_{\boldsymbol{R}} \cdot \left(\sum_{X_\alpha \in \text{inst}(\boldsymbol{R}, X)} \mathcal{I}(X_\alpha \in \mathcal{F}(\mathbb{T}(E, u))) \cdot f_{\boldsymbol{R}}(X_\alpha, E)\right)\right\}, \tag{3.25}$$

where the set of edges $\mathbb{T}(E, u)$ encodes the evidence-specific structure over X and $\mathcal{F}(\mathbb{T}(\cdot, \cdot))$ is the set of *grounded* features matching the structure $\mathbb{T}$ as defined in (3.17). Given the structure learning algorithm $\mathbb{T}(\cdot, \cdot)$ that is guaranteed to return low-treewidth structures *for the grounded model*, one can learn optimal feature weights w^* and perform inference at test time exactly:

Observation 46. Relational ESS-CRF log-likelihood is concave with respect to w. Moreover,

$$\frac{\partial \log P(\boldsymbol{X} \mid \boldsymbol{E}, \mathbb{R}, w, u)}{\partial w_{\boldsymbol{R}}} = \\ = \sum_{X_\alpha \in \text{inst}(\boldsymbol{R}, X)} \mathcal{I}(X_\alpha \in \mathcal{F}(\mathbb{T}(E, u))) \cdot \left(f_{\boldsymbol{R}}(\mathbf{X}_\alpha, \boldsymbol{E}) - \mathbb{E}_{P(\cdot \mid \boldsymbol{E}, \mathbb{R}, w, u)}\left[f_{\boldsymbol{R}}(X_\alpha, \boldsymbol{E})\right]\right). \tag{3.26}$$

Consider the changes one needs to make to the evidence-specific feature selection approach of Alg. 3.3 and 3.4 to adapt them to a relational setting. Because the complexity of inference in model (3.25) and computing the gradient (3.26) is determined by the the treewidth of the *grounded* model, the structure learning algorithm $\mathcal{A}_S$ and feature selection depending on the learned structure $\mathbb{T}$ per the equation (3.17) can be done with no changes at all from the propositional case. In fact, the structure learning algorithm $\mathcal{A}_S$ does not even need to be aware of the relational nature of the model. The only component that does need modifications for the relational case is the conditional density estimator $\widehat{P}(X_\gamma \mid E, u)$, because propositional conditional density estimators cannot be directly trained on relational data.

3.4.1 Adapting low-dimensional conditional density estimation to the relational setting

Fortunately, one can adapt any propositional conditional density estimator $\widehat{P}(X_\gamma \mid E, u)$ to a relational setting by (a) sharing the parameters between estimators for conditionals with the same variable types, in the same manner as the relational CRFs (3.25) share feature weights between different groundings of the same relation and (b) treating the values of every subset X_γ of X with appropriate variable types as a separate datapoint for the purposes of training. In other words, we will create a set of *representative propositional estimators* for every *combination of variable types* that may be required by the structure learning algorithm. Then, the conditional distribution $\widehat{P}(X_\gamma \mid E, u)$ can be approximated using the corresponding propositional estimator for the types of variables in X_γ.

Formally, let $\boldsymbol{K}(X_\gamma)$ denote the set of types of the ground variables $X_\gamma \subseteq X$ in a relational model. One can think of $\boldsymbol{K}$ as a relation with a trivial feature (identically equal to 0). Denote $|\boldsymbol{K}|$ to be the number of variables specified by $\boldsymbol{K}$ and $Y_{\boldsymbol{K}}$ to be a set of *representative variables* of $\boldsymbol{K}$. We require that $|Y_{\boldsymbol{K}}| = |\boldsymbol{K}|$ and the types of individual variables in $Y_{\boldsymbol{K}}$ match the variable types specify by $|\boldsymbol{K}|$. One can see that

$$\text{inst}(\boldsymbol{K}, Y_{\boldsymbol{K}}) = \{Y_{\boldsymbol{K}}\}.$$

Denote $y_{\boldsymbol{K},i,\gamma}$ to be the variable from $Y_{\boldsymbol{K}}$ corresponding to $x_i \in X_\gamma$. The mapping $X_\gamma \to Y_{\boldsymbol{K}} : x_i \to y_{\boldsymbol{K},i,\gamma}$ is a isomorphism. For $X_\beta \subseteq X_\gamma$ and the corresponding set $Y_{\boldsymbol{K},\beta} = \{y_{\boldsymbol{K},i,\gamma} \mid x_i \in X_\beta\}$, denote also $X_\beta(Y_{\boldsymbol{K},\beta})$ to be the function that for every variable $x_i \in X_\beta$ sets the value $x_i = y_{\boldsymbol{K},i,\gamma}$. Finally, for $X_\beta \in \text{inst}(\boldsymbol{R}, X_\gamma)$ denote

$$f_{\boldsymbol{K},\gamma,\beta}(Y_\beta, E) \equiv f_{\boldsymbol{K},\gamma,\beta}(X_\beta(Y_{\boldsymbol{K},\beta}), E). \tag{3.27}$$

Using the notation establishing the connection between the variables X of the relational model and the representative propositional variables Y, we can now formulate, in Alg. 3.7 and 3.8, the evidence-specific feature selection approach for the relational setting, based on the propositional case of Alg. 3.3 and 3.4. The are only differences of Alg. 3.7 and 3.8 from their respective propositional case counterparts:

1. Whenever an approximation of the low-dimensional conditional density distribution $\widehat{P}(X_\gamma \mid E)$ is needed, the relational variables X_γ are mapped into the corresponding representative propositional variables $Y_{\boldsymbol{K}(\gamma)}$, and correspondingly for the features (lines 7 of Alg. 3.7 and 3 of Alg. 3.8), the conditional approximation is computed for $Y_{\boldsymbol{K}(\gamma)}$ using a propositional estimator, and then mapped back to X_γ (lines 3-5 of Alg. 3.8).

2. Every possible grounding of a relation type $\boldsymbol{K}$ in X is treated as a separate datapoint during the training of the propositional conditional density estimators (c.f. the construction of the propositional dataset $\mathbb{D}[\boldsymbol{K}]$ on lines 6-9 of Alg. 3.7). Although the propositional conditional density estimation approaches typically assume independent identically distributed training datapoints, and a transformation of lines 6-9, applied to a relational dataset, will *not* in general result in independent datapoints, we have found such a procedure to work well in practice. Notice that the datapoints corresponding to X_γ such that all the features involving X_γ are identically zero are not included into the propositional training dataset to improve efficiency. In practice, there is often a large number of such subsets X_γ, and not having to process the corresponding datapoints during the conditional estimators learning brings substantial speedups.

   Example. To illustrate the training of representative conditional density estimators on a relational model, consider Fig. 3.3. In Fig. 3.3a is a grounding of a model with 5 variables X of two possible types (x_1, x_2 of type $\boldsymbol{a}$ and x_3, x_4, x_5 of type $\boldsymbol{b}$). For simplicity, the evidence is not shown. Suppose all three possible relations are present. Then the grounding would be a fully connected graph; edge colors in Fig. 3.3a encode the relation types for which the edges represent groundings. For example, relation $\boldsymbol{aa}$ has only one grounding $X_\gamma = (x_1, x_2)$, while relation $\boldsymbol{bb}$ has 3 groundings.

   For a structure learning approach that only works with pairs of variables, such as Chow-Liu algorithm, or that of Shahaf et al. (2009), one needs to be able to estimate the conditional probability $\widehat{P}(\cdot, E)$ for any pair of variables from X. Here, because the relations of the model are also pairwise, we get a one to one correspondence of a relation to a representative conditional density estimator. However, in general a single density estimator may involve several different relations. In Fig. 3.3b-3.3d are shown the propositional datasets (without the corresponding feature values) that Alg. 3.7 would form on lines 6-9 to train the respective propositional estimators. Every row of the tables is a separate datapoint. Notice that because the grounded model is assumed to be a part of the training data, the concrete values of variables $\mathbf{x_i}$ are known and can be plugged in to form the propositional datapoints in Fig. 3.3b-3.3d.

As one can see from (3.25) and Algs. 3.7 and 3.8, an important property of the relational setting is that the dimensionality of parameters (w, u) only depends on the number of relations that define the model, but does *not* depend on the number of variables X or the number of grounded features f_α. In the relational

Algorithm 3.7: Learning low-treewidth evidence-specific structures, relational case: training

Input: Training data $\mathcal{D}_{train}$, low-treewidth structure learning algorithm $\mathcal{A}_S$, conditional density estimation algorithm $\mathcal{A}_D$, relations set $\mathbb{R}$.

$\mathbb{K} = \emptyset$ *// The set of all possible variable type combinations that $\mathcal{A}_S$ may encounter*

foreach $X_\gamma \in \mathcal{D}_{train}$ *that may be used by* $\mathcal{A}_S$ **do**

  $\mathbb{K} = \mathbb{K} \cup \{\boldsymbol{K}(X_\gamma)\}$

$\mathbb{D} = \emptyset$ *// Propositional datasets to train the representative conditional density estimators*

foreach $\boldsymbol{K} \in \mathbb{K}$ **do**

  foreach $X_\gamma \in \text{inst}(\boldsymbol{K}, X_{train})$ **do**

    $\mathcal{F}_{\boldsymbol{K}} = \cup_{\boldsymbol{R} \in \mathbb{R}} \cup_{X_\beta \in \text{inst}(\boldsymbol{R}, X_\gamma)} \{f_{\boldsymbol{K},\gamma,\beta}(Y_{\boldsymbol{K},\beta}, E)\}$ *// $f_{\boldsymbol{K},\gamma,\beta}$ is defined in* (3.27)

    if $\exists f_{\boldsymbol{K},\gamma,\beta}, \boldsymbol{Y}_{\boldsymbol{K},\beta}$ *s.t.* $f_{\boldsymbol{K},\gamma,\beta}(\boldsymbol{Y}_{\boldsymbol{K},\beta}, \boldsymbol{E}) \neq 0$ **then**

      $\mathbb{D}[\boldsymbol{K}] \leftarrow \mathbb{D}[\boldsymbol{K}] \cup (Y_{\boldsymbol{K}}(\mathbf{X}_\gamma), \mathcal{F}_{\boldsymbol{K}}(\cdot, \boldsymbol{E}))$

  $u_{\boldsymbol{K}} \leftarrow$ parameters from training the estimator of $P(Y_{\boldsymbol{K}} \mid \mathcal{F}_{\boldsymbol{K}}(\cdot, E))$ with $\mathcal{A}_D$ on data $\mathbb{D}[\boldsymbol{K}]$

return conditional density parameters $u = \cup_{\boldsymbol{K}} \{u_{\boldsymbol{K}}\}$

Algorithm 3.8: Learning low-treewidth evidence-specific structures, relational case: test time

Input: Evidence assignment $\boldsymbol{E}$, low-treewidth structure learning algorithm $\mathcal{A}_S$, conditional density estimation algorithm $\mathcal{A}_D$, conditional density parameters u, relations set $\mathbb{R}$.

foreach X_γ *that may be used by* $\mathcal{A}_S$ **do**

  $\boldsymbol{K} \leftarrow \boldsymbol{K}(X_\gamma)$

  $\mathcal{F}_{\boldsymbol{K}} = \cup_{\boldsymbol{R} \in \mathbb{R}} \cup_{X_\beta \in \text{inst}(\boldsymbol{R}, X_\gamma)} \{f_{\boldsymbol{K},\gamma,\beta}(Y_{\boldsymbol{K},\beta}, E)\}$ *// Same as on line 7 of Alg. 3.7*

  $\widehat{P}(Y_{\boldsymbol{K}} \mid \boldsymbol{E}) \leftarrow \mathcal{A}_D(\mathcal{F}_{\boldsymbol{K}}(\boldsymbol{E}), u_{\boldsymbol{K}})$ *// Representative propositional estimator*

  $\widehat{P}(X_\gamma \mid \boldsymbol{E}) \leftarrow \widehat{P}(Y_{\boldsymbol{K}}(X_\gamma) \mid \boldsymbol{E})$ *// Remap the conditional back to the grounding in question*

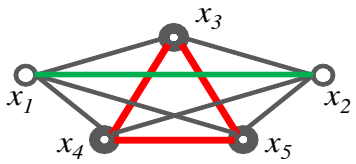
$\mathbb{T} \leftarrow \mathcal{A}_S(\widehat{P}(\cdot \mid \boldsymbol{E}))$ *// Use $\widehat{P}(X_\gamma \mid \boldsymbol{E})$ whenever $\mathcal{A}_S$ needs a marginal $P(X_\gamma)$*

return $\mathcal{F}(\mathbb{T})$ per the equation (3.17)

setting, one only needs to learn $O(|\mathbb{R}|)$ parameters, *regardless of the dataset size,* for both structure selection and feature weights, as opposed to $O(|X|^2)$ parameters for the propositional case. Thus, relational ESS-CRFs are typically much less prone to overfitting than propositional ones.

3.5 Alternative approaches for learning parameters of ESS-CRFs

Although optimizing the conditional log-likelihood of a model is a well-founded objective, it also has its limitations, and in some settings alternative objectives may be more attractive. One important limitation of the conditional log-likelihood is the computational cost (exponential in treewidth and thus infeasible in high-treewidth models). The other issue is that many applications, such as natural text processing (Lafferty et al., 2001), require models for predicting *only the most probable* assignment to the unknown variables X given the evidence, and do not care about the density estimation outside of the mode of the conditional distribution. Because conditional log-likelihood assigns equal importance to the model accuracy in both high and low probability areas, optimizing log-likelihood results in suboptimal models from the perspective of structured prediction (c.f. Taskar et al., 2003).



(a) A grounded model with all possible pairwise relations.

$y_{\boldsymbol{aa},1,\gamma}$	$y_{\boldsymbol{aa},2,\gamma}$
$\mathbf{x_1}$	$\mathbf{x_2}$

(b) Relation $\boldsymbol{K} = \boldsymbol{aa}$

$y_{\boldsymbol{ab},1,\gamma}$	$y_{\boldsymbol{ab},2,\gamma}$
$\mathbf{x_1}$	$\mathbf{x_3}$
$\mathbf{x_1}$	$\mathbf{x_4}$
$\mathbf{x_1}$	$\mathbf{x_5}$
$\mathbf{x_2}$	$\mathbf{x_3}$
$\mathbf{x_2}$	$\mathbf{x_4}$
$\mathbf{x_2}$	$\mathbf{x_5}$

(c) Relation $\boldsymbol{K} = \boldsymbol{ab}$

$y_{\boldsymbol{bb},1,\gamma}$	$y_{\boldsymbol{bb},2,\gamma}$
$\mathbf{x_3}$	$\mathbf{x_4}$
$\mathbf{x_3}$	$\mathbf{x_5}$
$\mathbf{x_5}$	$\mathbf{x_3}$

(d) Relation $\boldsymbol{K} = \boldsymbol{bb}$

Figure 3.3: An example of converting a grounding of a relational model to propositional datasets for training pairwise conditional estimators in Alg. 3.7. Variables x_1 and x_2 have type $\boldsymbol{a}$, variables x_3, x_4 and x_5 - type $\boldsymbol{b}$. In Fig. (b)-(d) are the propositional datasets (values of representative variables $y_{\boldsymbol{K},i,\gamma}$) that are formed by Alg. 3.7 to train a representative pairwise conditional density estimator for every possible combination of variable types.

For standard log-linear models, the above limitations of conditional log-likelihood have been addressed in the literature and alternative optimization objectives (and corresponding optimization techniques) have been introduced. In this section, we discuss how those alternative objectives, namely pseudolikelihood (Besag, 1974) for better computational efficiency and max-margin learning (Taskar et al., 2003) for structured prediction can be used in our framework of learning discriminative models with evidence-specific structure.

3.5.1 Pseudolikelihood learning

Computing the conditional likelihood of a log-linear model (3.9) for $P(X \mid E, w)$ requires summing over all possible values of X (to obtain the normalization constant). For variables of cardinality r, the resulting computational complexity is $O\left(r^{|X|}\right)$ for the naive computation or $O\left(r^{treewidth}\right)$ for a structured model. As a result, it is infeasible to compute the conditional log-likelihood and its gradient exactly for high-treewidth models. To improve computational efficiency, Besag (1974) have proposed *pseudo-likelihood* (1.5) as an approximation to the exact conditional log-likelihood. There are guarantees on the consistency of pseudolikelihood for parameter estimation for distributions that can be expressed exactly as a log-linear model with given features (Gidas, 1988).

The intuition behind the computational efficiency of pseudolikelihood is that after conditioning on the Markov blanket of a variable, it is not necessary to sum over all possible assignments to the Markov blanket. It is sufficient to only consider the *actual value of the Markov blanket variables observed in the data.* In fact, this is exactly the same trick that makes it possible for discriminative models to work with features that involve arbitrarily complicated dependencies on the evidence (see the discussion in Section 3.1).

For concreteness, consider the case of pairwise conditional density estimation. The goal here is to learn a model $P(x_i, x_j \mid E, u_{ij})$. One can see in (1.5) that for any feature $f_{ij}(x_i, x_j, E)$, for any fixed assignment $\mathbf{x_i} = k$ and $\mathbf{x_j} = m$ that occurs in the training data, to compute the pseudolikelihood one only needs to evaluate $f_{ij}(k, \cdot)$ and $f_{ij}(\cdot, m)$. If every variable x_i has cardinality r, the number of required edge

feature evaluations is $2r - 1$. On the other hand, both inference and belief propagation need to evaluate f_{ij} for every possible assignment to x_i, x_j, in order to compute the normalization constant for the model. The resulting number of required feature evaluations is r^2. The resulting speedup of pseudolikelihood is approximately $\frac{1}{2}r$, which is very large for high-cardinality variables.

More generally, when every variable $x_i \in X$ has at most m neighbors, then for a log-linear model $P(X \mid E, w)$ from (3.9), pseudolikelihood requires $O(|X|rm)$ time to compute. Compared to the $O\left(r^{treewidth}\right)$ complexity of computing the log-likelihood, pseudolikelihood achieves an exponential speedup in terms of the dependence on the variables cardinality r. Together with the fact that pseudolikelihood of a log-linear model is concave in feature weights, the computational efficiency makes pseudolikelihood a very attractive optimization objective for learning discriminative log-linear models.

Therefore, *even for tree-structured models,* using pseudolikelihood instead of the exact conditional log-likelihood can result in significant speedups (proportional to the cardinality of variables $x_i \in X$) in both the structure learning phase (optimizing parameters u on line 5 of Alg. 3.3) and feature weights w learning phase (line 3 of Alg. 3.2). Observe that it is possible to use exactly the same gradient-based optimization approach, such as L-BFGS, for optimizing the pseudolikelihood as for optimizing the exact log-likelihood, because pseudolikelihood is essentially a sum of single-variable likelihoods. We will denote the ESS-CRF approach with pseudolikelihood optimization on line 3 of Alg. 3.2 and 5 of Alg. 3.3 as PseudoLLH ESS-CRF.

3.5.2 Max-margin feature weights learning

In a large number of applications, such as webpage classification (Taskar et al., 2002), other natural text processing problems (Lafferty et al., 2001), semantic labeling of image regions in computer vision (Ladicky et al., 2009) and in other domains, discriminative probabilistic models are used for the problem of *structured prediction,* where one aims to find the most probable assignment to the unknown variables:

$$\boldsymbol{X}^*_E = \arg\max_X P(X \mid \boldsymbol{E}). \tag{3.28}$$

The term "structured" refers to the structure of the output space that can be captured, for example, by the low-dimensional features of a log-linear model (3.9). In structured prediction problems, although probabilistic interpretation of a model is often a useful tool, motivating concrete approaches to optimizing model parameters, ultimately the quality of the model is measured by its prediction accuracy. In particular, relational datasets that we used for experiments in this chapter give rise to structured prediction problems. One can see that if the structured prediction (3.28) is the only goal of the particular application, then one is not actually interested in accurate approximation of the conditional distribution $P(X \mid \boldsymbol{E})$ for all values of X. The only important property of the conditional model is the accuracy of the mode. It has long been recognized, both for single-variable prediction problems (Vapnik, 1995; Crammer and Singer, 2002), and for structured prediction using graphical models (Taskar et al., 2003), that conditional log-likelihood (3.9) is often not the optimal objective for learning a model if prediction of the most likely assignment is the only purpose. An objective that is more directly related to the prediction problem is the *margin* between the score of the true answer $\boldsymbol{X}$ observed in the training data and the highest scoring incorrect answer:

$$\text{margin}(\boldsymbol{X}, \boldsymbol{E}, w) \equiv \frac{1}{\|w\|}\left(\sum_{f_\alpha \in \mathcal{F}} w_\alpha f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) - \max_{\boldsymbol{X}' \neq \boldsymbol{X}} \sum_{f_\alpha \in \mathcal{F}} w_\alpha f_\alpha(\mathbf{X}'_\alpha, \boldsymbol{E})\right), \tag{3.29}$$

where normalization by $\|w\|$ prevents changing the margin arbitrarily by simply scaling the feature weights w. Intuitively, maximizing the objective (3.29), proposed by Cortes and Vapnik (1995), results in a model that gives correct predictions in the most robust manner, measured in terms of the gap to the incorrect answers. One can see that maximizing the margin in the case of a structured model (3.29) is infeasible: because the number of possible assignments to X is exponential in $|X|$, the maximum in (3.29) is taken over exponentially many assignments, so even computing the objective is intractable. Fortunately, for a slightly different notion of the margin exact optimization is tractable for low-treewidth models. In (Taskar et al., 2003), a *structured margin* is introduced, which takes into account the magnitude of prediction error (in terms of Hamming distance in this example, but more general notions of error are possible):

$$\text{hamming-margin}(\boldsymbol{X}, \boldsymbol{E}, w) \equiv \frac{1}{\|w\|} \min_{\boldsymbol{X}' \neq \boldsymbol{X}} \frac{1}{\sum_{x_i \in X} \mathcal{I}(\mathbf{x_i} \neq \mathbf{x'_i})} \sum_{f_\alpha \in \mathcal{F}} w_\alpha \left(f_\alpha(\mathbf{X}_\alpha, \boldsymbol{E}) - f_\alpha(\mathbf{X}'_\alpha, \boldsymbol{E}) \right). \tag{3.30}$$

Intuitively, the structured margin (3.30) normalizes the robustness of a prediction (the margin (3.29)) by the scale of possible error in predicting $\boldsymbol{X}'$ instead if the true assignment $\boldsymbol{X}$ to make sure that the prediction is more robust against significant errors. Moreover, the structured margin (3.30) also has computational benefits. As Taskar et al. (2003) have shown, maximizing (3.30) can be formulated as a polynomial-sized quadratic program that yields the exact optimum $\mathbf{w}^*$ whenever the features $\mathcal{F}$ induce a low-treewidth model $\mathbb{T}(\mathcal{F})$, and approximate optimum when $\mathbb{T}(\mathcal{F})$ have high treewidth. Moreover, a message passing similar to belief propagation can be used to solve the QP efficiently. Taskar et al. (2003) have demonstrated that max-margin learning of log-linear models yields better accuracy than conditional random fields with the same features on a variety of problems.

As the approach of Taskar et al. (2003) is simply a different way to learn parameters of a log-linear model, it can be plugged in directly on line 3 of the general evidence-specific approach in Alg. 3.2, replacing the conditional log-likelihood optimization. Because the rest of Alg. 3.2 remains unchanged, the treewidth of the model induced by the features $\mathbb{T}(\mathcal{F}(\boldsymbol{E}, u))$ is guaranteed to be small, resulting in exact optimization of the feature weights w. We will call such an approach ESS-M3N, after the original M3Ns (max-margin Markov networks) of Taskar et al. (2003). The quadratic program required for optimizing the margin with respect to w is solved by the CPLEX QP solver (IBM, 2010) in our implementation.

3.6 Experiments

We have tested the ESS-CRF approach on both propositional and relational data. With the large number of parameters needed for the propositional case, namely $O(|X|^2)$, our approach is only practical for cases of abundant data. So our experiments with propositional data serve only to prove the concept, verifying that ESS-CRF can successfully learn a model better than a single tree baseline. In contrast to the propositional settings, in the relational cases the relatively low parameter space dimensionality, namely $O(|\mathbb{R}|^2)$, almost eliminates the overfitting problem. As a result, on relational datasets ESS-CRF is an attractive approach in practice. Our experiments show that ESS-CRFs match the accuracy of the state of the art high-treewidth discriminative models on several real-life relational datasets and at the same time ESS-CRFs are an order of magnitude more efficient during testing.

Dataset	Variables	Edges
WebKB Cornell	280	509
WebKB Texas	291	524
WebKB Washington	315	634
WebKB Wisconsin	454	1419
WebKB average	335	771
Segmentation (Gould et al., 2008), average per image	228	621
Segmentation (Mori et al., 2004), average per image	205	532

Table 3.1: Characteristics of the relational datasets.

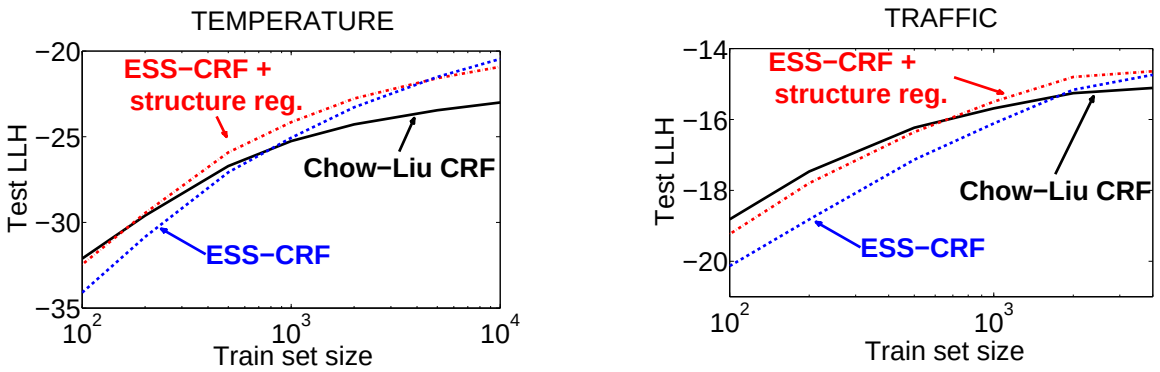


Figure 3.4: Test log-likelihood for TEMPERATURE dataset (left) and TRAFFIC dataset (right) depending on the amount of training data.

3.6.1 Propositional models

Here, we compare ESS-CRFs with fixed tree CRFs, where the tree structure learned by the Chow-Liu algorithm using $P(X)$. We used TEMPERATURE sensor network data (Deshpande et al., 2004) and San Francisco TRAFFIC data (Krause and Guestrin, 2005), discussed in more detail in section 2.4.2. For both datasets, 5 variables were used as evidence E and the rest as unknowns X. We have found it useful to regularize the conditional Chow-Liu (Alg. 3.6) by only choosing at test time from the edges that have been selected often enough during training. In Fig. 3.4, we compare conditional log-likelihood for ESS-CRFs and fixed tree CRF models with structures learned by the Chow-Liu algorithm. For the ESS-CRF approach, we plot results for both regularized ("ESS-CRF + structure reg.") and unregularized ("ESS-CRF") structures. One can see that in the limit of plentiful data ESS-CRF does indeed outperform the fixed tree baseline. However, because the space of available models is much larger for ESS-CRF, overfitting becomes an important issue and careful regularization is necessary.

3.6.2 Relational data: hypertext classification

Dataset description and experimental setting

This dataset, originally from Craven et al. (1998), contains the text and links structure of webpages from the computer science departments of four major universities. The goal is to classify every webpage into

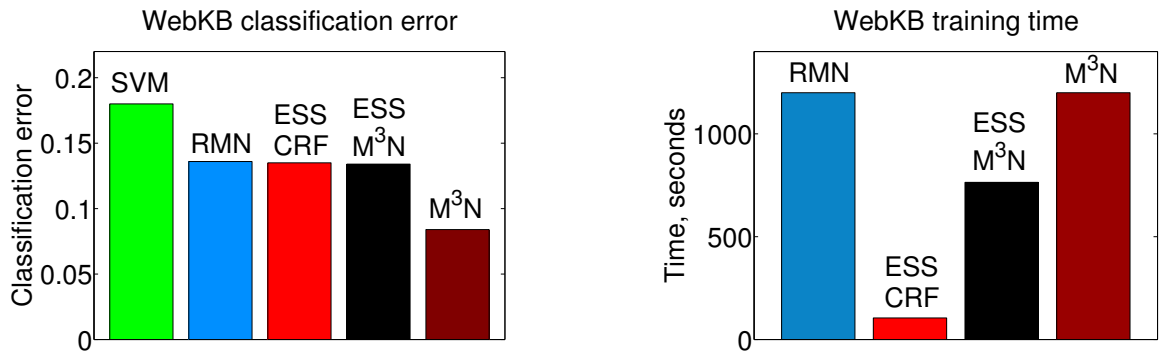


Figure 3.5: WebKB results. Classification errors (left) and training times (right).

one of `course`, `faculty`, `student`, `project`, or `other`. We have used the post-processed version of the data from Taskar et al. (2002). Here, every webpage is represented by a random variable x_i in the probabilistic graphical model. Webpage text is represented as a *bag of words*, that is, single-variable features are indicator functions:

$$f_i^{(j)}(x_i) = \mathcal{I}(\text{webpage } i \text{ contains word } j).$$

Link structure is represented with pairwise indicator features:

$$f_{ij}(x_i, x_j) = \mathcal{I}(\text{webpage } i \text{ links to webpage } j).$$

The number of variables and pairwise features for WebKB is shown in Table 3.1.

For every university, the links from its webpages outside the university website have been removed. As a result, every university is represented by a separate connected component in the model. In particular, such connectivity enables one to learn on the subset of the data corresponding to some universities and test on the remaining ones, without mixing the training and testing data. Following the previous work, we trained on 3 universities and tested on the remaining one, choosing the regularization parameters via cross-validation. All the reported results are averaged over the four possible choices of the test set.

We compare our ESS-CRF and ESS-M3N approaches to the single-node SVM classifier, discriminative relational Markov network (i.e., a CRF with loopy graph structure) that includes pairwise features for all of the hyperlinks (Taskar et al., 2002), and a max-margin Markov network (Taskar et al., 2003). Observe that all of the models except for the SVM use the same set of features, making the learning and inference algorithms the only source of the differences.

Results

WebKB results are shown in Fig. 3.5. From the comparison of the prediction accuracy (left plot), one can see that ESS-CRF provides the same accuracy as the loopy relational Markov network (RMN) with discriminatively learned parameters. In other words, if one optimizes the conditional log-likelihood of the model, then discarding some of the features to obtain a tree structure does not increase error for the WebKB dataset. On the other hand, comparing the max-margin versions of the two approaches (ESS-M3N versus the loopy M3N), one can see that ESS-M3N does not improve on ESS-CRF, unlike the loopy

M3N, which produces a significant improvement. It is possible that the lack of improvement is due to the fact that ESS-M3N is not a fully max-margin approach: while feature weights w are indeed learned to maximize the prediction margin, the structure selection parameters u in ESS-M3Ns are still learned to maximize conditional likelihood. Constructing a fully principled maximum margin counterpart for ESS-CRFs remains an open problem.

The main advantage of ESS-CRF approach over loopy models is in the efficiency. In the right plot of Fig. 3.5 we compare the parameter learning times of loopy models and their evidence-specific counterparts. The timings for RMNs and M3Ns here are from Getoor and Taskar (2007), who obtained their timings on a 800MHz Pentium III CPU. To make the timings directly comparable, we used a 800MHz Pentium M CPU, which has a similar architecture to Pentium III and the same frequency. One can see from Fig. 3.5 that ESS-CRFs are an order of magnitude faster to train than loopy models, which use discriminative parameter learning with approximate inference. At the same time, ESS-M3Ns take roughly the same time to learn as the loopy counterparts, which is likely due to the fact the generic QP solver from the CPLEX package is unable to recognize and exploit the tree structure of the constraints. At test time, ESS-CRFs also produce an order of magnitude speedup over loopy models: constructing an evidence-specific model and performing exact inference in it takes on average 0.17 seconds for both ESS-CRFs and ESS-M3Ns versus 7 seconds for loopy models.

To summarize, on WebKB dataset ESS-CRFs produce an order of magnitude speedup over high-treewidth discriminative models and yield the same prediction accuracy. On the other hand, the max-margin learning of weights w in the ESS-M3N approach does not produce an accuracy improvement, unlike the max-margin parameter learning in the high-treewidth case. Moreover, in order for the ESS-M3N approach to achieve the same speedup as ESS-CRF over the high-treewidth models one will need to apply a weight learning algorithm, such as sequential minimal optimization (Taskar et al., 2003) or subgradient descent (Ratliff et al., 2007), which is more specialized than the generic CPLEX QP solver and are able to exploit the tree structure of the model for efficient optimization.

3.6.3 Relational data: image segmentation

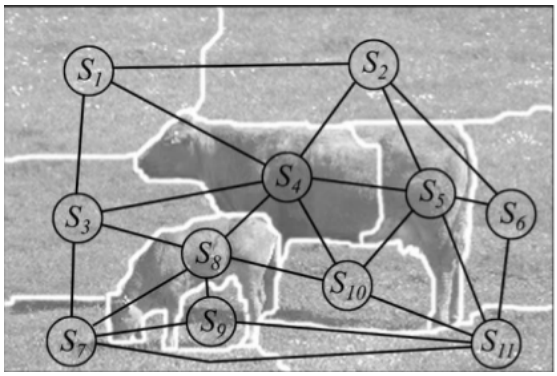
Dataset description and experimental setting

The problem of image segmentation, or, more precisely, of annotating every pixel of an image with the class of the object that the pixel corresponds to, is one of the fundamental problems of computer vision. One class of state of the art approaches to image segmentation treats the pixels or *superpixels* (Ren and Malik, 2003) of the image as variables in a graphical model, Pairwise or higher-order features are then used to encode the information about co-occurrence of different object classes, the preference for the label boundaries to occur along the lines of sharp contrast, etc. In addition to pairwise features, single-variable features are used to encode the local appearance information in the vicinity of the (super)pixel in question. The resulting models are conditional random fields, where query variables X denote the assignment of labels to pixels, and evidence E encodes the information about the image, such as brightness, color, response to Gabor filters, and so forth. CRF-based approaches achieve state of the art accuracy on standard benchmarks (Gould et al., 2008; Ladicky et al., 2009).

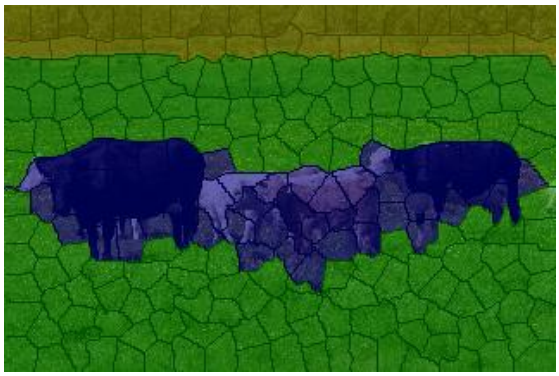
Here, we compare our ESS-CRF approach, and its PseudoLLH ESS-CRF modification, to a high-treewidth CRF model from Gould et al. (2008) and a logistic regression baseline that considers every superpixel in isolation. Because exact log-likelihood computation is infeasible for high-treewidth CRFs, following

(a) An example input image for the segmentation problem.

(b) An oversegmentation into superpixels.



(c) CRF connectivity defined by the superpixels, picture taken from Gould et al. (2008).



(d) The resulting segmentation obtained using inference on the CRF model. Blue color encodes class *cow*, green - *grass*, greenish yellow - *tree*.

Figure 3.6: An illustration of the workflow for image segmentation and labeling using a probabilistic graphical model.

Gould et al. (2008) we learned feature weights by optimizing pseudolikelihood. Both ESS-CRF and the high-treewidth model use the same features, described in detail in (Gould et al., 2008) and extracted using an implementation from the STAIR vision library (SVL, Gould et al., 2010). We notice that although other approaches, such as (Ladicky et al., 2009), have demonstrated better accuracy than Gould et al. (2008), much of that difference is due to improved design of the low-level features. Similarly to the WebKB experiments, the value of the comparison here is in eliminating all of the sources of performance differences except for the learning and inference algorithms.

The model construction, illustrated in Fig. 3.6, follows the approach of Gould et al. (2008). First, a source image (Fig. 3.6a) is over-segmented into superpixels (Fig. 3.6b) using an approach such as (Mori et al., 2004). Then, for every superpixel the values of raw *single-variable features* based on color and texture are extracted (Barnard et al., 2003). The raw single-variable features of the training images are then used as inputs for a set of one-against-all AdaBoost classifiers (Schapire and Singer, 1999). The outputs of those classifiers form the effective single-variable features used in the graphical models and in the baseline single-variable logistic regression classifier. For every pair of superpixels that share a boundary, *pairwise*

features are introduced (Fig. 3.6c). In (Gould et al., 2008), a constant feature and the coordinates of superpixels centroids were used as the pairwise features. We have also experimenting with the extending single-variable features to the pairwise case: for every feature $f_i^{(k)}(x_i)$, a set of corresponding pairwise features is introduced:

$$f_{ij}^{(km)}(x_i, x_j) = \mathcal{I}(x_j = m) \cdot f_i^{(k)}(x_i).$$

The resulting PGM is relational: all the edges in the model, and across the models corresponding to different images, share the feature weights, as do all variables for single-variable feature weights. A set of densely labeled images is used for learning the weights. During training, every superpixel is assigned a label corresponding to the majority vote of it member pixels. During testing, an unknown images is segmented into superpixels, the features are extracted, the relational model is grounded on those superpixels, and probabilistic inference is used to compute the most probable labels assignment for the superpixels. Every pixel then gets a label of its container superpixels (Fig. 3.6d). The labeling accuracy is computed on the level of individual pixels.

For the experiments, we use the MSRC dataset (Criminisi, 2004) with 591 images and 21 different object classes. We used the train/test split of Shotton et al. (2006), training on the 276 training images and testing on the remaining 315 (test and validation sets in Shotton et al., 2006). We used two alternative segmentations of images into superpixels: the one provided by Gould et al. (2008) (segmentation A), and the one resulting from running the implementation of Mori et al. (2004) on the source images (segmentation B). Both for ESS-CRF approach and for high-treewidth models, regularization was chosen using 5-fold cross-validation. Following the recommendations in (Gould et al., 2010), for pseudolikelihood training of the high-treewidth models, we only regularized the edge parameters, but not single-variable parameters. We have found that regularizing single-variable parameters decreases the quality of the pseudolikelihood results for this dataset. For ESS-CRFs, the same global regularization was used for both edge and single-variable parameters. All the experiments were run on a 2.7GHz Intel Core 2 Duo CPU.

Results

The results for the image segmentation problem are shown in Table 3.2 (accuracy information) and Table 3.3 (timing information) and summarized in the plots of Fig. 3.7 (accuracy) and Fig. 3.8 (timings). One can see that

1. ESS-CRFs provide essentially the same accuracy as the high-treewidth models.
2. ESS-CRFs and their pseudolikelihood modification are an order of magnitude faster than high-treewidth models during testing.
3. ESS-CRFs are one to two orders of magnitude slower than pseudolikelihood-based high-treewidth models during training.
4. PseudoLLH ESS-CRFs are more than an order of magnitude faster during training than the baseline ESS-CRFs, while capturing 85% of the accuracy gain of ESS-CRFs relative to single-variable models.

One can see that the performance and efficiency advantage of ESS-CRFs over high-treewidth models at test time is the same here as for the hypertext classification problem. However, unlike the WebKB dataset, the training time of ESS-CRFs here is much longer than of high-treewidth models. This difference has two interrelated causes: larger variable cardinality in the image segmentation setting (21 versus 5 for WebKB)

Pixelwise classification accuracy				
	Centroid edge features		Centroids + single-var. features on edges	
	SUM-PROD	MAX-PROD	SUM-PROD	MAX-PROD
Logistic regression segm. A	0.655			
Loopy CRF segm. A	0.760	0.761	0.759	0.761
ESS-CRF segm. A	0.750	0.753	0.759	0.751
PseudoLLH ESS-CRF seg. A	0.743	0.742	0.734	0.736
Logistic regression segm. B	0.658			
Loopy CRF segm. B	0.752	0.747	0.751	0.749
ESS-CRF segm. B	0.750	0.749	0.761	0.752
PseudoLLH ESS-CRF seg. B	0.739	0.739	0.737	0.736
Logistic regression average	0.657			
Loopy CRF average	0.756	0.754	0.755	0.755
ESS-CRF average	0.750	0.751	0.760	0.752
PseudoLLH ESS-CRF avg.	0.741	0.741	0.736	0.746

Table 3.2: Pixelwise classification accuracy results, on the image segmentation problem, for the ESS-CRF approach with parameters learned by optimizing conditional log-likelihood (ESS-CRF) and pseudolikelihood (PseudoLLH ESS-CRF), a high-treewidth CRF with parameters learned using pseudolikelihood, and a logistic regression model that only uses the single-variable features.

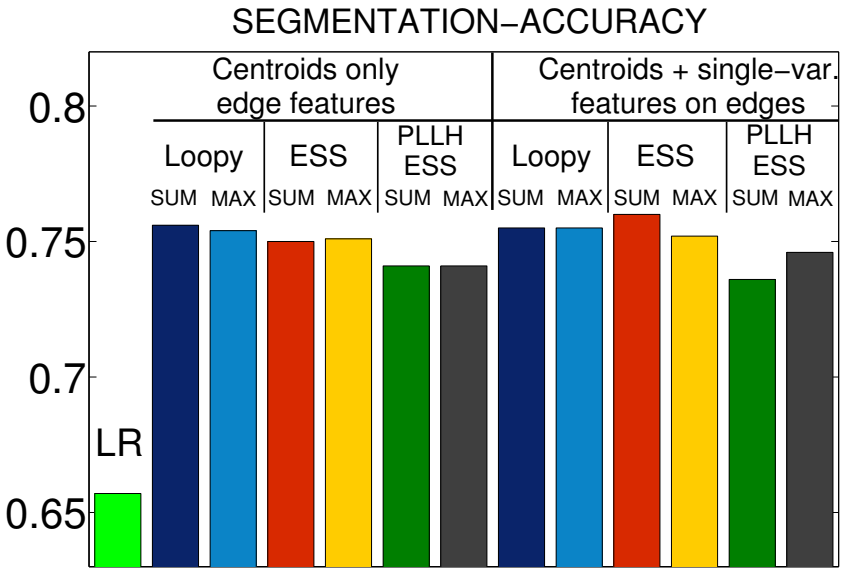


Figure 3.7: Image segmentation results: pixelwise classification accuracy for a high-treewidth CRF with parameters learned using pseudolikelihood (Loopy), ESS-CRF approach with parameters learned by optimizing conditional log-likelihood (ESS) and ESS-CRF approach with parameters learned using pseudolikelihood (PLLH ESS). SUM and MAX denote sum-product and max-product inference correspondingly. LR is logistic regression on single-node features.
The plots correspond to the numbers in Table. 3.2.

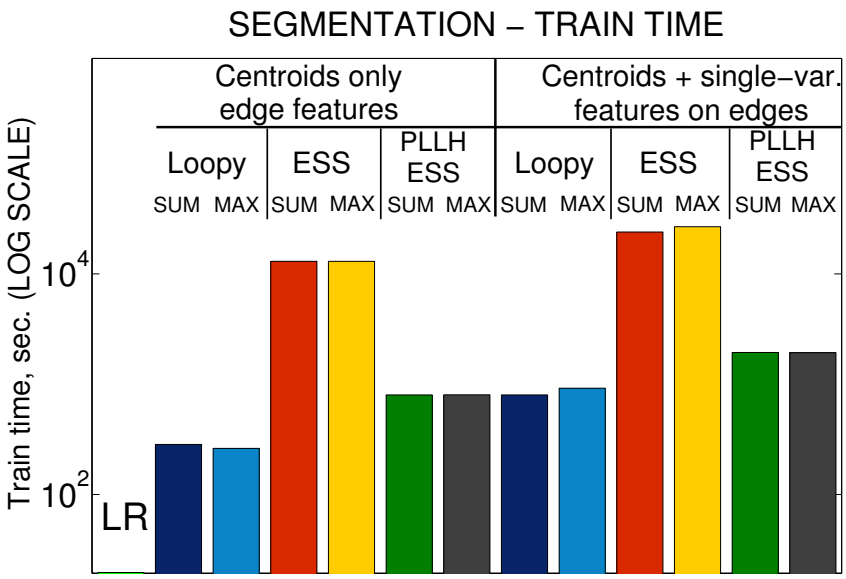
Training time for 276 images, seconds				
	Centroid edge features		Centroids + single-var. features on edges	
	SUM-PROD	MAX-PROD	SUM-PROD	MAX-PROD
Logistic regression segm. A	21.3			
Loopy CRF segm. A	249	256	801	800
ESS-CRF segm. A	$1.36 \cdot 10^4$	$1.36 \cdot 10^4$	$2.28 \cdot 10^4$	$2.84 \cdot 10^4$
PseudoLLH ESS-CRF seg. A	903	907	$2.23 \cdot 10^3$	$2.20 \cdot 10^3$
Logistic regression segm. B	17.9			
Loopy CRF segm. B	318	267	799	$1.04 \cdot 10^3$
ESS-CRF segm. B	$1.24 \cdot 10^4$	$1.24 \cdot 10^4$	$2.51 \cdot 10^4$	$2.51 \cdot 10^4$
PseudoLLH ESS-CRF seg. B	694	797	$1.65 \cdot 10^3$	$1.66 \cdot 10^3$
Logistic regression average	19.6			
Loopy CRF average	284	262	800	920
ESS-CRF average	$1.30 \cdot 10^4$	$1.30 \cdot 10^4$	$2.40 \cdot 10^4$	$2.68 \cdot 10^4$
PseudoLLH ESS-CRF avg.	799	802	$1.94 \cdot 10^3$	$1.93 \cdot 10^3$

Total test inference time for 315 images, seconds				
	Centroid edge features		Centroids + single-var. features on edges	
	SUM-PROD	MAX-PROD	SUM-PROD	MAX-PROD
Logistic regression segm. A	0.15			
Loopy CRF segm. A	397	126	390	132
ESS-CRF segm. A	6.1	6.1	12.7	12.7
PseudoLLH ESS-CRF seg. A	6.1	6.1	12.7	12.7
Logistic regression segm. B	0.11			
Loopy CRF segm. B	243	107	276	95
ESS-CRF segm. B	5.5	5.5	10.8	10.8
PseudoLLH ESS-CRF seg. B	5.5	5.5	10.8	10.8
Logistic regression average	0.13			
Loopy CRF average	320	116	333	113
ESS-CRF average	5.8	5.8	11.8	11.8
PseudoLLH ESS-CRF avg.	5.8	5.8	11.8	11.8

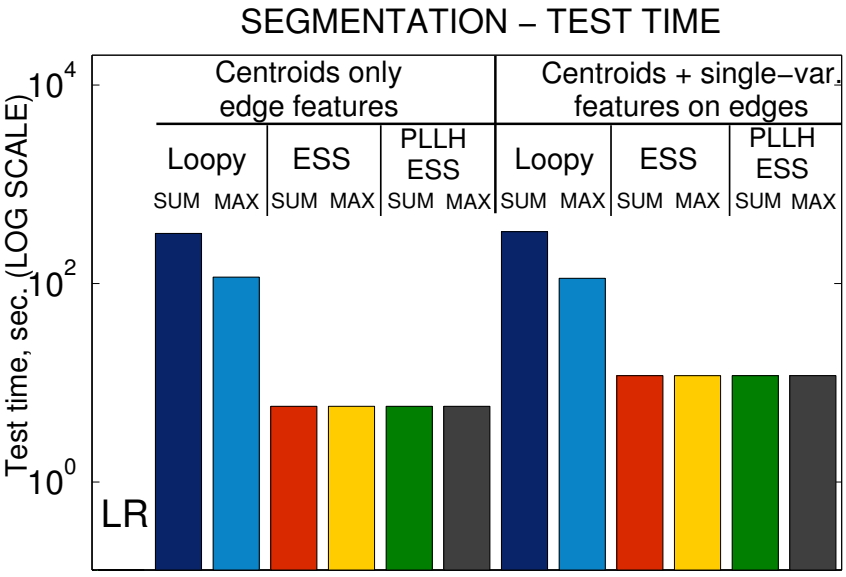
Table 3.3: Timing results, on the image segmentation problem, for the ESS-CRF approach, PseudoLLH ESS-CRFs, a high-treewidth CRF with parameters learned using pseudolikelihood, and a logistic regression model that only uses the single-variable features.

and the choice of objective for learning the parameters (pseudolikelihood for image segmentation versus conditional log-likelihood for WebKB).

As we discussed in Section 3.5.1, the advantage of pseudolikelihood in computational efficiency grows with the cardinality of the query variables. Even for tree-structured models, pseudolikelihood has an advantage of $O(r)$ times over the exact conditional likelihood. This $O(20)$ performance advantage (for the variable cardinality of 21 in the dataset) is consistent with the training time difference between ESS-CRFs and high-treewidth pseudolikelihood-based models in the image segmentation setting.



(a) Training time.



(b) Inference time during testing.

Figure 3.8: Train and test times for the image segmentation problem for the same approaches as in Figure 3.7. The plots correspond to the numbers in Table. 3.3. Note that both plots in this figure use a logarithmic scale.

The inference speedups at test time further confirm that, similar to the WebKB setting, learning the parameters for the image segmentation CRFs using belief propagation or other approximate inference techniques would be an order of magnitude slower than ESS-CRFs training. In other words, ESS-CRFs are in the middle of the training complexity spectrum between pseudolikelihood and approximate inference-based techniques, with PseudoLLH ESS-CRFs being closer to standard pseudolikelihood-based models. The fact that PseudoLLH ESS-CRFs take longer to train that pseudolikelihood-based models with fixed struc-

ture can be explained by two factors. First, while fixed structure models only have to learn one set of feature weights w, PseudoLLH ESS-CRFs need to learn both weights u that define the structure of the models, and the standard feature weights w. In other words, one can say that PseudoLLH ESS-CRFs has to learn twice. Second, when a relational dataset is converted to a set of propositional pairwise conditional density estimation problems (c.f. Alg. 3.7, lines 6-9), there is duplication of single-variable features for every relation instance that a variable participates in. Processing this duplicated data during learning also requires extra computation and increases learning complexity.

Finally, we observe that, because computation of conditional log-likelihood and gradient for every datapoint (image) is independent of the other datapoints, parameter learning is an *embarrassingly parallel* problem (Foster, 1995) and therefore is trivial to speed up both using multicore CPUs or multiple machines. On the other hand, although parallelizing inference based on message passing has also been successfully done (Gonzalez et al., 2009), the issues of cache consistency, locking and communications between the multiple machines in a cluster settings are more severe for inference. It follows that ESS-CRFs are an especially attractive approach in interactive and real-time systems, where test time latency is important.

3.7 Related work

The two general cornerstones of our ESS-CRF approach, namely using models that become more sparse when evidence is instantiated, and using *multiple* tractable models to avoid restrictions on the expressive power inherent to low-treewidth models, have been discussed separately in the existing literature, albeit in a significantly different concrete form.

Evidence-specific model simplification. Parametrizing a densely connected model in such a way that certain evidence assignments "disable" some of the dependencies that hold for the model in general is the key idea of context-specific independence (CSI, Boutilier et al., 1996). There, the equality constraints on the values of the factors can be exploited to obtain significant inference speedups. Moreover, using factors that possess context-specific independence properties can also be used as a means of regularization (desJardins et al., 2005). However, as we discussed in section 2.5, until recently, context-specific has been treated as a *local* property of the potentials. As a result, models exploiting CSI admit significant inference speedups for some evidence assignments, but do not guarantee tractable exact inference for all evidence assignments. More recently, the work of Lowd and Domingos (2008) and Gogate et al. (2010) provided a way to exploit CSI to learn tractable high-treewidth models. In both (Lowd and Domingos, 2008) and (Gogate et al., 2010), the tractability is achieved by almost completely discarding the traditional PGM graph as an instrument of analyzing complexity and focusing on the much finer structure of inference as an arithmetic expression. However, extending these approaches for tractable high-treewidth model learning to relational settings remains an interesting challenge.

Multiple structures to improve expressive power. The standard way to combine multiple simple models to represent a complex distribution is to use a *mixture model* (c.f., for example, McLachlan and Peel, 2000), where the resulting distribution is a convex combination of the elementary distributions corresponding to individual simple models:

$$P(X) \equiv \sum_{\text{model} \in \mathcal{M}} P(X \mid \text{model}) \cdot P(\text{model}).$$

Mixture models are a popular approach in machine learning and statistics in general. In the context of probabilistic graphical models, probabilistic mixtures have been used both on the "coarse" level of combining multiple simple model structures (Thiesson et al., 1998), and on the low level of representing low-dimensional marginal distributions during inference (Sudderth et al., 2003).

A very close in spirit to the approach of this chapter is the work of Meilă and Jordan (2001), who have shown that in a generative setting one can efficiently compute marginals for a mixture of all possible trees over the variables X, as long as (a) all the trees share the parameters of their respective common edges and (b) the probability of any given tree in the model factorizes into a product of edge weights, which are also shared across all trees. The sharing of edge parameters enabled Meilă and Jordan (2001) to avoid explicitly enumerating and marginalizing over the superexponential in $|X|$ number of all possible trees. Instead, marginalization operations are reduced to computing determinants of $|X| \times |X|$ matrices, which can be done in $O(|X|^3)$ time.

Our evidence-specific CRF approach and the work of Meilă and Jordan (2001) rely on the same key idea of different structures that share edge parameters for their common elements. At the same time, there are also important differences. Unlike our approach, which selects a single structure for a concrete evidence assignment, the approach of Meilă and Jordan (2001) always maintains a mixture of all possible trees. On the one hand, retaining all the possible structures makes the model more robust. On the other hand, our approach of choosing a single tractable structure at test time makes it possible to perform not only exact marginalization, but also exact MAP inference efficiently, while mixture models in general, including those of Meilă and Jordan (2001), only support marginalization.

Moreover, the approach of Meilă and Jordan (2001) is restricted to the generative setting, because it relies in an essential way on the parametrization of tree-structured distributions with single-variable and edge marginals (c.f. the discussion in section 2.1 of this thesis and in particular Lemma 7 for the more general case of junction trees) and the fact that all edge marginals agree on their respective single-variable marginals. In a discriminative setting, such a parametrization is typically not available, and one has to run sum-product algorithm on individual structures to compute marginals, rendering the approach of Meilă and Jordan (2001) inapplicable. It remains an open problem whether the robust mixtures over all possible trees of Meilă and Jordan (2001) can be adapted to the discriminative setting.

Importantly, the agreement constraints on edge potentials with respect to the single-variable marginals are *not* necessary for traditional mixture models, where it is feasible to process every individual mixture component separately. Whenever the number of mixture components is moderate, parameters of a mixture of log-linear models (3.6) can be learned in a discriminative way either using a generalized expectation maximization (EM) framework (Salojärvi et al., 2005) or by directly optimizing the conditional likelihood (Gunawardana et al., 2005). The extra parametrization consistency is only required by the approach of Meilă and Jordan (2001) in order to be able to process every edge just once during inference, instead of processing that edge separately for every possible tree containing that edge.

Learning fixed structures in a discriminative setting. Because the discriminative setting, where one is interested in the conditional distribution $P(X \mid E)$, is a generalization of the generative setting, where one is interested in $P(X)$, the hardness results for the problem of structure learning discussed in section 2.5 carry over from the generative to the discriminative setting. It follows that learning an optimal structure in the discriminative setting is intractable for most formulations. Moreover, often the features of the model are designed using specific domain knowledge, and the model structure is not learned, but simply inferred from the scopes of the features. Among the hand-designed structures, especially popular are chains and skip-chains (e.g., Sutton and McCallum, 2004), as well as grid-like structures, because they both encode

intuitive information on local temporal or spacial interactions and result in compact or even tractable, in the case of chains, models.

As it is a more general problem, structure learning in the discriminative setting poses additional difficulties compared to the generative setting. The main source of the extra difficulty is the fact that the low-dimensional conditional distributions $P(X_\alpha \mid E)$ typically cannot be accurately approximated. In the generative case, results such as Hoeffding's inequality (Hoeffding, 1963) guarantee that the low-dimensional marginal distributions $P(X_\alpha)$ can be estimated arbitrarily accurately using only a polynomial in the inverse accuracy $\frac{1}{\Delta}$ number of samples. As a consequence, low-dimensional sufficient statistics of the *true* underlying distribution, such as entropy, can also be estimated with arbitrary accuracy using only a moderate amount of data (c.f. Theorem 30 originally from Höffgen, 1993). It follows that in the generative case, structure learning algorithms are able to use low-dimensional empirical marginals $\overline{P}(X_\alpha)$ as substitutes for the true low-dimensional marginals $P(X_\alpha)$ and still retain the accuracy guarantees on the likelihood of candidate models (for score-based approaches) and consistency of independence testing (for constraint-based approaches). In the discriminative setting, however, to get equivalent quality guarantees, one would needs to accurately approximate the low-dimensional conditionals $P(X_\alpha \mid E)$ *for every possible value of* E, which is a much harder problem than approximating $P(X_\alpha)$, because the evidence is high-dimensional. As a result, in addition to the problem of selecting a high-scoring model out of the very rich space of possible structures, in the discriminative case there is also a problem of scoring the structures accurately. As Bradley and Guestrin (2010) have shown, even in the infinite samples limit and for the class of distributions $P(X \mid E)$ corresponding to compact log-linear conditional models (3.9), no linear combination of low-dimensional conditional entropies yields a model score that is guaranteed have the structure of the true PGM that produced the distribution as the global maximum. As an immediate corollary, Bradley and Guestrin (2010) have shown that even with the ability to find the globally optimal structure given the clique scores, optimal CRF structure learning using only local clique scores based on conditional entropies is impossible.

The impossibility result of Bradley and Guestrin (2010) means, in particular, that the approach of this chapter of replacing the approximate low-dimensional marginals $P(X_\alpha)$ with low-dimensional conditionals $P(X_\alpha \mid E = \boldsymbol{E})$ in a score-based structure learning algorithm is only a heuristic. However, both the empirical of this chapter, and for a related local score-based approach (Shahaf et al., 2009) indicate that such a heuristic often works well in practice.

As a result of the accuracy breakdowns in scoring the cliques locally, more popular in the discriminative setting are structure learning approaches based on L_1 regularization (e.g., Schmidt et al., 2008), because that do not rely on the decomposable scores in the first place. Another approach is to iteratively add features to the model, using the optimal parameters of the current structure to approximately evaluate feature quality. After a feature has been added, the parameters can be re-learned to both calculate the feature quality exactly and obtain a better basis for candidate feature evaluation for the next iteration (Torralba et al., 2004). More generally, any structure learning approach that does use decomposable model scores or local independence tests can be adapted to the discriminative setting in a straightforward way, but at a price of increasing the structure scoring complexity by a factor of $|\mathcal{D}|$, because inference has to be performed separately for every datapoint to score a structure. In particular, local search in the space of tractable structures with exact conditional likelihood scoring is possible, but evaluating candidate moves is quite expensive, especially with a large number of datapoints.

Compatibility with other approaches for learning discriminative graphical models.

Here, we briefly review some of the existing approaches related to learning discriminative PGMs that can be used to complement our framework of learning models with evidence-specific structure. Unlike the literature discussed above in this section, the following research results do not present an alternative to our approach. Rather, those existing results address the steps in the general graphical models workflow of Fig. 1.3 that our approach does not affect, such as parameters learning and feature selection. As we will show, these complementary approaches can be readily combined with the ESS-CRF framework of this chapter.

Learning discriminative parameters: conditional likelihood and maximum margin. The primary focus of this chapter was on learning discriminative log-linear models with tractable evidence-specific structure that maximize the conditional log-likelihood of the data (3.10). However, conditional log-likelihood is by no means the only possible optimization objective for our ESS-CRF framework. Depending on the end goal of applying the model and computational efficiency requirements, other objectives can be more suitable. For example, in section 3.5 we have shown that both pseudolikelihood (for improved efficiency) and classification margin (for structured prediction) can be adapted to our framework. In section 3.6.3, we have empirically demonstrated the large computational benefits and relatively small accuracy penalty of pseudolikelihood learning for evidence-specific models. Here, we review the related work on parameter learning more broadly and discuss how it can be integrated in our approach.

Although optimizing conditional log-likelihood (c.f. section 3.1) and max-margin parameter learning (c.f. section 3.5.2) are distinct objectives with different motivations, Zhu and Xing (2009) have shown that both objectives are special cases of a general unified framework. Moreover, tractable PGM structure yields similar benefits for learning with either objective. For conditional log-likelihood, exact inference enables the exact computation of the convex objective and its gradient, leading to efficient application of convex optimization techniques. For max-margin objective, as Taskar et al. (2003) have shown, a polynomial-sized QP can be formulated for learning the feature weights of a tractable log-linear model. Also, exact max-product algorithm can be used as an oracle of the most violated constraint for learning the max-margin weights via constraint generation (Altun et al., 2003). Finally, for an unconstrained max-margin optimization objective and the corresponding subgradient learning approach derived by Ratliff et al. (2007), inferring exact MAP assignment for the current parameters is needed in the inner loop of subgradient computation. It follows that learning optimal feature weights for tractable models is quite straightforward. Also, for subgradient-based maximum margin parameters learning (Ratliff et al., 2007), and for constraint generation max-margin approach (Altun et al., 2003), which only require inference to find only the most probable assignment, but not expected feature values, exact MAP inference via graph cuts can also be used for certain types of potentials (Kolmogorov and Zabih, 2004).

For high-treewidth models, where exact inference is infeasible, one can either use approximate inference or modify the objective so as to make computation tractable. Using approximate inference techniques, such as belief propagation (Pearl, 1988; Yedidia et al., 2000) or Gibbs sampling (Geman and Geman, 1984) often works well in practice, but leads to a loss of convergence guarantees, either because inference inaccuracies directly affect the (sub)gradient computation, because max-product belief propagation is not guaranteed to converge and thus detect a violated constraint even if there is one for the constraint generation approach (Altun et al., 2003), or because the constraints in the quadratic program designed to describe the marginal polytope only describe an outer polytope in the high-treewidth case (Taskar et al., 2003).

Feature selection. Although the ability to use correlated features without explicitly modeling their respective correlations is an important advantage of discriminative models (see the discussion in Section 3.1), in the small-sample regime overfitting remains an issue. Especially problematic are cases where it is not known in advance which features are actually informative for describing the dependence between the evidence E and the query variables X. In such cases, practitioners often include all the features that may possibly be useful into the model and rely on the discriminative learning framework to automatically determine which features are relevant by assigning them large weights.

To reduce overfitting, one typically maximizes a *regularized* conditional likelihood:

$$w^* = \arg\max CLLH(\mathcal{D} \mid w) - h(w),$$

where $CLLH(\mathcal{D} \mid w)$ is the conditional log-likelihood (3.10) and $h(w)$ is the *penalty term* that biases the weights towards a uniform distribution. A particularly popular penalty term is L_2 penalty: $h_2(w) = \lambda\|w\|_2^2$, where $\lambda > 0$ is the regularization constant. The main reason for the popularity of L_2 regularization is the fact that it keeps the overall optimization objective (3.7) not only concave, but also continuously differentiable. As a result, it is straightforward to optimize the regularized log-likelihood using one of the many existing gradient-based approaches. The drawback of L_2 regularization, however, is that it *does not induce sparsity:* every feature f_α typically gets a (small) nonzero weight $\boldsymbol{w}_\alpha \neq 0$. Therefore, when a lot of irrelevant features are included in the model, optimizing an L_2-regularized conditional likelihood will assign every irrelevant feature a nonzero weight, resulting in additional errors in estimating the conditional distribution $P(X \mid E, w)$ using the model (3.9).

One can see that even though in the limit of plentiful data discriminative learning with an L_2 will reduce the weights of irrelevant features to zero, in the finite sample case it is desirable to perform an extra step of *feature selection* (Guyon and Elisseeff, 2003) to try and filter out the irrelevant features. The problem of feature selection has received a significant amount of attention in the literature, and highly effective approaches have been developed (Schmidt et al., 2007; Zhu et al., 2010; Liu and Zhang, 2009).

The primary basis of most of the existing feature selection methods in log-linear models is simply to adopt L_1 regularization instead of L_2. Because the L_1 norm has a singularity at 0, it induces sparse model weights at optimum. Unfortunately, the very same property of the L_1 norm, namely the singularity at origin, also makes many of the standard continuous optimization techniques, such as L-BFGS (Liu and Nocedal, 1989) or conjugate gradient (Fletcher and Reeves, 1964), inapplicable, because with L_1 regularization the overall objective ceases to be continuously differentiable. As a result, alternative optimization approaches have been developed. Two main directions are (a) interleaving the search in feature space with optimizing the parameters for only the subset of the currently selected features (Zhu et al., 2010; Perkins et al., 2003) and (b) gradient-based methods that are robust to the discontinuities in the derivative of the objective (Schmidt et al., 2007; Andrew and Gao, 2007). Consistency results in terms of selecting the right set of features have also been obtained (Liu and Zhang, 2009).

Both grafting-style and direct gradient-based approaches for feature selection via optimization of an L_1-regularized log-likelihood can be directly combined with our framework of evidence-specific structures. Consider the two instances where feature selection is needed in more detail:

- The process of **learning** evidence-specific **structures** (Alg. 3.3 and 3.4). One can see that, given the low-treewidth propositional structure learning algorithm $\mathcal{A}_S$, learning evidence-specific structures using Alg. 3.3 and 3.4 is reduced to the problem of low-dimensional density estimation (c.f. line 3 of Alg. 3.4). Moreover, all the sparse feature selection approaches discussed in this section are conditional density estimation algorithms operating with log-linear models. As a result, any of the

L_1-regularized feature selection procedures (Zhu et al., 2010; Schmidt et al., 2007; Andrew and Gao, 2007) can be used directly as algorithm $\mathcal{A}_D$ in Alg 3.3 and 3.4.

Importantly, during evidence-specific structure learning (i.e., optimizing the parameters u), feature selection is only done for the purposes of low-dimensional density estimation. At the time of constructing the actual structures by Alg. 3.4, after the propositional structure learning algorithm $\mathcal{A}_S$ finds a low-treewidth structure $\mathbb{T}$ over the query variables on line 3 of Alg. 3.4, the set of features $\mathcal{F}(\mathbb{T})$ returned on line 4 of Alg. 3.4 includes *all of the original features $f_\alpha \in \mathcal{F}$ that match the structure $\mathbb{T}$,* including features that were assigned zero weight, $u_\alpha = 0$, by Alg. $\mathcal{A}_D$.

For example, suppose that for an edge $x_i - x_j$ there are two candidate features, namely $f_{ij}^{(1)}(x_i, x_j, E)$ and $f_{ij}^{(2)}(x_i, x_j, E)$, and L_1-regularized density estimation of the conditional distribution $\mathbf{P}\left(x_i x_j \mid \boldsymbol{E}\right)$ resulted in weights $u_{ij}^{(1)} = 0.7$ and $u_{ij}^{(1)} = 0$. Suppose further that for the evidence value $E = \boldsymbol{E}$, that edge $x_i - x_j$ belongs to the evidence-specific structure: $(i-j) \in \mathbb{T}$, where $\mathbb{T} \equiv \mathcal{A}_S(\widehat{P}(\cdot \mid \boldsymbol{E}))$. Then Alg. 3.4 would return *both* $f_{ij}^{(1)}$ and $f_{ij}^{(2)}$ as a part of the low-dimensional set of candidate features $\mathcal{F}(\mathbb{T})$. Such a behavior is due to the fact that Alg. 3.4 by construction is not aware of the nature of the conditional density estimator $\mathcal{A}_D$, and in particular the sparsity of the feature weights u induced by $\mathcal{A}_D$,. The treatment of the conditional density estimator $\mathcal{A}_D$, as a black box has two advantages. First, estimators of other forms than log-linear models can be plugged in without changing the rest of the approach. Second, even for sparse log-linear models, the optimal subsets of features $\mathcal{F}$ for estimating the low-dimensional conditionals $\mathbf{P}\left(X_\alpha \mid \boldsymbol{E}\right)$ and the structured (according to a low-treewidth $\mathbb{T}(E)$) high-dimensional conditionals $\mathbf{P}\left(X \mid \boldsymbol{E}\right)$ can be different. Committing to only the features selected for estimating the low-dimensional conditionals would yield suboptimal accuracy when estimating the high-dimensional conditional $\mathbf{P}\left(X \mid \boldsymbol{E}\right)$.

- The process of **learning the feature weights** w (line 3 of Alg. 3.7). Here, we can again plug any of the existing feature selection procedures (Zhu et al., 2010; Schmidt et al., 2007; Andrew and Gao, 2007) by exploiting two observations. First, existing approaches for optimizing the L_1-regularized log-likelihood also work in a high-dimensional setting, as long as the log-likelihood itself can be computed exactly. Second, the candidate features $\mathcal{F}(\mathbb{T})$ induce a low-treewidth structure over the variables X. Third, graph treewidth is monotonic in the set of edges, so dropping some of the features from $\mathcal{F}(\mathbb{T})$ can only decrease the induced treewidth over X.. Finally, for any candidate feature set $\mathcal{F}' \subseteq \mathcal{F}$ that induces a low-treewidth graph over X the log-likelihood for the high-dimensional conditional distribution (3.15) can be computed exactly.

One can see from the above examples that the modular structure of our approach (Alg. 3.3 and 3.4) enables one to directly plug in many of the existing and future approaches for specific subproblems in the discriminative graphical models workflow. This flexibility can be used to both construct novel approaches for learning evidence-specific models and to convert existing application-specific instances of discriminative PGM pipelines to the evidence-specific framework.

3.8 Discussion and future work

In this section, we have described the following main contributions:

1. A novel way to exploit information about the *values* of variables which are observed at test time to select the structure of discriminative probabilistic graphical models that is specific to the evidence

values at hand. The extra flexibility in the structure choice allows one to keep the benefits of efficient exact inference and exact parameters learning, without suffering a penalty on the expressive power arising from committing to a *single* tractable model. The key advantage of our approach over existing work on context-specific independence is the ability to guarantee low treewidth of the resulting models, and thus tractability of exact inference, not only in a propositional, but also in a relational setting.

Moreover, our approach (a) relies on *exactly the same input data as traditional graphical models* at both training and testing time, and (b) only affects the structure selection step of the general graphical models workflow of Fig. 1.3. As a result, practitioners not only can adapt ESS-CRFs for *any* setting where a standard CRF is currently used (i.e., there is *no loss of generality* with respect to traditional PGMs), but also such an adaptation requires very little effort: one only needs to replace the structure selection step of the full pipeline with our approach and retain all the other steps without any changes. In particular, any feature selection and parameter learning approaches can be used with our ESS models in a transparent manner.

2. A general framework that allows one to leverage the existing work on learning the structure of propositional tractable models and low-dimensional conditional density estimation to construct algorithms for learning discriminative models with evidence-specific structure.

3. An extension of the general framework for learning discriminative models with evidence-specific structure to the relational setting. Importantly, with our generalization, one can still use *propositional* algorithms for structure learning and low-dimensional conditional density estimation as building blocks. Structure learning in the propositional setting is much more extensively studied compared to the relational setting. Therefore, keeping the ability to rely on propositional algorithms significantly expands the range of existing approaches one can build upon.

4. An empirical evaluation demonstrating that in the relational setting our approach has equal or better accuracy than the state of the art algorithms for densely connected models, and at the same time is much more efficient computationally.

Here, we outline the possible directions for future work.

Experiments with more general thin junction trees. Our framework of Alg. 3.7 and 3.8 (for the relational setting) allows, in principle, a wide range of existing techniques for learning thin junction trees to be used for evidence-specific structure selection. In particular, the approach of chapter 2 of this thesis, and algorithms of Shahaf et al. (2009), Karger and Srebro (2001), Elidan and Gould (2008), and methods based on local search in the space of thin junction trees can all be plugged into Alg. 3.8 to yield a concrete evidence-specific structure learning approach. In practice, however, one will have to make a choice carefully, to achieve an optimal tradeoff of computational efficiency, expressive power, and sample complexity.

In our experiments, we have used the Chow-Liu algorithm, which is one of the most efficient and robust approaches, but only works with a class of models with rather limited expressive power. Potentially, using thin junction trees with higher treewidth may bring better approximation accuracy because of their ability to capture a richer set of dependencies. However, because the dimensionality of the parameters and the space of possible structures grow quickly with the treewidth, careful regularization may be needed with higher treewidth junction trees. The experiments of section 3.6.1 with the propositional data, where the absence of parameter sharing between different edges leads to high dimensionality of parameters, illustrate that sample complexity of ESS-CRFs can be quite high in the absence of effective regularization. One

possible regularization technique is to restrict the set of structure elements (such as junction tree cliques) to those that were selected frequently enough during training - the approach we took in the experiments of section 3.6.1. Another alternative is to bias towards the prior distribution $\widehat{P}(X_\gamma)$ the conditional density estimates $\widehat{P}(X_\gamma \mid E, u)$ that are used to score cliques X_γ by the structure learning approach. Such a regularization of the conditional density estimates will implicitly regularize the evidence specific structures by only selecting cliques X_γ absent from the generative model structure if their conditional distribution given $\boldsymbol{E}$ is sufficiently different from the prior.

A theoretical analysis of the feature importance scores. Our procedure of evidence-specific feature selection in Alg. 3.3 and 3.4, based on using existing algorithms for a generative setting and substitute local conditional models $\widehat{P}(X_\gamma \mid \boldsymbol{E}, u_\gamma)$ for the marginals $P(X_\gamma)$ needed by structure learning algorithms is in general a heuristic. However, because of the connections to the generative structure learning, there is a natural special case when our heuristic is optimal. Namely, when the local conditional models $\widehat{P}(X_\gamma \mid \boldsymbol{E}, u_\gamma)$ are perfect, meaning that true conditional probabilities are recovered: $\widehat{P}(X_\gamma \mid E, u_\gamma) = P(X_\gamma \mid E)$, it follows that the optimal evidence-specific structure will be recovered (up to the quality guarantees of the underlying structure learning algorithm).

However, with finite datasets and the need for regularization, the local models are typically imperfect. Moreover, as we discussed in section 3.3.1, for the same evidence value $\boldsymbol{E}$ the local estimates $\widehat{P}(X_\gamma \mid \boldsymbol{E}, u_\gamma)$ and $\widehat{P}(X_\beta \mid \boldsymbol{E}, u_\beta)$ may not agree on the conditional distributions $\widehat{P}(X_\gamma \cap X_\gamma \mid \boldsymbol{E})$, violating the assumption that most structure learning algorithm for the generative case rely on and introducing another source of suboptimality for evidence-specific structures. It is therefore desirable to both characterize the difference between the log-likelihood of a structure $\mathbb{T}$ calculated using the local approximate conditionals $\widehat{P}(X_\gamma \mid \boldsymbol{E}, u_\gamma)$ and the log-likelihood of a full discriminative model (3.15) with the same structure.

Evidence-specific max-margin models. As one can see from the experiments in section 3.6.2, adopting the max-margin objective to learn the feature weights w instead of optimizing conditional log-likelihood of the model (3.15) has not resulted in accuracy improvements for our evidence-specific approach. Such behavior is in sharp contrast with the results of Taskar et al. (2003) and Zhu and Xing (2009) for high-treewidth models with fixed structure, where weights learned by maximizing the margin resulted in significantly better prediction accuracy than weights optimizing the conditional log-likelihood (i.e., the CRF weights). The reason for such a discrepancy remains an open question that will have to be resolved in order to realize the gains in computational efficiency of our evidence-specific approach compared to the high-treewidth max-margin models without sacrificing accuracy. Here, we list some of the possible hypothesis that need to be checked:

1. The representational power of evidence-specific models of treewidth 1, which were used in all our experiments is inherently limited, and feature weights learned via maximizing conditional log-likelihood achieve the optimal prediction accuracy of evidence-specific trees. If true, this hypothesis implies that switching to junction trees with higher treewidth, by replacing Chow-Liu algorithm as parameter $\mathcal{A}_S$ of Alg. 3.3 and 3.4 by, for example, the approach of Shahaf et al. (2009) will lead to larger improvements in accuracy for max-margin feature weights compared to feature weights maximizing the conditional log-likelihood.

2. In addition to feature weights learning, the feature selection procedure (algorithm $\mathcal{A}_S$ in the notation of Alg.. 3.3 and 3.4) also needs to be adjusted to achieve the benefits of max-margin learning. Currently, the general framework for evidence-specific feature selection of Alg. 3.3 and 3.4 adapts structure learning algorithms that aim to maximize the *likelihood* of a structure in the generative setting. By replacing the marginals $P(X_\gamma)$ with approximate conditionals $\widehat{P}(X_\gamma \mid \boldsymbol{E})$, we arrive at

approaches that aim to maximize the approximate conditional likelihood of a structure for $E = \boldsymbol{E}$.

Observe that replacing the conditional likelihood with structured margin for *feature weight learning* on line 3 of Alg. 3.2 does not affect the *structure selection* stage on line 2, which still tries to maximize structure likelihood. It follows that in the ESS-CRF approach both structure selection and weights learning aim to maximize the same objective (conditional likelihood), in the current ESS-M3N approach the two stages aim to maximize different objectives. It is possible that such a mismatch of objectives on different stages leads to a penalty on the resulting accuracy. If the conflict of objectives is a problem, it should be possible to achieve higher accuracy with max-margin weights learning with a better evidence-specific feature selection approach, even keeping the space of evidence-specific structures to junction trees of treewidth 1 as in our experiments.

Unfortunately, to our knowledge, there are no approaches for learning the evidence-independent structure of max-margin linear models that either rely on the low-dimensional conditionals $\widehat{P}(X_\gamma \mid E)$ or guarantee low treewidth of the resulting structure. Therefore the general framework of Alg. 3.3 and 3.4 that works for the conditional random fields cannot readily accommodate existing algorithm $\mathcal{A}_S$ for learning the structure of the max-margin models for the lack of suitable algorithm. The only approach we are aware of is L_1 regularization of weights in the full max-margin QP of Taskar et al. (2003), which lacks both local structure and treewidth guarantees that our current evidence-specific framework needs. Therefore, a suitable approach for learning the structure of max-margin linear models will need to be developed, probably using the generalizations of max-margin models by Zhu and Xing (2009), although they also focus on sparsity of the resulting structures, but not treewidth.

An alternative, more heuristic, approach to structure selection for max-margin models would be to retain as basic components the structure learning algorithms that aim to maximize the likelihood of the generative structure, such as Chow-Liu algorithm, but change the junction tree clique score to better relate to the max-margin objective of the full ESS-CRF model. For example, instead of optimizing the parameters u of the local conditional density estimators $\widehat{P}X_\gamma \mid E, u_\gamma$ to maximize the likelihood of the local models, one can aim to maximize the margin of the local models. Moreover, instead of clique likelihood or mutual information between clique variables, one can use the margin of the locally optimal assignment $\mathbf{X}^*_\gamma(E, u) = \arg\max_{X_\gamma} \widehat{P}(X_\gamma \mid E, u_\gamma)$ as the clique quality measure. Other choices of local scores and models are possible. Moreover, any theoretical advances linking the clique scores computed via local models $\widehat{P}(X_\gamma \mid E, u_\gamma)$ with the benefits of having the clique X_γ in the full evidence-specific model (3.15) for the conditional log-likelihood setting, will likely also yield insights into appropriate clique scores for the max-margin setting.

Generalizing to other classes of tractable models. The evidence-specific CRF approach of this chapter relies in a substantial way on low-treewidth models to achieve efficient exact inference and parameter learning. Low treewidth models are not the only model class where exact inference is tractable. In some high-treewidth models exact inference can be also done efficiently by exploiting context-specific independence (Boutilier et al., 1996) and determinism. Formulations of high-treewidth PGMs where inference is tractable include AND/OR trees (Dechter and Mateescu, 2007), also known as feature trees (Gogate et al., 2010), and arithmetic circuits (Lowd and Domingos, 2008). There exist approaches for learning the structure of high-treewidth tractable models in the generative propositional setting: Gogate et al. (2010) and Lowd and Domingos (2008) have shown that such models can be competitive with low-treewidth models both in approximation accuracy and computational efficiency. It is therefore desirable to extend our evidence-specific approach to support tractable high-treewidth models. However, such a

generalization is not straightforward.

The key difference of tractable high-treewidth tractable models from thin junction trees is that high-treewidth models exploit the fine-grained structure of values *within a feature* to achieve tractability. For example, while in a thin junction tree a feature $f_\gamma(X_\gamma)$ can be an arbitrary function of X_γ, in a feature tree $f_\gamma(X_\gamma)$ has to have a form

$$f_\gamma(X_\gamma) = f_\gamma^{(1)}(X_\gamma^{(1)}) \times \cdots \times f_\gamma^{(m)}(X_\gamma^{(m)}) \text{ such that } X_\gamma^{(i)} \cap X_\gamma^{(j)} = \emptyset. \tag{3.31}$$

As a result, algorithms that learn high treewidth tractable structures, such as that of Gogate et al. (2010), are able to work with features over large scopes X_γ in an efficient way by exploiting the within-feature structure.

To guarantee exact tractable inference in a high-treewidth model, in addition to introducing the structure within individual features such as (3.31), it is also necessary to place joint constraints on the structure of different features. For instance, in feature trees (Gogate et al., 2010), the subfeatures $f_\gamma^{(i)}(X_\gamma^{(i)})$ of *all* features $f_\gamma(X_\gamma)$ have to form a hierarchy such that any two features can only share the top several subfeatures of a hierarchy:

$$\forall f_\gamma(X_\gamma), f_\beta(X_\beta) \text{ it holds that } \exists m \text{ s.t. } X_\gamma \cap X_\beta = \cup_{i=1}^{m} X_\gamma^{(i)}) \text{ and } f_\gamma^{(i)}(X_\gamma^{(i)}) = f_\beta^{(i)}(X_\beta^{(i)}) \text{ for } i = \overline{1, m}.$$

In other words, once two features diverge going down the subfeature hierarchy, they cannot share any lower-level subfeatures. As a result of the restrictions necessary to maintain tractability in high-treewidth models, features over larger sets of variables X_γ may be needed to represent a given distribution. For example, to represent in a generative setting a distribution that factorizes according to a junction tree of treewidth k, a feature tree with features over up to $k \log |X|$ variables, i.e. a factor of $\log |X|$ larger than the junction tree features, may be needed (Dechter and Mateescu, 2007). Even though efficient exact inference is available with high-treewidth tractable models during both training and testing, learning a weight for every high-dimensional composite feature, as ESS-CRF approach of this chapter does for the features of low-treewidth models, is certain to result in overfitting, because the number of possible high-dimensional features grows quickly with the feature size. A similar kind of problem arises with learning arithmetic circuits (Lowd and Domingos, 2008), where the leaves of an arithmetic circuit tree are weighted, but individual edges are the equivalent of elementary features.

There are several ways to reduce overfitting with high-treewidth evidence-specific models. First, one can restrict the maximum size of composite features (3.31), which would automatically reduce the feature space. Second, one can restrict the set of features that are considered at test time to only those features that occurred often enough during training time, analogously to our approach in the experiments on propositional data in section 3.6.1. Such a restriction will also require modifications to the feature tree learning algorithm (Gogate et al., 2010) to work in the restricted feature space. Finally, a more challenging, but potentially promising, direction is to learn the weights of subfeatures $f_\gamma^{(i)}(X_\gamma^{(i)})$ and combine the component weights to obtain the weight for the composite feature $f_\gamma(X_\gamma)$. Such an approach would automatically reduce overfitting of the evidence-specific approach to the level of low-treewidth ESS models, as long as the scope size $|X_\gamma^{(i)}|$ of subfeatures $f_\gamma^{(i)}(X_\gamma^{(i)})$ is bounded. However, it remains an open question whether an accurate decomposition of the optimal weight for $f_\gamma(X_\gamma)$ into subfeature weights exists in a discriminative setting, and what form such a decomposition should take.

Part II

Query-Specific Inference in High-Treewidth Graphical Models

Chapter 4

Query-Specific Belief Propagation

In part I of this thesis, we have argued that learning high-quality low-treewidth probabilistic graphical models from data in both generative and discriminative settings is often preferable to working with high treewidth models that require approximate inference. However, in some applications one cannot avoid using a high-treewidth model.

One setting where a high-treewidth may have higher accuracy is when **individual features** of the model are **not very informative in isolation**, but taken jointly, the connectivity pattern of the model may reveal more information. For example, in the WebKB hypertext classification model (c.f. section 3.6.2) every hyperlink is represented with a simple binary feature, making every link equally important from the perspective of the model. The ability of a high-treewidth model to extract additional information from the connectivity pattern of equally important links is a possible, although not certain, explanation of the accuracy advantage of high-treewidth max-margin Markov networks over our low-treewidth ESS-CRF approach.

Another setting where high-treewidth models are preferable are those where the features simply **encode** the **precisely known laws of** local **interaction** between the variables. For example, in a protein folding problem (Yanover and Weiss, 2002), the features simply encode the laws of physics describing the electromagnetic interactions between atoms (in fact, for this problem no learning is necessary, and only probabilistic inference at test time is required). In graphical models for electric circuit diagnosis (Borgelt and Kruse, 2005), the model structure has to reflect the structure of the circuit in order for the results to be interpretable.

In this chapter, we address the problem of probabilistic inference in high-treewidth graphical models, which is useful for the cases when learning an alternative model with tractable structure is impossible or undesirable. More specifically, we consider the setting where the user is only interested in a small number of the unknown variables $Q \in X$ and the rest of X are nuisance variables. Applications with a large number of nuisance variables occur quite often. For example, in activity recognition in a smart home (Pentney et al., 2006), to decide when to turn on the air conditioning, the system may be interested only in the likely time for the person to arrive home. The exact action sequences leading to the person arriving home then need to be marginalized out. For patient monitoring (Beinlich et al., 1988), the only query variable may be whether the patient needs immediate attention of the medical staff, and all the possible complications that require a nurse to intervene need to be marginalized out.

For the setting where a only relatively small set of variables of a PGM is of immediate interest to the

user, we build on a state of the art residual belief propagation algorithm (RBP, Elidan et al., 2006) for approximate inference. We introduce an approach that focuses the computation on the areas of the model that have large influence on the query marginals. Our query-specific inference approach takes into account both the structure and parameters of the model to adjust the computation prioritization, resulting in significantly faster convergence.

4.1 Factor Graphs and Belief Propagation

We briefly review a particular formalism of probabilistic graphical models, namely factor graphs (for details, see Koller and Friedman, 2009), and loopy belief propagation, an approximate inference algorithm.

Probabilistic graphical models represent *factorized* probability distributions, where the distribution $P(X)$ over a large set of random variables X is decomposed into a product of low-dimensional functions:

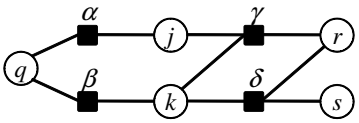
$$P(X) = \frac{1}{Z} \prod_{\psi_\alpha \in \mathcal{F}} \psi_\alpha(X_\alpha), \tag{4.1}$$

where every $X_\alpha \subseteq X$ is a subset of X (typically, $|X_\alpha| \ll |X|$), $\psi_\alpha \geq 0$ are **factors** and Z is the normalization constant. A probabilistic graphical model is a combination of the factorized distribution (4.1) and graphical structure induced by the factors ψ_α. Several alternative PGM formulations exist, depending on the type of graphs being used. Here, we use **factor graphs:** given the factorized distribution (4.1), the corresponding factor graph is a bipartite graph $(\{X, \mathcal{F}\}, \mathbb{T})$ with one node for every factor $\psi_\alpha \in \mathcal{F}$ and every random variable $x_i \in X$, and an undirected edge $(\alpha - i)$ for every pair of ψ_α and x_i such that $x_i \in X_\alpha$ (see Fig. 4.1a for an example).

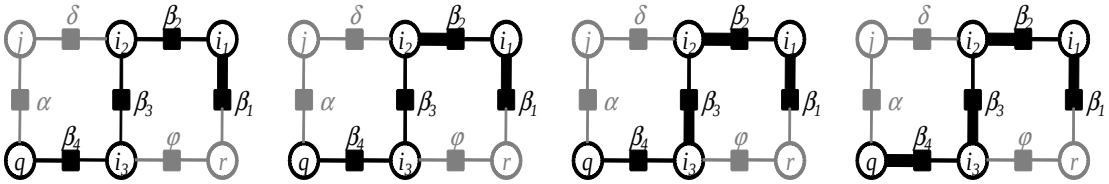
Probabilistic inference problem. The central problem of this chapter is, given the factor graph G and query variables Q to find the marginal distribution $P(Q)$. Unfortunately, this problem of *probabilistic inference* is known to be #P-complete in the exact case and NP-hard in the approximate case (Roth, 1996). In practice, approximate inference algorithms that allow for moderate computational complexity, such as belief propagation (Pearl, 1988), Gibbs sampling (Geman and Geman, 1984) and variational inference (Wainwright and Jordan, 2008) are typically used. Because approximate inference is intractable in the worst case, these algorithms typically lack meaningful guarantees on the results quality. However, in practice approximate inference algorithms often produce quite accurate results, which makes these approaches very useful. One can see that approximate inference approaches can be improved in two different directions: increasing the quality of the results at convergence (possibly at the expense of computational complexity) and speeding up the convergence itself while keeping the end results the same. In this chapter, we focus on the latter objective and address the problem of improving the convergence speed of belief propagation.

Loopy belief propagation (LBP), first proposed by Pearl (1988), is an algorithm for approximate inference in factor graphs, which has been very successful in practice (McEliece et al., 1998; Yedidia et al., 2003). Let Γ_α be the set of neighbors of node α in a factor graph. LBP is an iterative algorithm that repeatedly updates the messages $m_{\alpha-i}$ from factors ψ_α to their respective variables x_i until convergence, as follows:

$$m_{\alpha-i}^{(t+1)}(x_i) = \sum_{\boldsymbol{X}_{\alpha \setminus i}} \left(\psi_\alpha \left(\boldsymbol{X}_{\alpha \setminus i}, x_i \right) \times \prod_{j \in \Gamma_\alpha \setminus i} \frac{\tilde{\mathbf{P}}^{(t)}(\boldsymbol{x}_j)}{m_{\alpha-j}^{(t)}(\boldsymbol{x}_j)} \right), \tag{4.2}$$



(a) A factor graph



(b) 0 updates (c) 1 update (d) 2 updates (e) 3 updates

Figure 4.1: An example factor graph (a); A simple path $\pi = (\beta_1 - i_1 - \cdots - \beta_k - q)$ (solid black) in a factor graph (b)-(e). Thick edges indicate messages that become functions of $\nu_{\beta_1 - i_1}$ after the corresponding number of LBP updates.

where $\tilde{P}(\cdot)$, called **beliefs**, are the estimates of single-variable marginals defined as

$$\tilde{P}(x_i) \propto \prod_{\alpha \in \Gamma_i} m_{\alpha - i}(x_i). \tag{4.3}$$

We will denote as m the vector of messages $m_{\alpha - i}$ corresponding to all the edges $(\alpha - i) \in \mathbb{T}$. One can interpret $m_{\alpha - i}$ as an element of m with index $(\alpha - i)$. Loopy BP approximates the joint distribution $P(X)$ as a product of single-variable marginals: $\tilde{P}(X) \equiv \prod_{x_i \in X} \tilde{P}(x_i)$.

LBP is guaranteed to find an accurate solution in some special cases: graphs without cycles (converges to the exact marginals; see Weiss, 2000), graphs with a single cycle and Gaussian models (always converges on single-cycle models, finds the true maximum a posteriori assignment whenever messages m converge; see Weiss and Freeman, 2001b). For general factor graphs, however, there are no guarantees. Although at least one fixed point $\boldsymbol{m}^*$ for BP updates is guaranteed to exist for models with strictly positive factors (Yedidia et al., 2005) the fixed point beliefs may be arbitrary far from the true marginals. Moreover, a model may have multiple BP fixed points. Finally, the convergence to any fixed point is not guaranteed as fixed points may be unstable (Mooij and Kappen, 2005).

Despite the theoretical limitations, LBP beliefs are often successfully used instead of true marginals in applications (Yanover and Weiss, 2002). In this chapter, we assume that LBP converges and do not address the question of quality of the resulting beliefs. Instead, we are concerned with speeding up the convergence of LBP: our goal is to recover $\tilde{P}(Q)$ at a fixed point $\boldsymbol{m}^*$ of LBP at the minimal computation cost. While this is different from recovering the true query marginal $P(Q)$, the fact that LBP fixed point beliefs have adequate accuracy for many applications makes improving LBP efficiency an important problem by itself. In the discussion, we will assume that the fixed point $\boldsymbol{m}^*$ is unique, but nothing in the proposed algorithms depends on the uniqueness of the fixed point - our approach is applicable in practice to arbitrary models.

Log-space messages and updates. It is often more convenient to analyze the behavior of belief propagation if the messages are represented in the logarithmic space:

$$\nu_{\alpha - i} \equiv \log m_{\alpha - i}.$$

Same as with the vector m of all messages, we will denote as ν the vector of logarithms of all messages. Belief propagation updates (4.2) and single-variable belief computation (4.3) can be directly applied to log-space messages ν as

$$\nu_{\alpha-i}^{(t+1)}(x_i) = \log \sum_{\boldsymbol{X}_{\alpha\setminus i}} \left(\psi_\alpha \left(\boldsymbol{X}_{\alpha\setminus i}, x_i \right) \exp \sum_{j \in \Gamma_\alpha \setminus i} \left(\tilde{\boldsymbol{\Upsilon}}^{(t)} \left(\boldsymbol{x}_j \right) - \nu_{\alpha-j}^{(t)}(\boldsymbol{x}_j) \right) \right), \tag{4.4}$$

where $\tilde{\boldsymbol{\Upsilon}}\left(\cdot\right)$ is the log-belief

$$\tilde{\boldsymbol{\Upsilon}}\left(x_i\right) \equiv \log \tilde{P}\left(x_i\right) = \sum_{\alpha \in \Gamma_i} \nu_{\alpha-i}(x_i) - \log \sum_{\boldsymbol{x}_i} \exp \sum_{\alpha \in \Gamma_i} \nu_{\alpha-i}(\boldsymbol{x}_i). \tag{4.5}$$

Residual belief propagation (RBP). An effective and popular technique for improving the computational efficiency of belief propagation is updates scheduling. In the standard LBP, on every iteration all the messages are recomputed and updated per the Equation 4.4. However, often many messages change very little between two iterations. Updating such messages on every iteration wastes computation. To improve efficiency, residual BP (Elidan et al., 2006) updates only one message per iteration, namely the one that would have changed the most under LBP updates.

More concretely, RBP maintains two messages for every edge, a current message $\nu_{\alpha-j}^{(t)}$ and a new message, which is the result of a BP update (4.4) given all the other current messages:

$$\widehat{\nu}_{\alpha-i}^{(t)}(x_i) = \log \sum_{\boldsymbol{X}_{\alpha\setminus i}} \left(\psi_\alpha \left(\boldsymbol{X}_{\alpha\setminus i}, x_i \right) \exp \sum_{j \in \Gamma_\alpha \setminus i} \left(\tilde{\boldsymbol{\Upsilon}}^{(t)} \left(\boldsymbol{x}_j \right) - \nu_{\alpha-j}^{(t)}(\boldsymbol{x}_j) \right) \right). \tag{4.6}$$

The difference between the old and new messages is called the **residual** $r_{\alpha-i}$:

$$r_{\alpha-i} \equiv \|\widehat{\nu}_{\alpha-i}^{(t)} - \nu_{\alpha-i}^{(t)}\|. \tag{4.7}$$

On iteration $t+1$, RBP updates only the message with the *largest residual.* That is $\nu_{\alpha-i}^{(t+1)} = \widehat{\nu}_{\alpha-i}^{(t)}$ is set for the edge $(\alpha - i) = \arg\max_{(\alpha-i)\in\mathbb{T}} r_{\alpha-i}$ and the old message $\nu_{\beta-j}^{(t+1)} = \nu_{\beta-j}^{(t)}$ is kept for all the other edges. After updating $\nu_{\alpha-i}$, the new messages $\widehat{\nu}_{\beta-j}^{(t+1)}$ only have to be recomputed if $\beta \in \Gamma_i$.

Intuitively, by always updating the message with the largest residual, RBP achieves larger change in beliefs per unit computation and thus traces the trajectory towards the fixed point in the beliefs space faster than the standard loopy BP. In this chapter, we add the notion of query to residual BP so as to concentrate the computation on the parts of the model that are more important to the query and thereby further increase computational efficiency.

4.2 Measuring Importance of Messages to the Query

In this section, we introduce the basic measure of edge importance with respect to a query Q that quantifies the connection between changes in the message for a given edge and the resulting future changes of belief over Q. Edge importance forms the basis of our query-specific inference approach. For simplicity, we

first develop the case where the query consists of exactly one variable: $Q \equiv \{x_q\}$. Generalization to multivariable queries is then presented in Section 4.2.2.

Let us consider a case when residual BP update scheduling fails. This example will both illustrate the need for an edge importance measure and highlight the properties of the model that are useful for determining edge importance. Consider the factor graph in Fig. 4.1a. Assume variable x_q is the query and on iteration t message residuals are such that $r_{\delta-s} > r_{\gamma-j}$. Which of the two messages should be updated next? Although $\nu_{\delta-s}$ has a larger residual, updating it will only change the belief over the irrelevant variable x_s and will not affect any other messages or beliefs in the future. Updating $\nu_{\gamma-j}$, on the other hand, would entail recomputing $\widehat{\nu}^{(t)}_{\alpha-q}$ and may change the query belief $\tilde{P}(x_q)$ on the next step. Thus, one should prefer updating $\nu_{\gamma-j}$ over $\nu_{\delta-s}$. However, the standard BP algorithms have no notion of the query Q and are unable to prioritize message updates by their importance to the convergence of $\tilde{P}(Q)$: RBP will update the irrelevant $\nu_{\delta-s}$. One can see that edge $(\gamma - j)$ should receive a boost in priority compared to edge $(\delta - s)$ based on the *model structure* and the respective location of those edges with respect to the query variable x_q. Moreover, to quantify the difference between different paths to the query in the PGM graph, *model parameters* need to be taken into account as well.

One can see from the LBP updates (4.4) that a change in message $\nu_{\alpha-i}$ propagates through the graph with consecutive LBP iterations: after one update, only the immediate neighbors of x_i are affected, then their respective neighbors and so on. For example, in Fig. 4.1a, a change in message $\nu_{\delta-r}$ after one LBP update will affect $\nu_{\gamma-k}$ and $\nu_{\gamma-j}$, after the next update – $\nu_{\alpha-q}$, $\nu_{\beta-q}$, $\nu_{\delta-r}$ and $\nu_{\delta-s}$ and so on. Notice that the change in $\nu_{\delta-r}$ will impact the beliefs $\tilde{P}(x_q)$ via different paths, for example $\delta - r - \gamma - k - \beta - q$ and $\delta - r - \gamma - j - \alpha - q$. Let us first quantify the importance of every such single path to the query belief $\tilde{P}(x_q)$.

Consider a directed simple path $\pi = (\beta_1 \to i_1 \to \cdots \to \beta_k \to q)$ from factor β_1 to the query variable q in the full factor graph $G = (\{X, \mathcal{F}\}, \mathbb{T})$. Fix the messages $\nu_{-\pi}$ corresponding to all the edges not in π. Let us repeatedly apply LBP updates (4.4) to the messages corresponding to the edges in π. Then one can see that $\nu_{\beta_2-i_2}$ is a function of $\nu_{\beta_1-i_1}$, $\nu_{\beta_3-i_3}$ is a function of $\nu_{\beta_2-i_2}$ and so on. After $k-1$ LBP updates, ν_{β_k-q} becomes a function of $\nu_{\beta_1-i_1}$ (see Fig. 4.1). Denote this function $\nu_{\beta_k-q} = F_\pi(\nu_{\beta_1-i_1}, \nu_{-\pi})$. The sensitivity of ν_{β_k-q} to changes in $\nu_{\beta_1-i_1}$ due to dependencies along path π is thus determined by the following derivative (for notational convenience, denote $i_k \equiv q$):

$$\left.\frac{\partial \nu_{\beta_k-q}}{\partial \nu_{\beta_1-i_1}}\right|_\pi \equiv \frac{\partial F_\pi}{\partial \nu_{\beta_1-i_1}} = \prod_{d=1}^{k-1} \frac{\partial \nu_{\beta_{d+1}-i_{d+1}}}{\partial \nu_{\beta_d-i_d}}. \tag{4.8}$$

By the mean value theorem, we can bound from above the change in ν_{β_k-q} due to the change in $\nu_{\beta_1-i_1}$, provided that all the messages not in π are fixed:

$$\begin{aligned}
\left\| \Delta\nu_{\beta_k-q}|_\pi \right\| &\leq \|\Delta\nu_{\beta_1-i_1}\| \cdot \sup_{\nu_{-\pi}} \left\| \frac{\partial F_\pi}{\partial \nu_{\beta_1-i_1}} \right\| \\
&= \|\Delta\nu_{\beta_1-i_1}\| \cdot \sup_{\nu_{-\pi}} \left\| \prod_{d=1}^{k-1} \frac{\partial \nu_{\beta_{d+1}-i_{d+1}}}{\partial \nu_{\beta_d-i_d}} \right\| \\
&\leq \|\Delta\nu_{\beta_1-i_1}\| \cdot \prod_{d=1}^{k-1} \sup_{\nu_{-\pi}} \left\| \frac{\partial \nu_{\beta_{d+1}-i_{d+1}}}{\partial \nu_{\beta_d-i_d}} \right\|.
\end{aligned} \tag{4.9}$$

Observe that the upper bound on the derivative magnitude in the last inequality decomposes into a product of local terms, each depending only on one factor (namely $\psi_{\beta_{d+1}}$ for d-th component), which makes

tractable computation possible. We adopt the decomposable upper bound in (4.9) as a measure of influence of path π on the query variable:

Definition 47. The **sensitivity strength** of a directed **path** $\pi = (\beta_1 \to i_1 \to \cdots \to q)$ is

$$sensitivity(\pi) = \prod_{d=1}^{k-1} \sup_{\nu_{-\pi}} \left\| \frac{\partial \nu_{\beta_{d+1}-i_{d+1}}}{\partial \nu_{\beta_d - i_d}} \right\|. \tag{4.10}$$

Complexity of computing the suprema in (4.10) differs depending on the choice of norm $\|\cdot\|$. Fortunately, Mooij and Kappen (2007) have shown that for a particular norm, namely the *log-dynamic range*

$$\|\nu_{\alpha-j}\| \equiv \frac{1}{2}\left(\max_{x_j} \nu_{\alpha-j}(x_j) - \min_{x_j} \nu_{\alpha-j}(x_j)\right), \tag{4.11}$$

the suprema in (4.10) can be computed in closed form:

$$\sup_{\nu} \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\| = \max_{\boldsymbol{X}_{\alpha\setminus i}, \boldsymbol{X}'_{\alpha\setminus i}, \boldsymbol{x}_i \neq \boldsymbol{x}'_i, \boldsymbol{x}_j \neq \boldsymbol{x}'_j} \tanh\left(\frac{1}{4}\log \frac{\psi_\alpha\left(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{X}_{\alpha\setminus ij}\right)\psi_\alpha\left(\boldsymbol{x}'_i, \boldsymbol{x}'_j, \boldsymbol{X}'_{\alpha\setminus i}\right)}{\psi_\alpha\left(\boldsymbol{x}'_i, \boldsymbol{x}_j, \boldsymbol{X}_{\alpha\setminus i}\right)\psi_\alpha\left(\boldsymbol{x}_i, \boldsymbol{x}'_j, \boldsymbol{X}'_{\alpha\setminus i}\right)}\right). \tag{4.12}$$

Here, we adopt the log-dynamic range norm (4.11) and use (4.12) to compute the suprema in (4.10).

For a given simple path π, from (4.9) it follows that, if the messages $\nu_{-\pi}$ are kept constant and $\nu_{\beta_1 - i_1}$ is changed by Δ, the message $\nu_{\beta_k - q}$ will eventually change by at most $\Delta \times sensitivity(\pi)$. In loopy models, however, there are typically infinitely many directed paths starting with $\beta_1 - i_1$ and ending in q (taking paths with loops into account). Therefore, the effect of changing $\nu_{\beta_k - q}$ on the $\tilde{P}(x_q)$ will be eventually transferred along many paths. As a basic notion of importance of any message $\nu_{\alpha-i}$ to the query, we adopt the sensitivity of the *single strongest* directed *path* from an edge $(\alpha \to i)$ to the query q :

Definition 48. Given the query variable x_q, the **maximum sensitivity importance value** of an edge $(\alpha - i)$ is defined to be

$$max\text{-}sensitivity(\alpha - i, q) \equiv \max_{\pi \in \Pi(\alpha - i, q)} sensitivity(\pi),$$

where $\Pi(\alpha - i, q)$ is the set of all directed paths that start with $(\alpha \to i)$ and end in q.

One can think of the maximum sensitivity importance as a first-order approximation of influence of a message on the query, discarding all of the smaller terms corresponding to paths with weaker dependencies.

4.2.1 Efficiently Computing Edge Importance

Notice that computing the maximum sensitivity importance of an edge $\alpha - i$ involves a maximization over all possible paths from x_i to the query variable x_q in the factor graph $G = (\{X, \mathcal{F}\}, \mathbb{T})$. Clearly, it is impossible to explicitly enumerate and maximize over all possible paths in all but the simplest (e.g., forest-structured) graphical models, because of the number of such paths is infinite in general. Fortunately, it is possible to find a path with the largest sensitivity efficiently for the same reason it is possible to find a shortest path between two nodes in a graph efficiently: path sensitivity is *monotonic* in the sense that

Algorithm 4.1: Edge importance computation

Input: Factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, query $Q \in X$.

1 $\mathbb{L} = \emptyset$ is priority queue, $\rho_{\alpha-i}$ is edge priority

2 **foreach** $(\alpha - i) \in \mathbb{T}$ **do** add $(\alpha - i)$ to $\mathbb{L}$ with priority $\rho_{\alpha-i} = \begin{cases} 1, & \text{if } i \in Q \\ 0 & \text{otherwise} \end{cases}$

3 **while** $\mathbb{L} \neq \emptyset$ **do**

4 denote $(\alpha - i)$ to be the top of $\mathbb{L}$

5 set $w_{\alpha-i} \leftarrow \rho_{\alpha-i}$, remove $(\alpha - i)$ from $\mathbb{L}$

6 **foreach** $j \in \Gamma_\alpha \setminus i$ **foreach** $\beta \in \Gamma_j \setminus \alpha$ **such that** $(\beta - j) \in \mathbb{L}$ **do**

7 $\rho_{\beta-j} \leftarrow \max\left(\rho_{\beta-j}, \rho_{\alpha-i} \cdot \sup_\nu \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\| \right)$

8 update the position of $(\beta - j)$ in $\mathbb{L}$ to match priority $\rho_{\beta-j}$

9 **return** $W = \{w_{\alpha-i} \mid (\alpha - i) \in \mathbb{T}\}$ - importance values for all the edges

for any path π' that is a subpath of π it holds that $sensitivity(\pi) < sensitivity(\pi')$. The monotonicity property follows directly from Definition 47 and the fact that from (4.12), for any factor ψ_α it holds that $\sup_\nu \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\| \in [0, 1)$. Therefore, directly analogous to Dijkstra's shortest path algorithm (Dijkstra, 1959), which expands graph edges in the order of their shortest-path distance from the starting point, we can construct an edge importance computation algorithm (Alg. 4.1) that expands edges in the order of decreasing importance and computes the exact importance values for every edge efficiently.

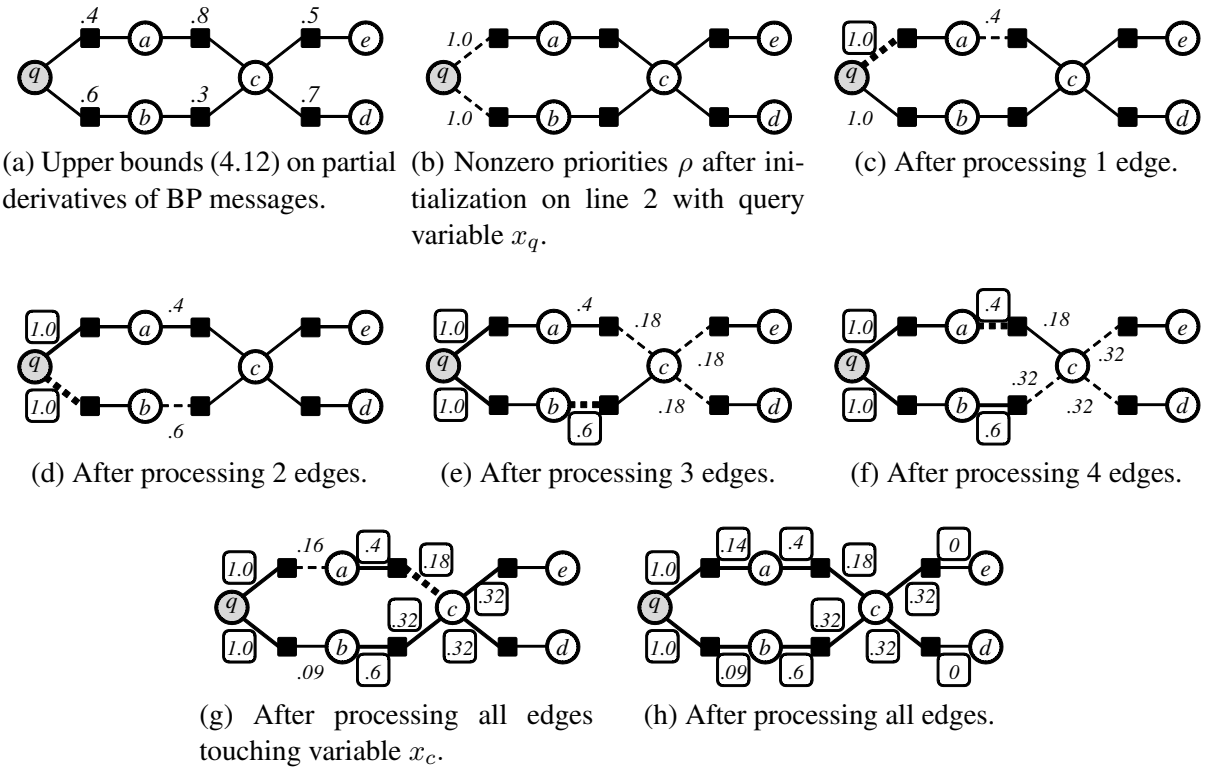
Proposition 49. *Alg. 4.1 computes the exact maximum sensitivity importance values of Def. 48 for every edge $\alpha - i$: on line 5 of Alg. 4.1 it holds that $w_{\alpha-i} = max\text{-}sensitivity(\alpha - i, q)$.*

Proof sketch: By construction of edge priorities, at any time during execution of Alg. 4.1 for every edge $\alpha - i$ there exists a path $\pi = (\alpha \to i \to \cdots \to q)$ such that $\rho_{\alpha-i} = sensitivity(\pi)$. Thus, the only possible failure mode of Alg. 4.1 is to get $w_{\alpha-i} < max\text{-}sensitivity(\alpha - i, q)$. The proof that such a failure is also impossible is by contradiction.

Denote $\mathbb{T}'$ the set of edges for which Alg. 4.1 returns $w_{\alpha-i} < max\text{-}sensitivity(\alpha - i, q)$. Let $\alpha - i$ be the edge from $\mathbb{T}'$ with the largest true importance $w_{\alpha-i}$. Consider the moment when $\alpha - i$ reaches the top of the priority queue $\mathbb{L}$ and is processed on line 4. The resulting importance weights $w_{\beta-j}$ for all $\beta - j$ still in $\mathbb{L}$ can be no greater than $w_{\alpha-i}$ because $\sup_\nu \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\| < 1$ and currently $\rho_{\beta-j} \leq \rho_{\alpha-i}\ \forall (\beta - j) \in \mathbb{L}$. Therefore, all edges $\beta - j$ with $w_{\beta-j} \geq w_{\alpha-i}$ have already been expanded by Alg. 4.1. Denote $\pi^* = (\alpha \to i \to \gamma \to k \cdots \to q)$ to be the largest sensitivity path from $\alpha - i$ to q. Because $sensitivity(\gamma \to k \cdots \to q) > sensitivity(\pi^*)$, it follows that $w_{\gamma-k} > w_{\alpha-i}$ and thus the edge $\gamma - k$ has already been expanded by Alg. 4.1 with $\rho_{\gamma-k} = max\text{-}sensitivity(\gamma - k, q)$. Therefore, the correct value $\rho_{\alpha-i} = w_{\alpha-i}$ should have been set on line 7 of Alg. 4.1 during the expansion of $\gamma - k$, a contradiction.□

Proposition 50. *Suppose every factor of the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ contains at most d_f variables and every variable participates in at most d_v factors. Then the complexity of Alg. 4.1 with priority queue organized as Fibonacci heap is $O(|\mathbb{T}|(\log |\mathbb{T}| + d_f d_v))$.*

Proof sketch: Priority queue $\mathbb{L}$ has size $|\mathbb{T}|$, so extracting a highest priority element costs $O(|\mathbb{T}|)$ and updating the priority of an edge costs amortized $O(1)$ (c.f., for example, Cormen et al., 2001, we only need to actually modify the queue if the priority increases). Every edge is expanded on lines 4-5 exactly once, and edge expansion entails updating priorities of at most $d_f d_v$ other edges.□



(a) Upper bounds (4.12) on partial derivatives of BP messages.

(b) Nonzero priorities ρ after initialization on line 2 with query variable x_q.

(c) After processing 1 edge.

(d) After processing 2 edges.

(e) After processing 3 edges.

(f) After processing 4 edges.

(g) After processing all edges touching variable x_c.

(h) After processing all edges.

Figure 4.2: A trace of Alg. 4.1 computing maximum sensitivity edge importance values for query variable x_q.
Figure (a): the upper bound (4.12) on the magnitude of the derivative of LBP updates (4.4). The upper bound is the same in both directions for every factor.
Figures (b)-(h): the sequence of edges processed by Alg. 4.1.
Thick dashed line is the edge $(\alpha - i)$ taken from the top of priority queue on line 4. *Thin dashed lines* are the edges $(\beta - j)$ whose priorities are updated on line 7 as $(\alpha - i)$ is processed. *Numbers in solid rectangles* denote the edge importance weights $W_{\alpha - i}$ that are already known to Alg. 4.1 after the respective number of edges $(\alpha - i)$ have been taken off the top of the priority queue $\mathbb{L}$ and processed on lines 4-8. *Numbers without rectangles* are the current priorities of edges in $\mathbb{L}$.

Example: see Fig. 4.2 for an example trace of Alg. 4.1. Observe that it is possible for edges to have zero importance even though those edges are connected to the query if the factor graph is treated as an undirected graph (see edges touching variables e and d in Fig. 4.2). Zero importance reflects the fact that changing the messages on those edges has no way to affect the query beliefs due to the *directions* in which influence spreads and the fact that messages $\nu_{\alpha - i}$ and $\nu_{\alpha - j}$ do not influence each other directly. Variable e, for example, becomes a "dead end" for its incoming message, stopping propagation of any changes in that message to other parts of the model.

4.2.2 Edge Importance for Multi-Variable Queries

Recall the interpretation of maximum sensitivity edge importance weight in Definition 48 for edge $(\alpha - i)$ as the sensitivity of the single strongest path from $(\alpha - i)$ to the query variable x_q. This intuition generalizes immediately to multi-variable queries Q by maximizing over all paths to the query set instead of all paths to one variable:

Definition 51. Given the set Q of query variables, the **maximum sensitivity importance value** of an edge $(\alpha - i)$ is defined to be

$$max\text{-}sensitivity(\alpha - i, Q) \equiv \max_{x_q \in Q} \max_{\pi \in \Pi(\alpha - i, q)} sensitivity(\pi). \tag{4.13}$$

where $\Pi(\alpha - i, q)$ is the set of all directed paths that start with $(\alpha \to i)$ and end in q.

The advantage of the generalization of Def. 51 is the fact that the maximum sensitivity importance values for multi-variable queries can be computed as efficiently as for single-variable queries. Indeed, the only change needed to adapt Alg. 4.1 to multiple variable queries is to replace the condition $i = q$ with $x_i \in Q$ on line 2.

Proposition 52. *Alg. 4.1 with the initial edge priorities on line 2 set to*

$$\rho_{\alpha - i} = \begin{cases} 1, & \text{if } x_i \in Q \\ 0 & \text{otherwise} \end{cases}$$

computes the exact maximum sensitivity importance values of Def. 51 for every edge $\alpha - i$: on line 5 of Alg. 4.1 it holds that $w_{\alpha - i} = max\text{-}sensitivity(\alpha - i, Q)$.

The proof is almost identical to that of Prop. 49.

4.3 Query-Specific Residual Belief Propagation

In this section, we review the theoretical basis of residual belief propagation (RBP; Elidan et al., 2006), a simple and successful technique for speeding up BP convergence, and show how edge importance weights of Section 4.2 can be naturally integrated in RBP framework to yield a query-specific inference approach.

4.3.1 Residual Belief Propagation

As Elidan et al. (2006) have shown, belief propagation convergence speed is significantly increased if the message updates are prioritized by message residual. Their algorithm, called residual belief propagation (RBP), maintains a priority queue over model edges, with current residuals (4.7) as priorities. On every step, the edge $(\alpha - i)$ with the highest residual is pulled off the top of the queue, and BP update (4.4) is applied for that edge. The residuals for neighbors of x_i are then recomputed.

The residuals quantify the immediate effect of pending updates on the beliefs. Intuitively, prioritizing by the residual lets RBP concentrate on the least converged and more difficult parts of the model. Formally, the prioritization is justified for cases when belief propagation updates (4.4) yield a contraction mapping by the following bound. Under max-norm on the log-message space,

$$\|\nu\|_\infty \equiv \max_{(\alpha - i) \in \mathbb{T}} \|\nu_{\alpha - i}\|,$$

Elidan et al. (2006) show that for any function $g : \nu \to \nu$ that is a contraction mapping with a fixed point ν^*, meaning that $\|g(\nu) - \nu^*\|_\infty \leq \alpha \|\nu - \nu^*\|_\infty$ for some $\alpha < 1$, it holds that

$$\|\nu - \nu^*\|_\infty - \|g(\nu) - \nu^*\|_\infty \geq \frac{1-\alpha}{1+\alpha} \|g(\nu) - \nu\|_\infty .$$

In other words, for any contraction mapping $g(\cdot)$, an update $\nu \leftarrow g(\nu)$ reduces the distance to the fixed point by at least a fixed fraction of the residual. Therefore, in cases when belief propagation updates (4.4) form a contraction mapping, residual belief propagation can be viewed as greedily minimizing the max-norm distance to the fixed point $\|\nu - \nu^*\|_\infty$. Although BP updates do not always define a contraction mapping (for some factor graphs, belief propagation may even have multiple fixed points), Mooij and Kappen (2007) give sufficient conditions for BP updates to form a contraction mapping. Also, BP can be a contraction mapping in the vicinity of a fixed point.

4.3.2 Edge Importance Weights and Query-Specific Inference

In contrast to the standard probabilistic inference setting, where every variable, and consequently every BP message, is equally important, in the query-specific setting only the query marginal $\tilde{P}(Q)$ (equivalently, the log-marginal $\tilde{\boldsymbol{\Upsilon}}(Q)$) is important. Therefore, instead of the global distance to the fixed point $\|\nu - \nu^*\|$ we would like to minimize an analogous norm that only takes query marginals into account. Given a message vector ν, denote $\nu|_Q$ to be ν with all entries zeroed out except for messages directly incoming to the query:

$$\nu|_Q(\alpha - i) = \begin{cases} \nu_{\alpha-i}, & x_i \in Q \\ 0 & \text{otherwise} \end{cases}$$

Then we need to minimize the query-specific norm

$$\|\nu - \nu^*\|_Q \equiv \|\nu|_Q - \nu^*|_Q\|, \tag{4.14}$$

The query-specific norm (4.14) ignores the messages corresponding to non-query variables. Therefore, only updates to messages directly touching the query variables can *immediately* change the distance to the fixed point $\|\nu - \nu^*\|_Q$. On the other hand, in the vast majority of settings it is impossible to arrive at the fixed-point query belief by only applying updates directly impacting the query. The query belief depends on beliefs of non-query variables, so it is necessary to also make non-query messages close to the fixed point, even though (4.14) ignores those messages. It follows that it is impossible to minimize the query-specific (QS) distance from the fixed point by greedily choosing updates that will change the QS distance $\|\cdot\|_Q$ right away. Instead, one has to reason about the impact of a message update *several iterations in the future.* For the case of single-variable queries, the analysis of Section 4.2 provides one way to estimate the eventual impact of an arbitrary message update on the query: the estimate for message $\nu_{\alpha-i}$ is simply $w_{\alpha-i} \cdot r_{\alpha-i}$. It is then natural in the query-specific setting to replace the L_∞ norm used by residual belief propagation with *weighted* L_∞ norm:

$$\|\nu\| \equiv \max_{\alpha-i \in \mathbb{T}} w_{\alpha-i} \|\nu_{\alpha-i}\|. \tag{4.15}$$

The resulting algorithm, which we call query-specific belief propagation (QSBP, Alg. 4.2) prioritizes message updates in the order of their estimated *eventual* impact on the *query* marginals instead of *immediate* impact on *all* the marginals as is the case for the standard RBP. Observe that the changes from RBP, highlighted in the algorithm text, are minimal, and it is easy to modify an existing implementation of RBP to make it query-specific.

Algorithm 4.2: Query-specific belief propagation. Red underlined font denotes differences from RBP.

Input: Factor graph $G = (\{X, \mathcal{F}\}, \mathbb{T})$, query $Q \in X$.

$\mathbb{M}$ is a priority queue

$W \leftarrow$ Alg.4.1(G, Q) (find edge importance values)

foreach $(\alpha - i) \in \mathbb{T}$ initialize the message $\nu_{\alpha-i}$

foreach $(\alpha - i) \in \mathbb{T}$ **do**

  compute $\widehat{\nu}_{\alpha-i}, r_{\alpha-i}$ using (4.6,4.7)

  add $(\alpha - i)$ to $\mathbb{M}$ with priority $w_{\alpha-i} \times r_{\alpha-i}$

while *not converged* **do**

  denote $(\alpha - i)$ to be the top of $\mathbb{M}$

  $\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}$

  set priority of $(\alpha - i)$ in $\mathbb{M}$ to 0

  foreach $\beta \in \Gamma_i \setminus \alpha$ *and* $j \in \Gamma_\beta \setminus i$ **do**

    recompute $\widehat{\nu}_{\beta-j}$ and $r_{\beta-j}$ using Eq. 4.6, 4.7

    set priority of $(\beta - j)$ in $\mathbb{M}$ to $w_{\beta-j} \times r_{\beta-j}$

return $\tilde{P}(x_q)$ for $x_q \in Q$ using Eq. 4.5

Example. To illustrate how QSBP focuses the computation on the most relevant to the query parts of the model, we show in Fig. 4.4 how message update counts are distributed for the two algorithms after the same running time on an image denoising problem. The model in question takes an four-level grayscale image corrupted with Gaussian noise such as Fig. 4.4a, and tries to recover the clean image (see Fig. 4.4b for an example result) by penalizing disagreement between the neighboring pixels. More specifically (see Fig. 4.3 for the factor graph structure for the case of a 3×3 image), the model has two types of variables: for every pixel i there a variable x_i denoting the true color of the pixel that is unknown at test time and needs to be inferred, and a variable y_i that, conditioned on x_i, is distributed normally with $P(y_i \mid x_i) = \mathcal{N}(x_i, 1)$. At test time, the value of y_i is known. There are also two types of factors. First, for every pixel there is a factor $\psi_{x_i y_i} = \mathcal{N}_{x_i,1}(y_i)$, encoding the probability of a noisy observation given the true pixel brightness. Second, for every pair of pixels i, j that are neighbors on the grid there is an agreement factor $\psi_{x_i x_j} = \exp\{-C \cdot \mathcal{I}(x_i \neq x_j)\}$ that penalizes brightness discontinuities ($\mathcal{I}(\cdot)$ here is the indicator function, and $C > 0$ is a constant). The model thus attempts to enforce both consistency between observations and true colors, and local smoothness of true colors.

We have selected three variables to be the queries, denoted by the red circles in Fig 4.4. One can see in Fig. 4.4c that residual belief propagation, having no way to use the information about the query, concentrates most of the computation near the segment boundaries. Because pixels near the boundaries have the most conflicting nearby evidence, it takes more BP updates for their beliefs to converge than for pixels in the inner regions of segments, where the local evidence is mostly in agreement. Thus RBP successfully concentrates the effort on the hardest parts of the model. However, Fig. 4.4c shows that many regions with high update counts are far away from the query variables and thus have little influence on the query beliefs. In contrast, Figures 4.4d-4.4g show that QSBP applies updates in a much more concentrated manner, focusing on the regions of the model that have affect the query variables. Observe that *within the region of the model that affects the query,* QSBP exhibits similar behavior to RBP, applying more updates to the pixels near segment boundaries, which are the slowest to converge. At the same time, QSBP was able to use updates more efficiently by ignoring hard, but irrelevant, regions.

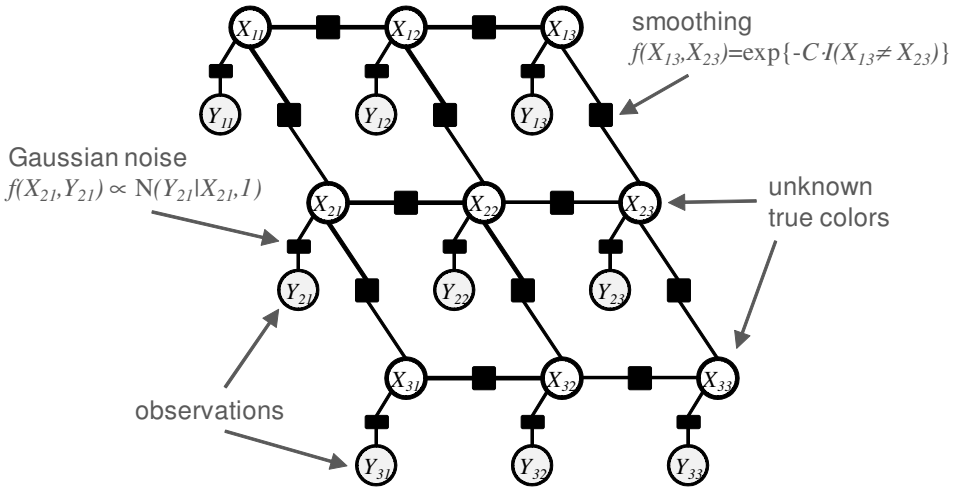


Figure 4.3: A graphical model for image denoising (here for a 3×3 image).
y_{ij} are the observed noisy pixel values. x_{ij} are the unknown "true" pixel values, which are inferred using belief propagation.

4.4 Anytime Query-Specific Residual BP

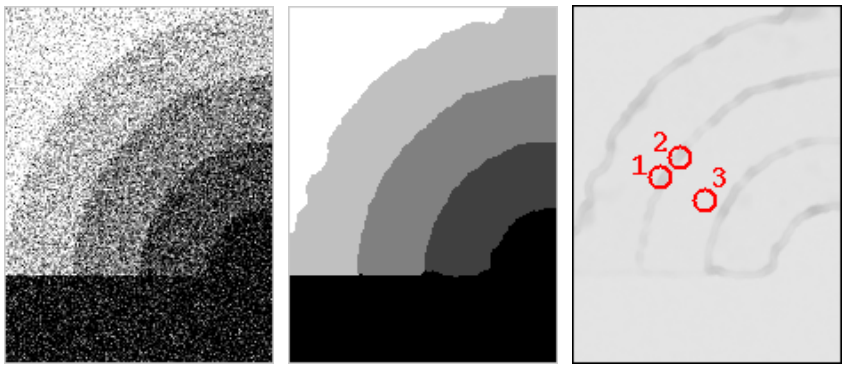
One attractive property of the standard belief propagation algorithms is the fact that they are almost anytime. Both the standard loopy BP and residual BP have initialization complexity of $O(|\mathbb{T}|)$ before applying the first BP update (initializing the messages themselves costs $O(1)$ per message, and a priority queue for RBP using a Fibonacci heap can be initialized in $O(|\mathbb{T}|)$ time). After this lightweight message initialization, one can stop either algorithm at any time and read off the current estimate of the query belief according to (4.5). Although QSBP initialization stage is also quite efficient, it is nevertheless asymptotically more expensive than LBP and RBP: the initialization stage involves calling Alg. 4.1 with complexity $O(|\mathbb{T}|(\log|\mathbb{T}| + d_f d_v))$. In this section, we show how to postpone much of the QSBP initialization until later stages of the inference process, and thus make QSBP anytime, by interleaving edge weighting and inference. Further, in section 4.4.1 we will extend this technique of deferring initialization to also defer a large part of the initialization of the baseline RBP (computing the initial values of new messages and the residuals). As a result, even though query-specific belief propagation involves an extra overhead (compared to RBP) of computing edge importance weights, the resulting algorithm with deferred initialization has *better* anytime properties than the baseline RBP. Empirically (c.f., section 4.7) we have found that it is the latter step of deferring RBP initialization that produces the most benefits in practice, because of the different constants in Dijkstra's algorithm complexity and RBP initialization complexity.

The key insight that allows one to postpone computing importance weights for many edges of the model is that in the query-specific belief propagation (Alg. 4.2) the choice of the message to update on the next step depends only on the *top element* $(\alpha - i)$ in the priority queue $\mathbb{M}$ (c.f. line 8) and does *not* depend on the exact priorities of other edges. The only part that matters is that those other edges all have priorities smaller than $(\alpha - i)$. Therefore, for any upper bound $u(\beta - j)$ on the weighted residual,

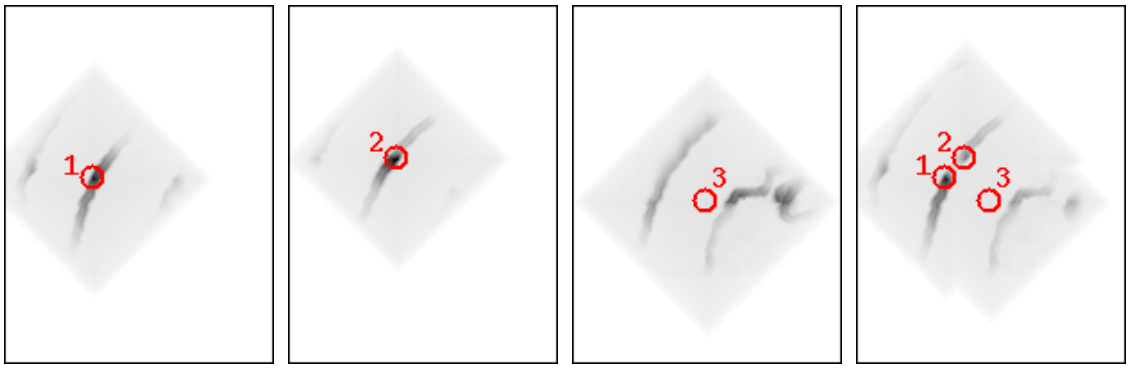
$$u(\beta - j) \geq w_{\beta-j} \cdot r_{\beta-j},$$

as long as the *exact* weighted residual of $(\alpha - i)$ is greater than $u(\cdot)$ for all other edges, that is

$$r_{\alpha-i} \cdot w_{\alpha-i} > u(\beta - j) \;\; \forall(\beta - j) \neq (\alpha - i), \tag{4.16}$$



(a) Noisy observation (b) Denoised image (c) Residual BP updates distribution



(d) QSBP updates, $Q = x_1$ (e) QSBP updates, $Q = x_2$ (f) QSBP updates, $Q = x_3$ (g) QSBP updates, $Q = \{x_1, x_2, x_3\}$

Figure 4.4: The noisy image in (a) is used as observations Y_{ij} for the model in Fig. 4.3. Belief propagation is then used to infer the "true" values of every pixel; an example result of the inference is in (b).
Fig. (c) and bottom row: edge update densities for RBP and QSBP. Darker shade means larger number of updates, lighter - smaller number, white means zero updates. The total number of updates and the correspondence of updates number to the shade of gray are the same for every setting. Query variables are centers of the red circles.

one can guarantee that Alg. 4.2 will update message $\nu_{\alpha-i}$ on the next step without computing the exact weighted residuals for edges $(\beta - j)$. Naturally, the upper bound $u(\cdot)$ has to be computationally cheap (i.e., cheaper than actually computing the exact edge importance weights) to be useful. Fortunately, a key invariant maintained by Alg. 4.1 immediately yields an upper bound at essentially no extra cost:
Invariant 53. Throughout the runtime of Alg. 4.1, for any edge $(\beta - j) \in \mathbb{L}$ it holds that

$$w_{\beta-j} \leq \text{pt}(\mathbb{L}), \tag{4.17}$$

where $\text{pt}(\cdot)$ denotes the priority of the top element in a priority queue.

The proof follows directly from the proof of Prop. 49.

Observe that at any point in the runtime of Alg. 4.1, for every edge $\beta - j \in \mathbb{T}$ either the upper bound 4.17 applies, or the importance value $w_{\beta-j}$ is already known exactly, because all of the edges of the model

Algorithm 4.3: Anytime query-specific belief propagation

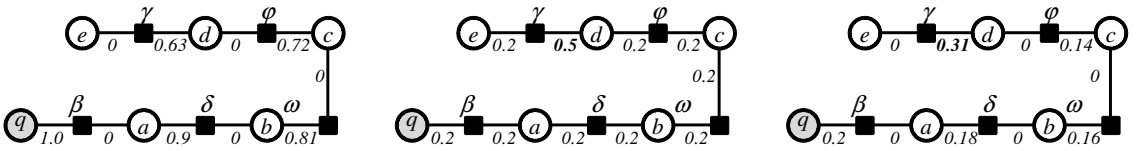
Input: Factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, query set $Q \in X$.

1 $\mathbb{L} = \emptyset$ is priority queue for computing edge importance, $\rho_{\alpha-i}$ is edge priority in $\mathbb{L}$
2 $\mathbb{M} = \emptyset$ is priority queue for prioritizing BP updates
3 $\mathbb{K} = \emptyset$ is priority queue for edges with not yet known importance values
4 **foreach** $(\alpha - i) \in \mathbb{T}$ initialize the message $\nu_{\alpha-i}$
5 **foreach** $(\alpha - i) \in \mathbb{T}$ **do**
6 add $(\alpha - i)$ to $\mathbb{L}$ with priority $\rho_{\alpha-i} = \begin{cases} 1, & x_i \in Q \\ 0, & x_i \notin Q \end{cases}$
7 compute $\widehat{\nu}_{\alpha-i}, r_{\alpha-i}$ using (4.6,4.7), add $(\alpha - i)$ to $\mathbb{K}$ with priority $r_{\alpha-i}$
8 **while** *not converged* **do**
9 **if** $\mathbb{M} \neq \emptyset$ ***AND*** $[\mathbb{L} \neq \emptyset$ ***AND*** $pt(\mathbb{M}) > pt(\mathbb{L}) \cdot pt(\mathbb{K})]$ **then**
   // Condition (4.16) *holds. Edge* $(\alpha - i)$ *is guaranteed to have the largest weighted residual.*
10 denote $(\alpha - i)$ to be the top of $\mathbb{M}$
11 $\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}$
12 set priority of $(\alpha - i)$ in $\mathbb{M}$ to 0
13 **foreach** $\beta \in \Gamma_i \setminus \alpha$ **foreach** $j \in \Gamma_\beta \setminus i$ **do**
14 recompute $\widehat{\nu}_{\beta-j}$ and $r_{\beta-j}$ using Eq. 4.6, 4.7
15 **if** $(\beta - j) \in \mathbb{M}$ **then**
16 set priority of $(\beta - j)$ in $\mathbb{M}$ to $r_{\beta-j} \cdot w_{\beta-j}$
17 **else**
18 set priority of $(\beta - j)$ in $\mathbb{K}$ to $r_{\beta-j}$
19 **else**
   // Condition (4.16) *does not hold. Need to tighten the weighted residual*
   // upper bounds or find an edge with a larger exact weighted residual.
20 denote $(\alpha - i)$ to be the top of $\mathbb{L}$, remove $(\alpha - i)$ from $\mathbb{L}$ and $\mathbb{K}$
21 set $w_{\alpha-i} \leftarrow \rho_{\alpha-i}$, add $(\alpha - i)$ to $\mathbb{M}$ with priority $w_{\alpha-i} \cdot r_{\alpha-i}$
22 **foreach** $j \in \Gamma_\alpha \setminus i$ **foreach** $\beta \in \Gamma_j \setminus \alpha$ **such that** $(\beta - j) \in \mathbb{L}$ **do**
23 $\rho_{\beta-j} \leftarrow \max\left(\rho_{\beta-j}, \rho_{\alpha-i} \cdot \sup_\nu \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\|\right)$
24 update the position of $(\beta - j)$ in $\mathbb{L}$ to match priority $\rho_{\beta-j}$
25 **return** $\tilde{P}(Q)$ using Eq. 4.5

$G = (\{X, \mathcal{F}\}, \mathbb{T})$ are added by Alg. 4.1 to the priority queue $\mathbb{L}$ initially (line 2), and every edge $(\beta - j)$ is only removed from $\mathbb{L}$ when its importance value $w_{\beta-j}$ is known exactly (line 5). Because the message residuals $r_{\beta-j}$ are always nonnegative, one can bound the weighted residual $w_{\beta-j} \cdot r_{\beta-j}$ from above by replacing the exact importance value $w_{\beta-j}$ with its upper bound from Invariant 53:

$$w_{\beta-j} \cdot r_{\beta-j} \leq u(\beta - j) \equiv r_{\beta-j} \cdot \begin{cases} \text{pt}(\mathbb{L}), & (\beta - j) \in \mathbb{L} \\ w_{\beta-j} & \text{otherwise} \end{cases} . \tag{4.18}$$

Incorporating condition (4.16) with weighted residual upper bound 4.18 into Alg. 4.2, we arrive at Alg. 4.3, which we call the *anytime query-specific belief propagation* (**AQSBP**). Instead of computing all the edge importance values in advance, Alg. 4.3 applies BP updates as soon as an edge with the highest *exact known* weighted residual is guaranteed to have the highest *exact overall* weighted residual by (4.16). One can



(a) Edge importance values for $Q = \{q\}$.

(b) Possible initial message residuals for every edge. Edge $(\gamma - d)$ has the largest residual of 0.5.

(c) Initial weighted residuals corresponding to importance weights in (a) and unweighted residuals in (b). Edge $(\gamma - d)$ has the largest weighted residual of 0.31.

Figure 4.5: An example of a setting where anytime QSBP (Alg. 4.3) will compute exact importance weights of all the edges with positive importance before applying the first BP update, resulting in the same initialization time as the standard QSBP.
Fig. (a) shows the possible maximum sensitivity edge importance weights for query $Q = \{q\}$. Both Alg. 4.1 (as a subroutine of QSBP) and AQSBP (on lines 20-24) process edges in the order of decreasing importance weights, so the edge $(\gamma - d)$ will be processed last out of edges with positive importance. At the same time, Fig. (c) shows that $(\gamma - d)$ has the highest weighted residual, and therefore will receive the first BP update by both QSBP and AQSBP. Thus in this case AQSBP and QSBP have the same initialization time.

show that such an interleaving of edge importance computation and BP updates yields identical inference results to QSBP:

Proposition 54. *Assuming the same message initialization and the same outcomes of breaking ties between edges of equal priority, the sequence of message updates performed by Alg. 4.3 is the same as for QSBP.*

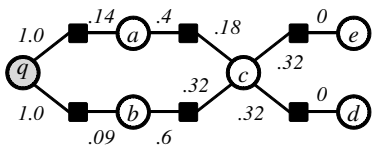
Proof sketch. The proof is by induction. Induction base: after initialization, the message vectors ν^{QSBP} and ν^{AQSBP} are the same. It follows that all the residuals are also the same: $r^{QSBP} = r^{AQSBP}$. Suppose the sequence of first t updates was the same for QSBP and AQSBP. It follows that the messages $\nu^{(t)}$ and residuals $r^{(t)}$ are also the same after t steps. Therefore, $\arg\max w_{\gamma-k} r^{(QSBP)}_{\gamma-k} = \arg\max w_{\gamma-k} r^{(AQSBP)}_{\gamma-k}$. QSBP will then choose the edge $(\alpha - i) = \arg\max w_{\gamma-k} r^{(QSBP)}_{\gamma-k}$ because the priority queue $\mathbb{M}$ is prioritized by the weighted residual. Because $\mathbb{M}$ in Alg. 4.3 is also prioritized by the exact weighted residual, to show that AQSBP will choose the same edge $(\alpha - i)$, probably after computing edge importance values $w_{\beta-j}$ for some edges $(\beta - j)$ on lines 20-21, it is sufficient to show that all the edges $(\gamma - k)$ with the same weighted residual as $(\alpha - i)$, will be in $\mathbb{M}$ whenever line 10 is reached next. Assume that $(\gamma - k)$ is such that

$$w_{\gamma-k} r_{\gamma-k} = w_{\alpha-i} r_{\alpha-i} \text{ and } (\gamma - k) \notin \mathbb{M} \text{ on line 10.}$$

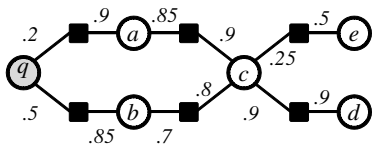
It follows that

$$\text{pt}(\mathbb{L}) \cdot \text{pt}(\mathbb{K}) \geq \text{pt}(\mathbb{L}) \cdot r_{\gamma-k} = u(\gamma - k) \geq w_{\gamma-k} \cdot r_{\gamma-k} \geq \text{pt}(\mathbb{M}),$$

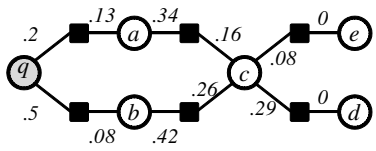
which directly contradicts the fact that condition on line 9 of Alg. 4.3 has to be false for the line 10 to be reached. Thus all edges $(\gamma - k)$ with the highest weighted residual are in $\mathbb{M}$. From the assumption that the ties are broken with the same outcomes as in Alg. 4.2, int follows that $(\alpha - i)$ is chosen by Alg. 4.3 for the



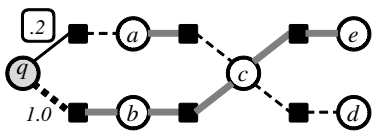
(a) Edge importance values for the model from Fig. 4.2 with query variable x_q.



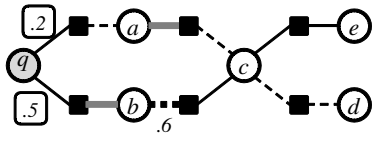
(b) Possible initial message residuals for every edge. The largest unweighted residual is 0.9.



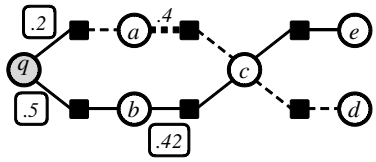
(c) Weighted residuals resulting from edge weights in (a) and message residuals in (b).



(d) After computing edge importance of 1 edge. Remaining edges have weighted residuals of at most $1.0 \times .9 = .9$.



(e) After computing edge importance of 2 edges. Remaining edges have weighted residuals of at most $.6 \times .9 = .54$.



(f) After computing edge importance of 3 edges. Remaining edges have weighted residuals of at most $.4 \times .9 = .36$.

Figure 4.6: An example trace of anytime QSBP (Alg. 4.2) until the first BP update is applied
Top row: edge importance values (a), initial message residuals (b) and the resulting weighted residuals (c). The edge with weighted residual 0.5 touching variable q has the largest weighted residual and should receive the first BP update.
Bottom row: The sequence of edges whose exact importance weight are computed on lines 20-24 of Alg. 4.3 until the first BP update is applied. *Thick dashed line* - the edge on top of the edge importance priority queue $\mathbb{L}$, the *number without a rectangle* is the priority ρ of that edge (i.e., pt($\mathbb{L}$)). *Thin dashed edges* are those with the unweighted residuals equal to pt($\mathbb{K}$). *Solid thick gray edges* are those whose weighted residual is *not* guaranteed to be smaller than pt($\mathbb{M}$).
After expanding 3 edges (d)-(f), all other edges are guaranteed to have weighted residual of at most $\text{pt}(\mathbb{L}) \cdot \text{pt}(\mathbb{K}) = 0.4 \cdot 0.9 = 0.36$, so the edge with weighted residual 0.5 touching the variable q is known to be the first one to receive a BP update on lines 10-18. Notice that AQSBP could not apply a BP update to the edge with weighted residual 0.5 right after expanding it in (e) - only after one more edge (with importance 0.6 and weighted residual 0.42) had been expanded in (f) was the condition on line 9 able to guarantee that no other edge had a weighted residual above 0.5.
Here, AQSBP has significant advantage over QSBP in terms of initialization time.

next update. Given that during the first t steps both QSBP and AQSBP have applied the same sequence of updates, we have shown that on step $t + 1$ they will also update the same message, proving the induction step. □

The guarantee of Prop. 54 entails that after any fixed number t of BP updates, for all the edges of the factor graph G, AQSBP and QSBP will arrive at exactly the same BP messages (and consequently same variable beliefs). Notice that the two inference results are guaranteed to be equal even in the presence of multiple BP fixed points and regardless of the convergence properties of belief propagation on G. The only difference of the anytime version is that typically much of the computation related to importance values of many edges will be delayed until the exact importance values for those edges are actually needed. More precisely, AQSBP starts to apply the first BP updates after only $O(|\mathbb{T}|)$ time (same as LBP and RBP) in

the best case, as opposed to $O(|\mathbb{T}|(\log|\mathbb{T}| + d_f d_v))$ for QSBP:
Proposition 55. *The complexity of AQSBP initialization (lines 4-7 of Alg. 4.3) is* $O(|\mathbb{T}|)$.

Proof sketch. Follows directly from the fact that initializing a Fibonacci heap with $|\mathbb{T}|$ elements costs $O(|\mathbb{T}|)$.□

Therefore, in the best case AQSBP arrives at intermediate results of the inference faster than QSBP. In the worst case, however, AQSBP will have to compute exact importance values for all edges with positive importance before applying even one BP update (see Fig. 4.5 for an example of such model). Therefore, formally one can only guarantee that the anytime version will apply t-th update *no later* than the plain QSBP, but not improve on QSBP by any amount. Nevertheless, the anytime version is useful in practice, because (a) the edges are pulled from $\mathbb{L}$ and added to $\mathbb{M}$ in the order of decreasing importance weight $w_{\alpha-i}$, so at any time $\mathbb{M}$ contains the edges with the highest importance and (b) in typical models the degenerate case of extremely high residuals corresponding to edges with the lowest importance is almost never realized. In particular, our experiments in Section 4.7 demonstrates that on real-life models Alg. 4.3 obtains intermediate results of reasonable quality significantly faster than Alg. 4.2.

Example: Fig. 4.6 shows an example trace of AQSBP. For a factor graph with 12 edges, in the example of Fig. 4.6 AQSBP only needs to compute exact importance weights of 3 edges (Fig. 4.6d-4.6f) before applying the first BP update. Observe that, like Alg. 4.1, AQSBP expands model edges in the order of decreasing importance value without any regard for the current message residuals on those edges. As a result, there are no guarantees on the ordering of expanded edges in terms of weighted residual: in our example, the weighted residuals of the first 3 expanded edges are 0.2 (Fig. 4.6d), 0.5 (Fig. 4.6e) and (Fig. 4.6f). Also, notice that even after AQSBP reaches the edge with the highest overall weighted residual (Fig. 4.6e) and computes its exact importance, it may not be able to apply a BP update right away. Instead, it may be necessary to compute exact importance weights for more edges (Fig. 4.6f) not to discover an edge with a higher weighted residual, but to tighten the upper bound 4.18 on the weighted residuals of the remaining edges. In Fig. 4.6e, the exact weighted residual of the just expanded edge is 0.5, but the $\text{pt}(\mathbb{L}) = 0.6$, corresponding to the importance of the edge touching variable b, and there are edges in the model with residual 0.9, so $\text{pt}(\mathbb{K}) = 0.9$. Thus, in Fig. 4.6e, from information available to AQSBP one can only guarantee that the weighted residual of the remaining edges is $\text{pt}(\mathbb{L}) \cdot \text{pt}(\mathbb{K}) = 0.54 > 0.5$. After processing the edge touching b in Fig. 4.6f, $\text{pt}(\mathbb{L})$ drops to 0.4 (corresponding to the edge touching a, so at that stage AQSBP is able to guarantee that 0.5 is the largest weighted residual over all the edges of the factor graph.

4.4.1 Pessimistic anytime query-specific belief propagation

One can see that while the any-time QSBP (Alg. 4.3) is typically able to start updating BP messages before all the edge importance weights have been computed, initial residuals for all the edge have to be computed before the first message update (c.f. line 7, which is executed for every edge before the message updates start). In our experience, the time required to initialize all the messages and compute all the initial residuals is comparable to the time required to compute edge importance weights, and often even exceeds the edge weighting time. Therefore, it is desirable to also defer until later stages of the inference process computing the initial residuals for those edges. In this section, we show how to postpone computation of initial residuals for low-importance edges while keeping the message update sequence the same as for baseline QSBP.

Observe in (4.18) that in the upper bound $u(\beta - j)$ for the weighted residual $w_{\beta-j} \cdot r_{\beta-j}$, we have used an

upper bound on the edge *importance weight* $w_{\beta-j}$, but an *exact* value of the edge *residual* $r_{\beta-j}$. Moreover, for edges $\beta - j$ with not yet known exact importance weights (i.e. edges still in priority queue $\mathbb{K}$ rather than in $\mathbb{M}$ in Alg. 4.3), the upper bound $u(\beta - j)$ is the only thing that depends on the exact residual $r_{\beta-j}$ (the condition on line 9 is the only place where an element of $\mathbb{K}$ is read). Therefore, if one can come up with an efficient upper bound $\overline{r_{\beta-j}}$ on the edge residual $r_{\beta-j}$, one could use $\overline{r_{\beta-j}}$ instead of the exact residual to obtain an alternative upper bound $u_p(\beta - j)$ on the weighted residual:

$$w_{\beta-j} \cdot r_{\beta-j} \leq u_p(\beta - j) \equiv \begin{cases} \text{pt}(\mathbb{L}) \cdot \overline{r_{\beta-j}}, & (\beta - j) \in \mathbb{L} \\ w_{\beta-j} \cdot r_{\beta-j} & \text{otherwise} \end{cases} .$$

Fortunately, a suitable upper bound on the unweighted residual can be readily derived using the work of Ihler et al. (2005). They have shown (see Theorem 8 and Corollary 9 in that paper) that for any two message vectors $\nu^{(1)}$ and $\nu^{(2)}$, it holds that

$$\left\| \widehat{\nu}_{\beta-j}\left(\nu^{(1)}\right) - \widehat{\nu}_{\beta-j}\left(\nu^{(2)}\right) \right\| \leq \|\psi_\beta\|. \tag{4.19}$$

Therefore, *as soon as* $\nu_{\beta-j}$ *has been updated at least once* using (4.6), it holds that $r_{\beta-j} \leq \|\psi_\beta\|$.

However, we cannot use $\|\psi_\beta\|$ in place of an upper bound $\overline{r_{\beta-j}}$, because we need the upper bound to hold even *before* $\nu_{\beta-j}$ *has been updated.* Because the residual $r_{\beta-j}$ before the first update to $\nu_{\beta-j}$ depends on the initial value of the message $\nu_{\beta-j}$, to obtain an upper bound that holds throughout the runtime of the algorithm, we need to commit to certain way of initializing the messages. A natural approach is to initialize all the messages $\nu_{\beta-j}$ to uniform: $\nu = \vec{0}$. Uniform messages are especially convenient as a starting point, because a uniform message does not need to be represented explicitly when it is used to compute the new values of *other* BP messages, and therefore actual message initialization (memory allocation and so on) can be deferred until the message has to be *updated,* not just read off.

Denote $r^{(0)}_{\beta-j} \equiv \|\widehat{\nu}_{\beta-j} - 0_{\beta-j}\|$ to be the residual for edge $(\beta - j)$ before the message $\nu_{\beta-j}$ receives the first update. Because $0_{\beta-j}$ is not necessarily a possible outgoing message computed using (4.6) - that is, there may not exist a vector of message values $\nu^{(0)}$ such that $0_{\beta-j} = \widehat{\nu}_{\beta-j}(\nu^{(0)})$, we cannot use (4.19) directly and need to apply a triangle inequality:

$$r^{(0)}_{\beta-j} \equiv \|\widehat{\nu}_{\beta-j} - 0_{\beta-j}\| = \|\widehat{\nu}_{\beta-j}\| \leq \|\widehat{\nu}_{\beta-j}(0)\| + \|\widehat{\nu}_{\beta-j}(\nu) - \widehat{\nu}_{\beta-j}(0)\| \leq \|\widehat{\nu}_{\beta-j}(0)\| + \|\psi_\beta\|. \tag{4.20}$$

It remains to bound $\|\widehat{\nu}_{\beta-j}(0)\|$. Observe that for $\nu = \vec{0}$, every variable log-belief $\tilde{\boldsymbol{\Upsilon}}\left(x_i\right)$ is also uniform and the update (4.6) becomes

$$\widehat{\nu}^{(t)}_{\alpha-i}(x_i \mid \vec{0}) = C + \log \sum_{\boldsymbol{X}_{\alpha\setminus i}} \psi_\alpha\left(\boldsymbol{X}_{\alpha\setminus i}, x_i\right),$$

where C is a constant. It follows that

$$\|\widehat{\nu}_{\beta-j}(0)\| \leq \|\psi_\beta\|. \tag{4.21}$$

and finally plugging (4.21) into (4.20), we get

$$r^{(0)}_{\beta-j} \equiv \|\widehat{\nu}_{\beta-j}\| \leq 2\|\psi_\beta\|. \tag{4.22}$$

Combining the upper bound (4.22) that holds before $\nu_{\beta-j}$ is updated, and (4.19) that bounds the residual $r_{\beta-j}$ after $\nu_{\beta-j}$ is updated at least once, we obtain the final upper bound that holds throughout the inference runtime:

$$r_{\beta-j} \leq 2\|\psi_\beta\| \equiv \overline{r_{\beta-j}}. \tag{4.23}$$

Algorithm 4.4: Pessimistic anytime query-specific belief propagation

Input: Factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, query set $Q \in X$.

1 $\mathbb{L} = \emptyset$ is priority queue for computing edge importance, $\rho_{\alpha-i}$ is edge priority in $\mathbb{L}$

2 $\mathbb{M} = \emptyset$ is priority queue for prioritizing BP updates

3 $\overline{\boldsymbol{r}} = 2 \max_{\psi \in \mathcal{F}} \|\psi\|$ is global unweighted residual upper bound

4 **foreach** $(\alpha - i) \in \mathbb{T}$ **do** add $(\alpha - i)$ to $\mathbb{L}$ with priority $\rho_{\alpha-i} = \begin{cases} 1, & x_i \in Q \\ 0, & x_i \notin Q \end{cases}$

5 **while** *not converged* **do**

6 **if** $\mathbb{M} \neq \emptyset$ ***AND*** $pt(\mathbb{M}) > \overline{\boldsymbol{r}} \cdot pt(\mathbb{L})$ **then**

// Condition (4.16) *holds. Edge* $(\alpha - i)$ *is guaranteed to have the largest weighted residual.*

7 denote $(\alpha - i)$ to be the top of $\mathbb{M}$

8 compute $\widehat{\nu}_{\alpha-i}$, set $\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}$

9 set priority of $(\alpha - i)$ in $\mathbb{M}$ to 0

10 **foreach** $\beta \in \Gamma_i \setminus \alpha$ **foreach** $j \in \Gamma_\beta \setminus i$ **such that** $(\beta - j) \in \mathbb{M}$ **do**

11 recompute $\widehat{\nu}_{\beta-j}$ and $r_{\beta-j}$ using Eq. 4.6, 4.7

12 set priority of $(\beta - j)$ in $\mathbb{M}$ to $r_{\beta-j} \cdot w_{\beta-j}$

13 **else**

// Condition (4.16) *does not hold. Need to tighten the weighted residual*
// upper bounds or find an edge with a larger exact weighted residual.

14 denote $(\alpha - i)$ to be the top of $\mathbb{L}$, remove $(\alpha - i)$ from $\mathbb{L}$

15 set $w_{\alpha-i} \leftarrow \rho_{\alpha-i}$, add $(\alpha - i)$ to $\mathbb{M}$ with priority $w_{\alpha-i} \cdot r_{\alpha-i}$

16 **foreach** $j \in \Gamma_\alpha \setminus i$ **foreach** $\beta \in \Gamma_j \setminus \alpha$ **such that** $(\beta - j) \in \mathbb{L}$ **do**

17 $\rho_{\beta-j} \leftarrow \max\left(\rho_{\beta-j}, \rho_{\alpha-i} \cdot \sup_\nu \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\| \right)$

18 update the position of $(\beta - j)$ in $\mathbb{L}$ to match priority $\rho_{\beta-j}$

19 **return** $\tilde{P}(Q)$ using Eq. 4.5

Now we can replace the computing the exact residuals for edges with not yet known importance values in Alg. 4.3 (where those exact residuals are stored in the priority queue $\mathbb{K}$ with the upper bound $\overline{r_{\beta-j}}$ from (4.23) to obtain Alg. 4.4. We will call Alg. 4.4 *pessimistic anytime QSBP*. "Pessimistic" here refers to the use of the upper bound (4.23), which is not as tight as the exact maximal residual pt($\mathbb{K}$) used by Alg. 4.3. Because of the looser upper bound on the residual of remaining edges, PQSBP may have to carry the exact edge weights computation farther than AQSBP before the first message update. However, the upper bound (4.23) has important advantages:

1. (4.23) is very cheap computationally: it only requires looking at every value of the factor once.

2. (4.23) does not depend on the actual message values ν. Therefore, it is sufficient to compute the global upper bound on unweighted residuals only once (c.f. line 3 of Alg. 4.4) instead of updating it every time a message is updated, as Alg. 4.3 does on line 18.

3. The messages $\nu_{\beta-j}$ do not have to be initialized explicitly until their respective first updates (line 8 of Alg. 4.4), making it possible to start updating messages with high importance much faster than Alg. 4.3, which needs to initialize all the messages and compute all the residuals before any message updates can be performed.

In our experience, the above advantages of pessimistic AQSBP outweigh the drawbacks of using a looser upper bound for the residuals and typically make pessimistic AQSBP a better alternative than the baseline AQSBP.

Similar to Alg. 4.3, we can also guarantee that pessimistic AQSBP will perform message updates in the same order as the basic query-specific BP:

Proposition 56. *Assuming the same message initialization and the same outcomes of breaking ties between edges of equal priority, the sequence of message updates performed by Alg. 4.4 is the same as for QSBP.*

Proof: Analogous to the proof of Prop. 54 and follows from the fact that $\overline{r_{\beta-j}} \equiv \|\psi_\beta\|$ is a valid upper bound on the residual.□

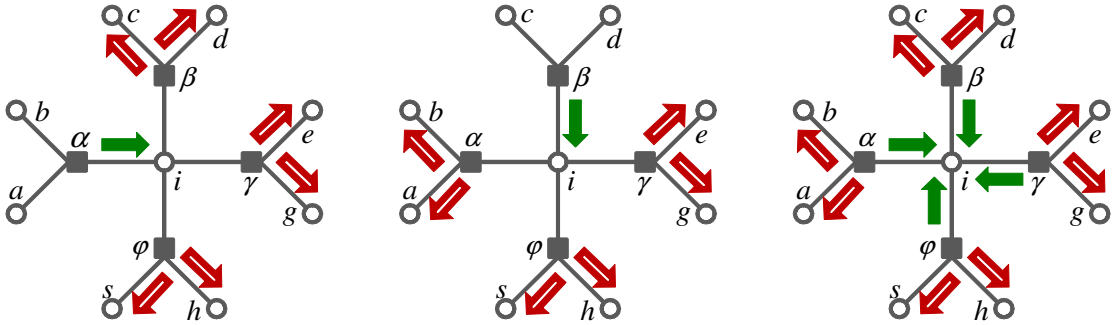
In practice, we have often found it useful to use a smaller estimate of the maximal edge residual $r_{\beta-j}$ than the upper bound $\overline{r_{\beta-j}} \equiv 2\|\psi_\beta\|$ to obtain faster initial updates. A scaled tighter version (not a proper upper bound), such as $\overline{r_{\beta-j}}' \equiv \frac{1}{5}\|\psi_\beta\|$, even though Prop. 56 does not hold for the resulting anytime inference algorithm in general, typically resulted in much faster initial updates without noticeable degradation of the results quality in the longer term.

4.5 Massively reusing computation via variable updates

Before presenting experimental results in the next section, here we address an issue that has little influence on the theoretical analysis of belief propagation, but makes a huge difference in practice and is crucial for obtaining state of the art performance. The issue is one of shared computation: for any two factors ψ_α and ψ_β that share a variable x_i, the set of BP messages that directly depend on message $\nu_{\alpha-i}$ is *almost the same* as the set of messages that directly depend on message $\nu_{\beta-i}$. Moreover, $\nu_{\alpha-i}$ and $\nu_{\beta-i}$ do not directly depend on each other. Therefore, right after updating $\nu_{\alpha-i}$ one should not immediately recompute the messages directly dependent on $\nu_{\alpha-i}$ (as, for example, both standard residual belief propagation and Alg. 4.2 do). Instead, one should right away also update $\nu_{\beta-i}$ and other messages incoming to variable x_i, and only after that recompute the necessary messages. Such a batch message update would have only marginally higher computation cost compared to updating a single message and immediately recomputing the directly dependent variables, and would result in d_i messages becoming up-to-date instead of just one. Therefore, the computation cost *per updated message* of the batch approach is lower by almost a factor of d_i than the cost of the standard single-edge update.

Let us look at the issue of computation sharing more closely. Consider Alg. 4.2 and a fragment of a factor graph in Fig. 4.7a. Suppose the edge $(\alpha - i)$ has the largest weighted residual in the full factor graph, so Alg. 4.2 will update $\nu_{\alpha-i}$ (shown by the green solid arrow in Fig. 4.7a) next. From lines 9-13, one can see that the update consists of two stages.

- The first stage (lines 9-10) only affects the the message $\nu_{\alpha-i}$ itself: the new value $\widehat{\nu}_{\alpha-i}$ is copied over to replace the old value, and the residual $r_{\alpha-i}$ is zeroed out. This stage is relatively cheap computationally.
- The second stage (lines 11-13) affects multiple other messages: every new message $\widehat{\nu}_{\gamma-g}$ that directly depends on $\nu_{\alpha-i}$ via (4.4) has to be recomputed, because $\nu_{\alpha-i}$ has just changed. Denote the



(a) A message $\nu_{\alpha-i}$ (solid green) and messages that directly depend on $\nu_{\alpha-i}$ via equation 4.4 and have to be recomputed whenever $\nu_{\alpha-i}$ is updated (red outline arrows).

(b) A message $\nu_{\beta-i}$ (solid green) and messages that directly depend on $\nu_{\beta-i}$ (red outline arrows). Notice that 4 out of 6 such messages (from factors γ and φ) also depend on $\nu_{\alpha-i}$.

(c) A batch update (Alg. 4.6) of all messages incoming for x_i. First, the incoming messages (solid green) are updated on line 2. Then, the outgoing messages (red outlines) and the corresponding residuals are recomputed on line 4.

Figure 4.7: Edge BP updates (e.g., line 9-12 of Alg. 4.2), applied to different edges connected to the same variable ($\nu_{\alpha-i}$ in Fig. (a) and $\nu_{\beta-i}$ in (b)) have to recompute outgoing messages for sets of edges that have many elements in common, which leads to inefficiencies.
Updating all of the edges connected to the same variable at once and only recomputing the outgoing messages afterwards (Alg. 4.6), illustrated in Fig. (c), eliminates redundant recomputations and significantly improves efficiency: instead of recomputing 6 outgoing message per one updated incoming message (c.f. Fig. (a) and (b)), Alg. 4.6 has to recompute only 2 outgoing messages per one updated incoming message (c.f. Fig. (c)).

set of edges with messages directly affected by $\nu_{\alpha-i}$ to be $\Gamma_{\alpha-i}$:

$$\Gamma_{\alpha-i} \equiv \bigcup_{\beta \in \Gamma_i \setminus \alpha} \bigcup_{j \in \Gamma_\beta \setminus i} (\beta - j).$$

In the example of Fig. 4.7a the affected messages, denoted with red arrows, are on edges $\Gamma_{\alpha-i} = \{(\beta-c), (\beta-d), (\gamma-e), (\gamma-g), (\varphi-h), (\varphi-s)\}$. In general, the number of affected messages depends on the connectivity pattern of the factor graph. If every variable has degree d_v and every factor involves d_f variables, then $(d_v-1)(d_f-1)$ messages need to be recomputed.

It is the second stage of an edge update that is responsible for the predominant share of computation required by both query-specific BP (in the form of Alg. 4.2) and residual BP. Therefore, substantially speeding up the second stage would result in almost equivalent overall speedup of the inference process.

Key to speeding up the message updates is the structure of dependencies between messages related to the same variable. Consider the messages that need to be recomputed after updating $\nu_{\alpha-i}$ (Fig. 4.7a) and after updating $\nu_{\beta-i}$ (Fig. 4.7b). We make two observations:

1. The messages $\nu_{\alpha-i}$ and $\nu_{\beta-i}$ do not directly depend on each other (more precisely, $\widehat{\nu}_{\alpha-i}$ does not

Algorithm 4.5: SequentialUpdateAllIncomingEdges

Input: Factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, old messages ν, new messages $\widehat{\nu}$, variable $x_i \in X$

foreach $\alpha \in \Gamma_i$ **do**
  $\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}$
  foreach $\beta \in \Gamma_i \setminus \alpha$ **foreach** $j \in \Gamma_\beta \setminus i$ **do**
    recompute $\widehat{\nu}_{\beta-j} \leftarrow \widehat{\nu}_{\beta-j}(\nu)$ using (4.4)

Algorithm 4.6: BatchUpdateAllIncomingEdges

Input: Factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, old messages ν, new messages $\widehat{\nu}$, variable $x_i \in X$

foreach $\alpha \in \Gamma_i$ **do**
  $\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}$
foreach $\beta \in \Gamma_i$ **foreach** $j \in \Gamma_\beta \setminus i$ **do**
  recompute $\widehat{\nu}_{\beta-j} \leftarrow \widehat{\nu}_{\beta-j}(\nu)$ using (4.4)

depend on $\nu_{\beta-i}$ and $\widehat{\nu}_{\beta-i}$ does not depend on $\nu_{\alpha-i}$). Therefore, a sequence of two updates,

$$(\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}) \;\Rightarrow\; (\text{recompute } \widehat{\nu} \text{ for } \Gamma_{\alpha-i}) \;\Rightarrow\; \Rightarrow\; (\nu_{\beta-i} \leftarrow \widehat{\nu}_{\beta-i}) \;\Rightarrow\; (\text{recompute } \widehat{\nu} \text{ for } \Gamma_{\beta-i}), \quad (4.24)$$

will have the same result as

$$(\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}) \;\Rightarrow\; (\nu_{\beta-i} \leftarrow \widehat{\nu}_{\beta-i}) \;\Rightarrow\; (\text{recompute } \widehat{\nu} \text{ for } \Gamma_{\alpha-i} \cup \Gamma_{\beta-i}). \quad (4.25)$$

2. The respective sets of messages $\Gamma_{\alpha-i}$ and $\Gamma_{\beta-i}$ that need to be recomputed after updating $\nu_{\alpha-i}$ and $\nu_{\beta-i}$ intersect to a large degree. In our example in Fig. 4.7, $\nu_{\alpha-i}$ and $\nu_{\beta-i}$ each directly affect 6 other messages, and 4 of those directly affected messages are the same, corresponding to edges $\Gamma_{\alpha-i} \cap \Gamma_{\beta-i} = \{(\gamma - e), (\gamma - g), (\varphi - h), (\varphi - s)\}$. More generally, if every variable has degree d_v and every factor involves d_f variables, then for any two messages $\nu_{\alpha-i}$ and $\nu_{\beta-i}$ incoming to the same variable, it holds that

$$\Gamma_{\alpha-i} = \Gamma_{\beta-i} = (d_v - 1)(d_f - 1), \quad \Gamma_{\alpha-i} \cap \Gamma_{\beta-i} = (d_v - 2)(d_f - 1) \quad (4.26)$$

and

$$\Gamma_{\alpha-i} \cup \Gamma_{\beta-i} = d_v(d_f - 1). \quad (4.27)$$

Denote $\boldsymbol{c}$ to be the cost of computing one new message $\widehat{\nu}_{\gamma-j} \in \Gamma_{\alpha-i} \cup \Gamma_{\beta-i}$. Then it follows from (4.26) and (4.27) that the complexity of update sequence (4.24) is $2 \cdot \boldsymbol{c} \cdot (d_v - 1)(d_f - 1)$, while the complexity of update sequence (4.25) is $\boldsymbol{c} \cdot d_v(d_f - 1)$. In other words, update sequence (4.25) is more computationally efficient than (4.24) by a factor of $2(1 - \frac{1}{d_v})$. Moreover, once any two messages $\nu_{\alpha-i}$ and $\nu_{\beta-i}$ incoming for the same variable x_i have been updated, extending the idea of batch update (4.25) to all of the other messages $\nu_{\gamma-i}$ incoming for the same variable x_i leads to *no increase at all in the total complexity* of the cumulative update. Consider the procedure *BatchUpdateAllIncomingEdges* (Alg. 4.6), which updates all the messages incoming for a given variable before recomputing the necessary new messages, and *SequentialUpdateAllIncomingEdges* (Alg. 4.5), which recomputes the necessary new messages after each edge update. One can show that the two procedures have the same effect, but drastically different computation costs:

Proposition 57. *Suppose that every variable in a factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ has degree d_v and every factor involves d_f variables. Let $\mathbf{c}$ be the cost of recomputing a single new message $\widehat{\nu}_{\beta-j}$ using* (4.4)*. Then for any variable x_i it holds that*

1. SequentialUpdateAllIncomingEdges*$(\{X, \mathcal{F}\}, \mathbb{T}), \nu, \widehat{\nu}, x_i)$ and* BatchUpdateAllIncomingEdges*$(\{X, \mathcal{F}\}, \mathbb{T}), \nu, \widehat{\nu}, x_i)$ have the **same result**, with $\nu_{\alpha-i} = \widehat{\nu}_{\alpha-i}$ for all $\alpha \in \Gamma_i$ and $\widehat{\nu}_{\beta-j}$ becoming up to date per the equation 4.4 for all $\beta \in \Gamma_i, j \in \Gamma_\beta \setminus i$.*

2. *Disregarding the costs of copying the new message values $\widehat{\nu}_{\alpha-i}$ to the old messages $\nu_{\alpha-i}$, the **time complexity** of* SequentialUpdateAllIncomingEdges *is $\mathbf{c}d_v(d_v-1)(d_f-1)$, while time complexity of* BatchUpdateAllIncomingEdges *is $\mathbf{c}d_v(d_f-1)$. Therefore,* BatchUpdateAllIncomingEdges *(Alg. 4.6) represents a factor of (d_v-1) speedup over* SequentialUpdateAllIncomingEdges *(Alg. 4.5).*

Proof.
Same result. None of the new messages $\widehat{\nu}_{\alpha-i}$ incoming for the variable x_i is updated by either Alg. 4.5 or Alg. 4.6. Both algorithms set $\nu_{\alpha-i}$ to $\widehat{\nu}_{\alpha-i}$ for every message incoming for x_i. Thus both old and new messages incoming for x_i are the same for both algorithms and equal to the values that $\widehat{\nu}_{\alpha-i}$ had before the updates. Denote $\boldsymbol{\nu}'$ to be the resulting values of the old messages.

As shown in (4.4), the values of the new messages $\widehat{\nu}$ only depend directly on the values of the old messages ν. Therefore, the values of any new message $\widehat{\nu}_{\beta-j}$ only depend on the values of the old messages ν at the moment when $\widehat{\nu}_{\beta-j}$ was recomputed most recently. Alg. 4.6 only recomputes every $\widehat{\nu}_{\beta-j}$ once, with $\nu = \boldsymbol{\nu}'$. The fact that $\nu = \boldsymbol{\nu}'$ also holds for the result of after calling Alg. 4.5 follows from the fact that every time Alg. 4.5 modifies an old message $\nu_{\alpha-i}$, it immediately recomputes every $\widehat{\nu}_{\beta-j}$ that depends on $\nu_{\alpha-i}$ per the equation 4.4. Therefore, none of the messages $\widehat{\nu}_{\beta-j}$ in the result of Alg. 4.5 relies on "obsolete" values of any message $\nu_{\alpha-i}$ incoming for x_i. Because eventually Alg. 4.5 sets $\nu = \boldsymbol{\nu}'$, it follows that $\widehat{\nu} = \widehat{\nu}(\boldsymbol{\nu}')$.

Time complexity. For Alg. 4.5, line 3 is reached $\Gamma_i = d_v$ times. Each time, $|\Gamma_i \setminus \alpha| = d_v - 1$ factors are processed, and for every factor $|\Gamma_\beta \setminus i| = d_f - 1$ outgoing messages are recomputed on line 4. The total complexity is thus $\mathbf{c}d_v(d_v-1)(d_f-1)$.

For Alg. 4.6, line 3 is reached once and the loop is over $|\Gamma_i| = d_v$ factors and $|\Gamma_\beta \setminus i| = d_f - 1$ outgoing messages for every factor, so the total complexity is $\mathbf{c}d_v(d_f-1)$.□

It is not uncommon, especially in large-scale relational models, for a variable to be involved in tens of factors, so the $(d_v - 1)$ is very significant. It is therefore natural to replace the edge updates on lines 9-13 of QSBP (Alg. 4.2), as well as the corresponding parts of Alg. 4.3 and Alg. 4.4 with *BatchUpdateAllIncomingEdges*. In other words, we transition from *edge updates* of Alg. 4.2 to *variable updates* affecting all the edges directly connected to a variable in question. Observe that strictly speaking, such a replacement breaks the behavior of always updating the edge with a highest weighted residual: whenever an edge $(\alpha - i)$ is chosen for an update, edges $(\beta - i)$ for all $\beta \in \Gamma_i$ will be updated right away, even though their respective residuals may be low. However, because those extra updates cost essentially nothing in terms of computation, there is no downside to including them all in the batch update. In the worst case, when all the incoming edges other than $(\alpha - i)$ have zero residual, the batch update would have the same result and only slightly higher cost as the original single-edge update. In the typical case, however, many of the edges $(\beta - i)$ would have substantial residuals and would need to be updated later on in any case, even if not immediately; batch update would make those updates almost for free, leading to total speedups on the order of d_v.

4.5.1 Prioritization by cumulative variable residual

In the previous section, we have discussed how switching from the standard edge updates to variable updates (i.e., updating all of the incoming messages for a variable, and only then recompute the new outgoing messages) can lead to inference speedups proportional to the connectivity density of the factor graph. However, we have not made any changes to the scheduling scheme, which means that updates would still be scheduled in the order of single-edge residuals. Even though scheduling updates by the largest edge residual is theoretically justified by the discussion of section 4.3.1 (adapted from Elidan et al., 2006), we argue that such a schedule is not ideal for the following reasons:

1. The basic schedule element in RBP, Alg. 4.2 and related algorithms is an edge. The priority queue $\mathbb{M}$ in Alg. 4.2 contains $|\mathbb{T}|$ elements. However, with the batch updates of Alg. 4.6, only $|X|$ distinct updates are possible, because the set of updated messages is determined by the instead of the edge. It is desirable to eliminate the redundancy in the elements of $\mathbb{M}$, thereby reducing the complexity of a single update to $\mathbb{M}$ from $O(\log |\mathbb{T}|)$ to $O(\log |X|)$.

2. The choice of a *set* of messages to update next (all messages incoming for a certain variable x_i) is made by *looking only at one edge from that set*, the one with the largest residual, and *ignoring* the residuals of *the remaining* $d_i - 1$ *edges,* where d_i is the degree of x_i in the factor graph.

3. Residual BP schedule (and query-specific BP schedule of Alg. 4.2 as a straightforward extension) does not take update complexity into account. Elidan et al. (2006) arrive at the RBP schedule as a greedy approach to minimizing the largest residual in *as few updates as possible.* In practice, one needs to minimize the largest residual in the *lowest amount of computation* possible.

To address the above issues, we adopt a different scheduling heuristic, formed by the following choices, each addressing the corresponding drawback above.

1. Use variables as prioritized elements of $\mathbb{M}$, which reduces the size of $\mathbb{M}$ to $|X|$.

2. As priority for every variable x_i, use *total* weighted *residual of the incoming edges* for x_i :

$$\text{total-residual}(x_i) = \sum_{\alpha \in \Gamma_i} w_{\alpha - i} r_{\alpha - i}. \tag{4.28}$$

3. To account for the update complexity, total weighted residual (4.28) for every variable x_i is further normalized by the complexity of the update (Alg. 4.6) for x_i :

$$\text{update-complexity-normalizer}(x_i) = |\Gamma_i| \cdot |\text{dom}(x_i)| + \sum_{\alpha \in \Gamma_i} (\Gamma_\alpha - 1) \cdot |\text{dom}(\psi_\alpha)|, \tag{4.29}$$

where $|\text{dom}(x_i)|$ is domain size (i.e., cardinality) of x_i and $|\text{dom}(\psi_\alpha)| \equiv \prod i \in \Gamma_\alpha |\text{dom}(x_i)|$ is the number of entries in the tabular representation of ψ_α. The first component of (4.29) reflects the complexity of copying the new messages $\widehat{\nu}_{\alpha - i}$ to $\nu_{\alpha - i}$, the second component - of recomputing the outgoing messages $\widehat{\nu}_{\alpha - j}$ for all $\alpha \in \Gamma_i$ and $j \in \Gamma_\alpha \setminus i$.

To summarize, our update prioritization heuristic assigns to every variable x_i the following priority:

$$\text{priority}(x_i) \equiv \frac{\text{total-residual}(x_i)}{\text{update-complexity-normalizer}(x_i)} = \frac{\sum_{\alpha \in \Gamma_i} w_{\alpha - i} r_{\alpha - i}}{|\Gamma_i| \cdot |\text{dom}(x_i)| + \sum_{\alpha \in \Gamma_i} (\Gamma_\alpha - 1) \cdot |\text{dom}(\psi_\alpha)|}, \tag{4.30}$$

which has a straightforward interpretation:

Algorithm 4.7: QSBP-V: Query-specific belief propagation with variable updates

Input: Factor graph $G = (\{X, \mathcal{F}\}, \mathbb{T})$, query $Q \in X$.

1 $\mathbb{M}$ is a priority queue
2 $W \leftarrow$ Alg.4.1(G, Q) (find edge importance values)
3 **foreach** $x_i \in X$ set $\mathcal{N}_i \leftarrow \left(|\Gamma_i| \cdot |\mathrm{dom}(x_i)| + \sum_{\alpha \in \Gamma_i} (\Gamma_\alpha - 1) \cdot |\mathrm{dom}(\psi_\alpha)|\right)$ *// From* (4.29)
4 **foreach** $(\alpha - i) \in \mathbb{T}$ initialize the message $\nu_{\alpha - i}$
5 **foreach** $(\alpha - i) \in \mathbb{T}$ compute $\widehat{\nu}_{\alpha-i}, r_{\alpha-i}$ using (4.6,4.7)
6 **foreach** $x_i \in X$ add x_i to $\mathbb{M}$ with priority $\mu_i = \mathcal{N}_i^{-1} \sum_{\alpha \in \Gamma_i} w_{\alpha-i} \cdot r_{\alpha-i}$
7 **while** *not converged* **do**
8 denote x_i to be the top of $\mathbb{M}$, set priority of x_i in $\mathbb{M}$ to $\mu_i \leftarrow 0$
9 **foreach** $\alpha \in \Gamma_i$ do $\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}$
10 **foreach** $\alpha \in \Gamma_i$ *and* $j \in \Gamma_\alpha \setminus i$ **do**
// Account for changes in μ_j due to changing $r_{\alpha-j}$ to avoid recomputing μ_j from scratch
11 $\mu_j \leftarrow \mu_j - \mathcal{N}_j^{-1} w_{\alpha-j} r_{\alpha-j}$ *// Subtract the contribution of the old residual*
12 recompute $\widehat{\nu}_{\alpha-j}$ and $r_{\alpha-j}$ using Eq. 4.6, 4.7
13 $\mu_j \leftarrow \mu_j + \mathcal{N}_j^{-1} w_{\alpha-j} r_{\alpha-j}$ *// Add the contribution of the new residual*
14 set priority of x_j in $\mathbb{M}$ to μ_j
15 **return** $\tilde{P}(x_q)$ for $x_q \in Q$ using Eq. 4.5

Observation 58. Ordering the variable updates of Alg. 4.6 in the order of decreasing priority (4.30) corresponds to the greedy minimization of the total weighted residual $\sum_{(\alpha-i)\in\mathbb{T}} w_{\alpha-i} r_{\alpha-i}$ with respect to computation time under the assumption that all of the updates in Alg. 4.6 of the outgoing messages $\nu_{\beta-j}$ on line 4 leave the residuals $r_{\beta-j}$ unchanged.

Combining the above prioritization with variable updates of Alg. 4.6, we arrive at Alg. 4.7, which we will call query-specific belief propagation with variable updates (QSBP-V).

4.5.2 Anytime inference and variable weighting

Although the modifications required to obtain QSBP-V (Alg. 4.7) from the original QSBP (Alg. 4.2) are quite simple, constructing anytime versions of QSBP-V in the spirit of Alg. 4.3 and Alg. 4.4 is not as straightforward. While plugging in the variable updates of Alg. 4.6 in the anytime algorithms is simple, modifying the scheduling scheme to use variables with priorities (4.30) is more complicated. The difficulty arises from the fact that the priority (4.30) aggregates the weighted residual of multiple edges. Because the upper bounds on the weighted residuals discussed in sections 4.4 and 4.4.1 are only valid for individual edges, we need a way to aggregate the upper bounds to obtain an upper bound on the priority of the corresponding variable. In this section, we first show that an obvious way to perform such an aggregation lacks the necessary properties for the anytime inference, and propose an efficient alternative approximation with quality guarantees.

In sections 4.4 and 4.4.1, we have discussed two possible upper bounds on the weighted residuals of edges whose importance weights are not yet known:

$$u(\alpha - i) = r_{\alpha-i} \cdot \mathrm{pt}(\mathbb{L}) \;\text{ and }\; u(\alpha - i) = 2 \max_{\psi \in \mathcal{F}} \|\psi\| \cdot \mathrm{pt}(\mathbb{L}), \tag{4.31}$$

where $\mathbb{L}$ is the priority queue containing candidate edge importance weights. A straightforward way to incorporate the upper bounds (4.31) into the variable priority (4.30) is to simply replace the exact weighted residuals $w_{\beta-j}r_{\beta-j}$ with their respective upper bounds from (4.31) for edges whose exact importance weights are not yet known:

$$\text{priority-ub}(x_i) = \frac{\sum_{\alpha\in\Gamma_i, w_{\alpha-i} \text{ is known}} w_{\alpha-i}r_{\alpha-i} + \sum_{\alpha\in\Gamma_i, w_{\alpha-i} \text{ is \textbf{not} known}} u(\alpha-i)}{|\Gamma_i|\cdot|\text{dom}(x_i)| + \sum_{\alpha\in\Gamma_i}(\Gamma_\alpha - 1)\cdot|\text{dom}(\psi_\alpha)|} \geq \text{priority}(x_i). \tag{4.32}$$

Unfortunately, the upper bound (4.32) has two serious drawbacks that render it unsuitable for the anytime version of Alg. 4.7:

1. To preserve efficiency, both Alg. 4.3 and Alg. 4.4 exploit the following observation. First, the condition (4.16), which guarantees that the edge $(\alpha - i)$ will be the next one updated by QSBP, can be decomposed into two complimentary conditions:

   $$w_{\alpha-i}\cdot r_{\alpha-i} \geq u(\beta-j)\ \forall(\beta-j)\in\mathbb{T} \Leftrightarrow$$
   $$w_{\alpha-i}\cdot r_{\alpha-i} \geq \max_{(\beta-j)\in\mathbb{T}, w_{\beta-j} \text{ is known}} w_{\beta-j}\cdot r_{\beta-j} \textbf{ AND } w_{\alpha-i}\cdot r_{\alpha-i} \geq \max_{(\beta-j)\in\mathbb{T}, w_{\beta-j} \text{ is \textbf{not} known}} u(\beta-j).$$

   Both of the conditions above can be checked efficiently: the first one is enforced using the priority queue $\mathbb{M}$, while for the second one we have

   $$\max_{(\beta-j)\in\mathbb{T}, w_{\beta-j} \text{ is \textbf{not} known}} u(\beta-j) = \text{pt}(\mathbb{L})\cdot \max_{(\beta-j)\in\mathbb{T}, w_{\beta-j} \text{ is \textbf{not} known}} \text{raw-residual-ub}(\beta-j), \tag{4.33}$$

   where raw-residual-ub$(\beta - j)$, equal to either pt$(\mathbb{K})$ in Alg. 4.3 or $2\max_{\psi\in\mathcal{F}}\|\psi\|$ in Alg. 4.4, is the upper bound on *unweighted* edge residuals. The product decomposition is crucial to efficient recomputation of the right-hand side of (4.33) after an edge $(\gamma - k)$ is pulled off the top of priority queue $\mathbb{L}$ by Dijkstra's algorithm: it is sufficient to either remove $(\gamma - k)$ from the raw residuals priority queue $\mathbb{K}$ or even do nothing at all using the $2\max_{\psi\in\mathcal{F}}\|\psi\|$ upper bound.

   In contrast to the upper bounds for single-edge weighted residual, the upper bound (4.32) on the total weighted residual for a variable *does not decompose* into a product of an upper bound on edge weights and an upper bound on raw residuals, because of the term $\sum_{\alpha\in\Gamma_i, w_{\alpha-i} \text{ is known}} w_{\alpha-i}r_{\alpha-i}$ that does not depend on those single-edge upper bounds. As a result, every time Dijkstra's algorithm removes an edge $(\gamma - k)$ off the top of the priority queue $\mathbb{L}$, the upper bound (4.32) has to be recomputed for every variable x_i that has at least one incoming edge $(\gamma - k)$ such that $w_{\gamma-k}$ is not yet known exactly. Therefore, the overhead of updating (4.32) for every change of $\mathbb{L}$ is $O(|X|)$, which is significantly more expensive than both $O(\log|\mathbb{T}|)$ for Alg. 4.3 and $O(1)$ for Alg. 4.4.

2. The exact priority value (4.30) for variable x_i can only be calculated after the importance weights for *all* edges incoming for x_i have been computed. In other words, it would be necessary to wait until Dijkstra's algorithm removes every edge $(\alpha - i), \alpha \in \Gamma_i$ from the priority queue $\mathbb{L}$ before updating the messages for x_i. Because Dijkstra's algorithm processes edges in the order of decreasing importance weight, it follows that the earliest time at which the messages incoming for x_i is determined by the least important incoming edge - even when all the remaining edges have high importance, a single low-importance edge would hold up the message updates, leading to more edges being processed upfront by Dijkstra's algorithm and compromising the anytime behavior.

   Example: consider the setting in Fig. 4.8. Assume that x_q is the only query variable, every factor is such that the upper bound (4.12) on the partial message derivatives is 0.6, and every edge residual

is 1 except for $r_{\alpha_1 - q} = r_{\alpha_2 - q} = 0$. Then the edges have maximum sensitivity importance values shown in Fig. 4.8a and Alg. 4.1 will expand the edges on line 5 in the order shown in Fig. 4.8c. Assume also for simplicity that the edge residuals are known exactly (i.e., the accuracy of an upper bound on the edge residuals is not an issue). We are interested in the minimal number of edges expanded by Alg. 4.1 that would allow one to unambiguously infer the variable which should be updated next.

From Fig. 4.8b, one can see that the variables with the highest total weighted residual (4.28) are x_{i_1} and x_{i_2}. Consider the evolution of the bounds on the total weighted residuals of x_{i_1} and x_{i_2} as Alg. 4.1 expands the edges of the model on line 5. It holds that

$$\begin{aligned} w_{\alpha_3 - i_1} \cdot r_{\alpha_3 - i_1} &\leq \text{total-residual}(x_{i_1}) = w_{\alpha_3 - i_1} \cdot r_{\alpha_3 - i_1} + w_{\alpha_1 - i_1} \cdot r_{\alpha_1 - i_1} \\ &\leq w_{\alpha_3 - i_1} \cdot r_{\alpha_3 - i_1} + \text{pt}(\mathbb{L}) \cdot r_{\alpha_1 - i_1}, \end{aligned}$$

and analogously for x_{i_2}. After Alg. 4.1 expands 4 edges, namely $(\alpha_1 - x_q)$, $(\alpha_2 - x_q)$, $(\alpha_3 - x_{i_1})$ and $(\alpha_4 - x_{i_2})$, plugging in the concrete weights and residuals, we get $\text{total-residual}(x_q) = 0$ and

$$\begin{gathered} \text{total-residual}(x_{i_1}) \in [0.6, 0.6 + \text{pt}(\mathbb{L})],\ \text{total-residual}(x_{i_2}) \in [0.6, 0.6 + \text{pt}(\mathbb{L})], \\ \text{total-residual}(x_{i_3}) \in [0.36, 0.36 + \text{pt}(\mathbb{L})],\ \text{total-residual}(x_{i_4}) \in [0.36, 0.36 + \text{pt}(\mathbb{L})], \end{gathered}$$

and for all the remaining variables x_i, because there are 2 incoming edges for every variable, we have $\text{total-residual}(x_i) \in [0, 2 \cdot \text{pt}(\mathbb{L})]$. After 2 more edges, namely $(\alpha_5 - x_{i_3})$ and $(\alpha_6 - x_{i_4})$, have been expanded by Alg. 4.1, it holds that $\text{pt}(\mathbb{L}) = 0.22$ (edge $(\alpha_7 - i_5)$ is the next to be expanded), so

$$\begin{gathered} \text{total-residual}(x_{i_1}) \in [0.6, 0.82],\ \text{total-residual}(x_{i_2}) \in [0.6, 0.82], \\ \text{total-residual}(x_{i_3}) \in [0.36, 0.58],\ \text{total-residual}(x_{i_4}) \in [0.36, 0.58], \end{gathered}$$

and

$$\text{total-residual}(x_i) \in [0, 0.44] \text{ for all } i \in \{i_5, i_6, i_7\}.$$

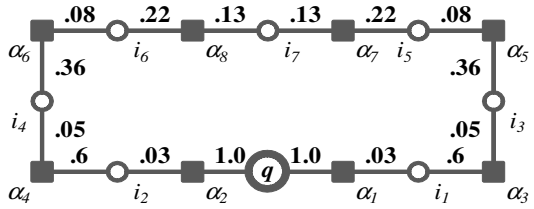
Therefore, after examining 6 edges of the model, one can already guarantee that either x_{i_1} or x_{i_2} has the highest total weighted residual (4.28). However, to find out which of x_{i_1} or x_{i_2} has higher total weighted residual (in our case, make sure that they are the same), one needs to find the exact edge importance weights for $(\alpha_1 - i_1)$ and $(\alpha_2 - i_2)$. Because $(\alpha_1 - i_1)$ and $(\alpha_2 - i_2)$ have the lowest importance weights out of all the model edges, they will be expanded last. Indeed, even after 14 out of 16 edges have been expanded, we have $\text{pt}(\mathbb{L}) = 0.03$ and

$$\text{total-residual}(x_{i_1}) \in [0.6, 0.63],\ \text{total-residual}(x_{i_2}) \in [0.6, 0.63],$$

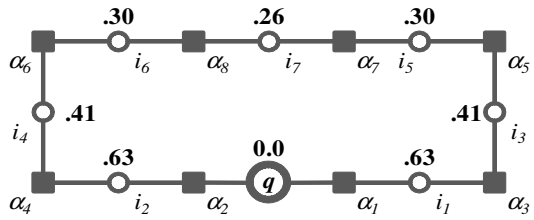
which means that there is no new information about which of x_{i_1} and x_{i_2} has the largest total weighted residual since edge number 4 (i.e., $(\alpha_4 - i_2)$) has been examined. Only after expanding $(\alpha_2 - i_2)$, edge 15 out of 16, we obtain

$$\begin{aligned} &\text{total-residual}(x_{i_1}) \in [0.6, 0.63],\ \text{total-residual}(x_{i_2}) = 0.63 \ \Rightarrow \\ &\qquad \Rightarrow\ \text{total-residual}(x_{i_2}) \geq \text{total-residual}(x_{i_1}). \end{aligned}$$

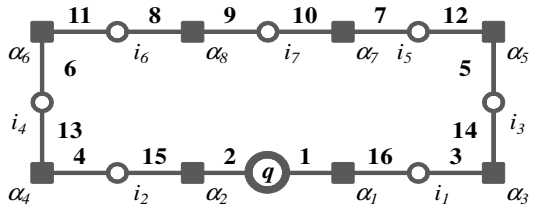
One can see that even if all the edge residuals are known exactly, and in the absence of pathological cases of low-importance edges having very high residuals, it may be necessary to process almost *all*



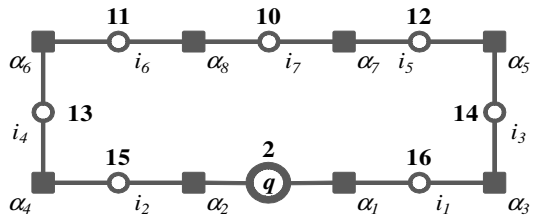
(a) Maximum sensitivity edge importance weights for the query $Q = \{x_q\}$.



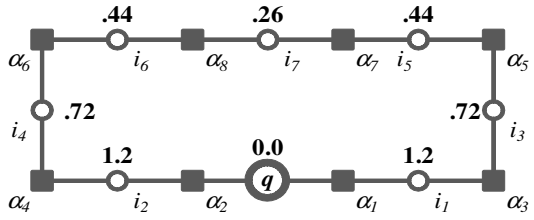
(b) Total variable residual values weighted by edge importance (c.f. equation 4.28) for the case where every edge residual $r_{\alpha-i} = 1$ except for the edges connected to the query variable x_q, where $r_{\alpha_1-q} = r_{\alpha_2-q} = 0$.



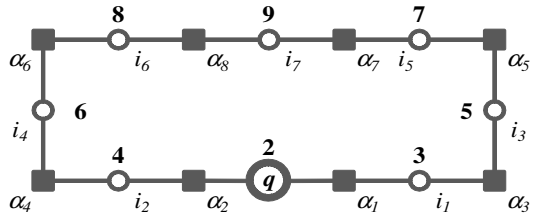
(c) The order in which Alg. 4.1 expands the edges of the model while computing their maximum sensitivity importance weight.



(d) The number of edges that are necessary for Alg. 4.1 to expand to compute for the respective variables the exact weighted total variable residuals (4.28) shown in Fig. (b).



(e) Total variable residuals weighted by the variable maximum sensitivity importance (c.f. equation 4.35) for the same edge residuals as in (b).



(f) The number of edges that are necessary for Alg. 4.1 to expand to compute for the respective variables the exact weighted total variable residuals (4.35) shown in Fig. (e).

Figure 4.8: An example setting (Fig. (a)-(d)) where the need for Alg. 4.1 to expand low-importance edges in order to compute the weighted total variable residuals (4.28) renders the anytime approach useless *regardless of the tightness of the upper bound on the unweighted residuals.*
Fig. (e) and (f) show that weighting residuals by variable importance per the equation (4.35) restores the anytime behavior.

of the edges of the model to compute the total weighted residual (4.28) of the *variable with the highest residual.* Such a property would lead to longer initialization stages of an anytime modification and larger delay before the first edge updates are applied, compared to e.g. Alg. 4.3 and Alg. 4.4

Notice that there are two completely distinct issues, each of which may require expanding many

low-priority edges of the model with Dijkstra's algorithm:

(a) The need to know the exact importance of *every* incoming edge for variable x_i before updating x_i. This problem would only affect an anytime modification of QSBP-V with (4.32) as an upper bound on the variable priority. Algorithms 4.3 and 4.4 are not affected by this issue.

(b) The upper bounds on the raw residuals, $\text{pt}(\mathbb{K})$ for Alg. 4.3 and $2\max_{\psi\in\mathcal{F}}\|f\|$ for Alg. 4.4 overestimate the actual residuals in the model (or, in case of Alg. 4.3, the edges with highest raw residuals actually have low importance). This issue affects both the anytime modifications of QSBP and would affect any anytime modification of QSBP-V to the same degree.

Fortunately, both of the problems discussed above can be addressed by only slightly changing the definition of priority for a given variable. Define the maximum sensitivity importance value for a variable as a maximum over the incoming edges:

Definition 59. Given the set Q of query variables and a factor graph $(\{X,\mathcal{F}\},\mathbb{T})$, the **maximum sensitivity importance value** of a variable x_i is defined to be

$$max\text{-}sensitivity(i,Q) \equiv \max_{\alpha\in\Gamma_i} max\text{-}sensitivity(\alpha - i, Q) = \max_{\alpha\in\Gamma_i}\max_{x_q\in Q}\max_{\pi\in\Pi(\alpha-i,q)} sensitivity(\pi), \tag{4.34}$$

where the second equality follows directly from Def. 51.

We further define the total residual weighted by variable importance, which will be used as variable priority in the anytime modification of QSBP-V, and can also be used in QSBP-V itself:

Definition 60. Given a factor graph $(\{X,\mathcal{F}\},\mathbb{T})$, the set Q of query variables, and a vector r of message residuals, define **total residual weighted by variable importance** for variable x_i as

$$\text{total-residual-vi}(x_i) = max\text{-}sensitivity(i,Q)\cdot\sum_{\alpha\in\Gamma_i} r_{\alpha-i}. \tag{4.35}$$

Before discussing how using (4.35) as variable priorities solves the issues identified in this section with anytime modifications, let us show that (4.35) is a close approximation of the total variable residual weighted with individual edge weights (4.28). In other words, the choice of (4.35) is justified not only by the computational convenience, but also by the approximation quality. The key observation here is the following:

Proposition 61. *For any factor graph $(\{X,\mathcal{F}\},\mathbb{T})$, set Q of query variables and variable x_i, at most one edge $(\alpha-i)\in\mathbb{T}$ involving x_i has $max\text{-}sensitivity(\alpha-i,Q) < max\text{-}sensitivity(i,Q)$. In other words, all the edges involving x_i except at most one have the same importance:*

$$|\{\alpha \mid \alpha\in\Gamma_i,\ max\text{-}sensitivity(\alpha-i,Q) = max\text{-}sensitivity(i,Q)\}| >= |\Gamma_i| - 1.$$

Proof. If $x_i\in Q$, it follows that $\forall\alpha\in\Gamma_i$, $max\text{-}sensitivity(\alpha-i,Q) = max\text{-}sensitivity(i,Q) = 1$. Therefore,

$$|\{\alpha \mid \alpha\in\Gamma_i,\ max\text{-}sensitivity(\alpha-i,Q) = max\text{-}sensitivity(i,Q)\}| = |\Gamma_i| > |\Gamma_i| - 1.$$

Consider the case $x_i\notin Q$. By Def. 59, there exists an edge $(\gamma-i)\in\mathbb{T}$ such that $max\text{-}sensitivity(\gamma-i,Q) = max\text{-}sensitivity(i,Q)$. From Def. 51 and Def. 48, it further follows that there exists a variable $x_q\in Q$ and a path $\pi = (\gamma - i - \beta_1 - j_1 - ...j_m - \beta_{m+1} - q)$ such that $sensitivity(\pi) =$

Algorithm 4.8: Variable importance computation

Input: Factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, query $Q \in X$.

$\mathbb{L} = \emptyset$ is priority queue, $\rho_{\alpha-i}$ is edge priority

foreach $x_i \in X$ **do** initialize $w_i = \emptyset$

foreach $(\alpha - i) \in \mathbb{T}$ **do** add $(\alpha - i)$ to $\mathbb{L}$ with priority $\rho_{\alpha-i} = \begin{cases} 1, & \text{if } i \in Q \\ 0 & \text{otherwise} \end{cases}$

while $\mathbb{L} \neq \emptyset$ **do**

  denote $(\alpha - i)$ to be the top of $\mathbb{L}$

  if $w_i = \emptyset$ **then** set $w_{\alpha-i} \leftarrow \rho_{\alpha-i}$

  remove $(\alpha - i)$ from $\mathbb{L}$

  foreach $j \in \Gamma_\alpha \setminus i$ **foreach** $\beta \in \Gamma_j \setminus \alpha$ **such that** $(\beta - j) \in \mathbb{L}$ **do**

    $\rho_{\beta-j} \leftarrow \max\left(\rho_{\beta-j}, \rho_{\alpha-i} \cdot \sup_\nu \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\| \right)$

    update the position of $(\beta - j)$ in $\mathbb{L}$ to match priority $\rho_{\beta-j}$

return $W = \{w_i \mid x_i \in X\}$ - importance values for all the variables

$max\text{-}sensitivity(i, Q)$. Consider any $\alpha \in \Gamma_i \setminus \beta_1$ and a path $\pi(\alpha) = (\alpha - i - \beta_1 - j_1 - \ldots - j_m - \beta_{m+1} - q)$ obtained by replacing γ with α in π. From (4.10) and (4.12) it follows that

$$\begin{aligned} sensitivity(\pi(\alpha)) &= \sup_{\nu_{-\pi}} \left\| \frac{\partial \nu_{\beta_1 - j_1}}{\partial \nu_{\alpha - i}} \right\| \times \cdots \times \sup_{\nu_{-\pi}} \left\| \frac{\partial \nu_{\beta_{m+1} - q}}{\partial \nu_{\beta_m - j_m}} \right\| \\ &= \sup_{\nu_{-\pi}} \left\| \frac{\partial \nu_{\beta_1 - j_1}}{\partial \nu_{\gamma - i}} \right\| \times \cdots \times \sup_{\nu_{-\pi}} \left\| \frac{\partial \nu_{\beta_{m+1} - q}}{\partial \nu_{\beta_m - j_m}} \right\| = sensitivity(\pi), \end{aligned}$$

where the second equality holds because by (4.12) $\left\| \frac{\partial \nu_{\beta_1 - i_1}}{\partial \nu_{\alpha - i}} \right\|$ does not depend on the factor ψ_α.

Finally, observe that

$$\begin{aligned} max\text{-}sensitivity(i, Q) &= sensitivity(\pi) = sensitivity(\pi(\alpha)) \\ \text{(Def. 51 and Def. 48)} \quad &\leq max\text{-}sensitivity(\alpha - i, Q) \\ \text{(Def. 59)} \quad &\leq max\text{-}sensitivity(i, Q), \end{aligned}$$

so $max\text{-}sensitivity(i, Q) = max\text{-}sensitivity(\alpha - i, Q)$ for every $\alpha \in \Gamma_i \setminus \beta_1$.□

An approximation quality guarantee follows from Prop. 61 directly:

Corollary 62. *For any vector of BP messages ν and corresponding new messages $\widehat{\nu}$ and residuals r for a factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, it holds that*

$$0 \leq \textit{total-residual-vi}(x_i) - \textit{total-residual}(x_i) \leq max\text{-}sensitivity(i, Q) \cdot \max_{\alpha \in \Gamma_i} r_{\alpha - i}.$$

Adjusting the maximum sensitivity importance edge importance computation (Alg. 4.1) and QSBP-V (Alg. 4.7) to work with variable importance values instead of edge importance values, we obtain correspondingly Alg. 4.8 and Alg. 4.9 with similar properties:

Proposition 63. *Alg. 4.8 computes the exact maximum sensitivity importance values of Def. 59 for every variable x_i : on line 11 of Alg. 4.8 it holds that $w_i = max\text{-}sensitivity(i, Q)$.*

Algorithm 4.9: QSBP-V: Query-specific belief propagation with variable updates

Input: Factor graph $G = (\{X, \mathcal{F}\}, \mathbb{T})$, query $Q \in X$.

1 $\mathbb{M}$ is a priority queue
2 $W \leftarrow$ Alg.4.8(G, Q) *// Find variable importance values*
3 **foreach** $x_i \in X$ set $\mathcal{N}_i \leftarrow \left(|\Gamma_i| \cdot |\mathrm{dom}(x_i)| + \sum_{\alpha \in \Gamma_i} (\Gamma_\alpha - 1) \cdot |\mathrm{dom}(\psi_\alpha)|\right)$ *// From* (4.29)
4 **foreach** $(\alpha - i) \in \mathbb{T}$ initialize the message $\nu_{\alpha - i}$
5 **foreach** $(\alpha - i) \in \mathbb{T}$ compute $\widehat{\nu}_{\alpha-i}, r_{\alpha-i}$ using (4.6,4.7)
6 **foreach** $x_i \in X$ add x_i to $\mathbb{M}$ with priority $\mu_i = \mathcal{N}_i^{-1} w_i \sum_{\alpha \in \Gamma_i} r_{\alpha - i}$
7 **while** *not converged* **do**
8 denote x_i to be the top of $\mathbb{M}$, set priority of x_i in $\mathbb{M}$ to 0, set $\mu_i \leftarrow 0$
9 **foreach** $\alpha \in \Gamma_i$ do $\nu_{\alpha - i} \leftarrow \widehat{\nu}_{\alpha - i}$
10 **foreach** $\alpha \in \Gamma_i$ *and* $j \in \Gamma_\alpha \setminus i$ **do**
// Account for changes in μ_j due to changing $r_{\alpha-j}$ to avoid recomputing μ_j from scratch
11 $\mu_j \leftarrow \mu_j - \mathcal{N}_j^{-1} w_j r_{\alpha - j}$ *// Subtract the contribution of the old residual*
12 recompute $\widehat{\nu}_{\alpha-j}$ and $r_{\alpha-j}$ using Eq. 4.6, 4.7
13 $\mu_j \leftarrow \mu_j + \mathcal{N}_j^{-1} w_j r_{\alpha - j}$ *// Add the contribution of the new residual*
14 set priority of x_j in $\mathbb{M}$ to μ_j
15 **return** $\tilde{P}(x_q)$ for $x_q \in Q$ using Eq. 4.5

Proof. Analogous to the proof of a similar proposition for Alg. 4.1 (Prop. 49). From the proof Prop. 49, whenever an edge $(\alpha - i)$ is pulled off the top of the priority queue $\mathbb{L}$, it holds that the priority $\rho_{\alpha-i}$ is equal to the maximum sensitivity importance value of $(\alpha - i)$.

Because Alg. 4.1 (and consequently Alg. 4.8) expand edges in the order of their decreasing maximum sensitivity values, the first expanded edge $(\alpha - i)$ connected to x_i will have $max\text{-}sensitivity(\alpha - i, Q) = max\text{-}sensitivity(i, Q)$. Therefore, setting $w_i = \rho_{\alpha-i}$ on line 6 for the first expanded $(\alpha - i)$ connected to x_i makes w_i equal to $max\text{-}sensitivity(i, Q)$. Because the weights w_i are not changed after the first assignment on line 6, w_i stays equal to $max\text{-}sensitivity(i, Q)$ until the end of the algorithm run.□

Proposition 64. *Suppose every factor of the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ contains at most d_f variables and every variable participates in at most d_v factors. Then the complexity of Alg. 4.8 with priority queue organized as Fibonacci heap is $O(|\mathbb{T}|(\log |\mathbb{T}| + d_f d_v))$.*

Proof. Same as for Prop. 50.□

Although the changes between QSBP-V and Alg. 4.9, highlighted in Alg. 4.9 using red underlined font, are minimal, elimination of edge-specific importance weights in Alg. 4.9 restores the multiplicative decomposition of the priority of the variable into an variable importance and the total unweighted residual. Such a multiplicative decomposition entails a decomposable upper bound on the priority of the yet unseen variables analogous to the upper bound used in Alg. 4.4. The resulting anytime modification of Alg. 4.9, shown in Alg. 4.10, is directly analogous to Alg. 4.4.

One can see that adopting the multiplicative upper decomposition for the upper bound on variable priority (c.f. line 10 of Alg. 4.10) resolves the issue 1 with computational complexity discussed in the beginning of this section. Observe that just as pulling an edge off the top of the importance values priority queue $\mathbb{L}$ on line 15 of Alg. 4.4 leads to at most $(d_v - 1)(d_f - 1)$ updates to $\mathbb{L}$ on lines 16-18, pulling an edge off the top of $\mathbb{L}$ on line 19 of Alg. 4.10 leads to at most $(d_v - 1)(d_f - 1)$ updates to $\mathbb{L}$ on lines 25-27.

Algorithm 4.10: Pessimistic anytime query-specific belief propagation with variable updates

Input: Factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, query set $Q \in X$.

$\mathbb{L} = \emptyset$ is priority queue for computing variable importance, $\rho_{\alpha-i}$ is edge priority in $\mathbb{L}$
$\mathbb{M} = \emptyset$ is priority queue for prioritizing BP updates
foreach $x_i \in X$ **do**
  $\mathcal{N}_i \leftarrow \left(|\Gamma_i| \cdot |\text{dom}(x_i)| + \sum_{\alpha \in \Gamma_i} (\Gamma_\alpha - 1) \cdot |\text{dom}(\psi_\alpha)|\right)$ *// From* (4.29)
  $\overline{\boldsymbol{r}_i} \leftarrow \mathcal{N}_i^{-1} \cdot \sum_{\alpha \in \Gamma_i} 2 \|\psi_\alpha\|$ *// Total complexity-adjusted residual upper bound for* x_i
$\overline{\boldsymbol{r}} = \max_{x_i \in X} \overline{\boldsymbol{r}_i}$ *// Global total complexity-adjusted residual upper bound for all variables*
foreach $x_i \in X$ **do** initialize $w_i = \emptyset$
foreach $(\alpha - i) \in \mathbb{T}$ **do** add $(\alpha - i)$ to $\mathbb{L}$ with priority $\rho_{\alpha-i} = \begin{cases} 1, & \text{if } i \in Q \\ 0 & \text{otherwise} \end{cases}$
while *not converged* **do**
  if $\mathbb{M} \neq \emptyset$ ***AND*** $pt(\mathbb{M}) > \overline{\boldsymbol{r}} \cdot pt(\mathbb{L})$ **then**
    // Variable x_i *is guaranteed to have the largest weighted residual*
    denote x_i to be the top of $\mathbb{M}$, set priority of x_i in $\mathbb{M}$ to $\mu_i \leftarrow 0$
    foreach $\alpha \in \Gamma_i$ do $\nu_{\alpha-i} \leftarrow \widehat{\nu}_{\alpha-i}$
    foreach $\alpha \in \Gamma_i$ **foreach** $j \in \Gamma_\alpha \setminus i$ **s.t.** $w_j \neq \emptyset$ **do**
      $\mu_j \leftarrow \mu_j - \mathcal{N}_j^{-1} w_j r_{\alpha-j}$ *// Subtract the contribution of the old residual*
      recompute $\widehat{\nu}_{\alpha-j}$ and $r_{\alpha-j}$ using Eq. 4.6, 4.7
      $\mu_j \leftarrow \mu_j + \mathcal{N}_j^{-1} w_j r_{\alpha-j}$ *// Add the contribution of the new residual*
      set priority of x_j in $\mathbb{M}$ to μ_j
  else
    // Need to tighten the weighted residual upper bounds
    // or find an variable with a larger exact weighted residual.
    denote $(\alpha - i)$ to be the top of $\mathbb{L}$
    if $w_i = \emptyset$ **then**
      set $w_{\alpha-i} \leftarrow \rho_{\alpha-i}$
      foreach $\alpha \in \Gamma_i$ compute $\widehat{\nu}_{\alpha-i}$ and $r_{\alpha-i}$ using Eq. 4.6, 4.7
      add x_i to $\mathbb{M}$ with priority $\mathcal{N}_i^{-1} w_i \sum_{\alpha \in \Gamma_i} r_{\alpha-i}$
    remove $(\alpha - i)$ from $\mathbb{L}$
    foreach $j \in \Gamma_\alpha \setminus i$ **foreach** $\beta \in \Gamma_j \setminus \alpha$ **such that** $(\beta - j) \in \mathbb{L}$ **do**
      $\rho_{\beta-j} \leftarrow \max\left(\rho_{\beta-j}, \rho_{\alpha-i} \cdot \sup_\nu \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\| \right)$
      update the position of $(\beta - j)$ in $\mathbb{L}$ to match priority $\rho_{\beta-j}$
return $\tilde{P}(Q)$ using Eq. 4.5

In other words, compared to a batch run of Dijkstra's algorithm to compute all the maximum sensitivity importance values in advance, neither Alg. 4.4 nor Alg. 4.10 have any extra overhead concerned with maintaining an upper bound on the exact priority of the yet unseen edges or variables.

Moreover, issue 2 of having to explore low-priority edges is also resolved by adopting (4.10) as variable priority. Because Dijkstra's algorithm processes edges in the order of decreasing importance, it follows that the first edge $(\alpha - i)$ processed on line 5 of Alg. 4.1 involving variable x_i will have the

same maximum sensitivity importance value as the variable x_i itself: $max\text{-}sensitivity(\alpha - i, Q) = max\text{-}sensitivity(i, Q)$. Therefore, instead of waiting for the *last* edge connected to x_i to be processed before computing the exact variable priority value for (4.30), with (4.35) it is sufficient to get the *first* such edge:

Example: consider again the setting in Fig. 4.8. Comparing the total weighted residuals (4.28) using maximum sensitivity importance weights for individual edges (Fig. 4.8b) and the total weighted residuals (4.35) using maximum sensitivity importance weights for variables (Fig. 4.8e), one can see that using per-variable importance weights leads to larger absolute values of the total weighted residuals (although in the case of Fig. 4.8 the relative ordering of the variables by the total weighted residual is the same in both cases).

The key computational difference between the edge-weighted total residual (4.28) and variable-weighted total residual (4.35) is illustrated by Fig. 4.8d and Fig. 4.8f, which show the respective number of edges that need to be expanded by Alg. 4.1 to find the exact total residual for every variable. Notice that first, for every variable variable-weighted total residual requires fewer edges to be expanded than edge-weighted total residual, and second, that the difference is especially pronounced for the variables with the highest residual: while computing the edge-weighted total residual (4.28) for x_{i_1} and x_{i_2} requires expanding 16 and 15 edges correspondingly, computing variable-weighted total residual (4.35) for the same variables requires expanding only 3 and 4 edges, so the first BP update can be applied much earlier with prioritization by variable-weighted total residuals (4.35) as compared to edge-weighted total residuals (4.28).

Finally, just as Alg. 4.2 and Alg. 4.4, it holds that Alg 4.9 and its anytime modification Alg 4.10 are guaranteed to apply BP updates in the same order:

Proposition 65. *Assuming that all the message are initialized to uniform values, $\nu = \vec{0}$, and the same outcomes of breaking ties between variables of equal priority, the sequence of variable updates performed by Alg. 4.10 is the same as for Alg. 4.9.*

Proof. Follows directly from the upper bound on the weighted total residuals for variables x_j with not yet known weight $w_j = \emptyset$:

$$w_j = \emptyset \Rightarrow \text{pt}(\mathbb{L}) \geq max\text{-}sensitivity(j, Q) \text{ and } \overline{\boldsymbol{r}} \geq \mathcal{N}_j^{-1} \sum_{\beta \in \Gamma_j} r_{\beta - j} \Rightarrow$$

$$\Rightarrow \overline{\boldsymbol{r}} \cdot \text{pt}(\mathbb{L}) \geq \mathcal{N}_j^{-1} max\text{-}sensitivity(j, Q) \sum_{\beta \in \Gamma_j} r_{\beta - j}.$$

Because right before a BP update on line 11 of Alg. 4.10 it holds that

$$\overline{\boldsymbol{r}} \cdot \text{pt}(\mathbb{L}) \leq \mu_i \equiv \mathcal{N}_i^{-1} max\text{-}sensitivity(i, Q) \sum_{\alpha \in \Gamma_i} r_{\alpha - i},$$

for x_i that has the highest priority μ_i in $\mathbb{M}$, it follows that x_i has a larger total weighted residual than any variable x_j with $w_j = \emptyset$. Therefore, x_i would be updated next even if all the variables x_j were present in $\mathbb{M}$ with priorities $\mathcal{N}_j^{-1} max\text{-}sensitivity(j, Q) \sum_{\beta \in \Gamma_j} r_{\beta - j}$ as in Alg. 4.9.□

4.6 Related Work

Query-specific inference algorithms introduced in this chapter involve three key ideas: focusing the computation on the parts of the model relevant to the query, taking not only model structure, but also param-

eters into account to quantify the relative importance of the model edges to the query, and finally using the analysis of interdependence of messages based on the work of Mooij and Kappen (2007) as a concrete way to tractably compute the edge importance *without running the inference itself first.* Here, we discuss the existing literature touching these three topics.

We also discuss more broadly the existing work on estimating the intervals that are guaranteed to contain of the fixed point BP beliefs of the full model. Those interval estimation approaches are more computationally demanding that our one-shot edge sensitivity estimates, but also have higher accuracy. Therefore, if one can overcome the computational issues, employing those more accurate estimates to prioritize belief propagation updates can further speed up the inference convergence.

4.6.1 Query-specific model simplification

For a factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, the inference problem of computing marginal distributions of variables $P(x)$ can be hard for two primary reasons:

1. The model $(\{X, \mathcal{F}\}, \mathbb{T})$ may have a lot of factors with high dynamic range, which tightly couple the values of multiple variables. Because the "standard" belief propagation-based algorithms update the messages one edge or variable at a time, it is easy for them to get stuck in a cycle oscillating between several sharp modes of the marginal distribution. Most of the theoretical criteria that guarantee BP convergence (c.f. Mooij and Kappen, 2007, and references therein) rely on the requirement that the factors $\mathcal{F}$ do not induce dependencies that are too strong. It is important to notice that the model $(\{X, \mathcal{F}\}, \mathbb{T})$ with strong dependencies between variables does not need to be very large for the inference problem to be difficult.

2. The model $(\{X, \mathcal{F}\}, \mathbb{T})$ is large, with many factors and variables. The issue of the sheer size of the graphical model has become especially acute with the wide adoption of relational models (Friedman et al., 1999; Richardson and Domingos, 2006; Taskar et al., 2002; Pentney et al., 2006). For the large-scale model, even if the variable dependencies are relatively weak and belief propagation converges after a moderate number of iterations, the cost of even a single BP iteration for every message may be quite large.

In the former case it may not be possible to accurately approximate the query belief $P(Q)$ without performing inference for the full model, because every factor may significantly affect $P(Q)$ via the strong dependencies. In the latter case, however, intuitively one expects that parts of the model that are far away from the query for some notion of distance in the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ have little effect on $P(Q)$. Therefore, one can simplify the inference problem by simply discarding the parts of the model that are far away from the query.

Although the approach of completely removing parts of the model that are far away from the query and performing inference in the simplified model instead of the full one has been successfully used in practice previously (Wellman et al., 1992; Pentney et al., 2006, 2007). However, the existing work in this direction has suffered from two significant problems, both of which have been addressed in our approach:

1. Query-specific submodel selection was usually done by breadth-first search in the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, including all the variables and potentials within a fixed radius from the query (Wellman et al., 1992; Pentney et al., 2006). Such an approach only pays attention to the model structure and completely ignores the parameters of the model, resulting in suboptimal simplified models.

In (Pentney et al., 2007), the strength of dependencies is taken into account by pruning the variables x that have low mutual information $I(x, Q)$ with the query. However, such a solution is also problematic, because it requires estimating all the pairwise mutual information values either directly from data, which may be difficult for relational models because of the lack of independent datapoints, or using inference in the full factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, which goes back to a problem that query-specific approaches aim to speed up in the first place.

2. Removing some factors and variables from the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ changes both the exact distribution induced by the factorized model (4.1) and belief propagation fixed points $\tilde{P}^*(X)$. Moreover, removing parts of the model introduces a tradeoff of approximation quality with respect to the full model and the error stemming from stopping BP before achieving convergence. Formally, let $\tilde{P}^*(X)$ to be a BP fixed point for the full model $(\{X, \mathcal{F}\}, \mathbb{T})$. Denote $\tilde{P}^*_{reduced}(X)$ to be the fixed point of a simplified model and $\tilde{P}_{reduced}(X, \tau)$ to be the belief obtained by running belief propagation for time τ on the reduced model. The total error of the BP belief can be bounded from above as

$$\|\tilde{P}^*(X) - \tilde{P}_{reduced}(X, \tau)\| \leq \|\tilde{P}^*(X) - \tilde{P}^*_{reduced}(X)\| + \|\tilde{P}^*_{reduced}(X) - \tilde{P}_{reduced}(X, \tau)\|, \tag{4.36}$$

where the first component is approximation error *inherent to the reduced submodel* and the second component is the error resulting from the *limited runtime* τ of belief propagation, which may be not sufficient for BP to achieve convergence on the simplified model.

Typically, the fewer elements we remove from $(\{X, \mathcal{F}\}, \mathbb{T})$, the smaller the resulting changes in the fixed points would be. Therefore, the first error component decreases as the size of the simplified model grows. On the other hand, the larger the simplified model, the slower BP progress towards the fixed point would be, so the second error component grows with the size of the simplified model.

Ideally, one wants to select the optimal size of the reduced submodel given the time budget τ. However, without taking model parameters into account, it is hard to imagine a principled way to find a good tradeoff point of the two error sources in (4.36) other than cross-validation, which is likely to be costly. The problem becomes even harder when the time budget τ is not known in advance and a way to incrementally grow the simplified model over time until it includes all of $(\{X, \mathcal{F}\}, \mathbb{T})$ is needed.

One can see that our query-specific approach based on edge importance weights (4.13) addresses both of the above issues. By taking into account the model parameters via the strength of message depenencies (4.12), the computation is better focused on parts of the model that most affect the query. Moreover, QSBP can be seen as adaptively growing the query-specific submodel over time in a principled way, eventually converging to the fixed point of a full factor graph. The submodel-growing interpretation is especially clear in the structure of anytime variants Alg. 4.4 and Alg. 4.10 of the basic QSBP, where the edges and variables with exactly known importance weights can be interpreted as the current query-specific submodel. Also, unlike previous approaches, one can guarantee that whenever QSBP converges, the resulting query beliefs correspond to a BP fixed point for the full model.

4.6.2 Estimating the effects of model simplification

In the previous section, we have discussed the construction of query-specific submodels via breadth-first search on the full factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ - a computationally cheap approach that often works well in

practice, but lacks the theoretical guarantees on the changes in the query marginal $P(Q)$ introduced by removing some parts of $(\{X, \mathcal{F}\}, \mathbb{T})$. Here, we discuss the approaches that form the field of *sensitivity analysis* of probabilistic graphical models, which belong to the opposite extreme of the complexity-accuracy tradeoff. Typically, these methods allow for accurate estimation of the effects of removing an edge from a factor graph, but have high computational complexity.

A common property of the approaches for model simplification discussed here is the need, in order to estimate the effect of removing an edge from a model, to perform inference (either exact or approximate) in the initial graphical model or in its simplified version. For example, in (Kjaerulff, 1993) a junction tree corresponding to the triangulated version of the PGM is first constructed, and then large cliques of the junction tree are broken down into smaller ones by introducing additional independence assertions. The choice of which independence assertions to introduce is guided purely by the local approximation error $\min_{\psi_{\alpha_1}, \psi_{\alpha_2}} D(\psi_\alpha \mid \psi_{\alpha_1} \cdot \psi_{\alpha_2})$, where X_α is the clique to be broken down into X_{α_1} and X_{α_2}, and $D(\cdot||\cdot)$ is an error measure such as KL divergence. Together with a union bound on the effect of a sequence of clique simplifications in a model, Kjaerulff (1993) propose a simple greedy approach of removing the weakest dependencies from the model. While such an approach works well for small models, the need to build a junction tree of the triangulated model and operate with clique potentials of that junction tree makes the it intractable for large-scale problems. In (van Engelen, 1997), a similar approach is proposed that works directly with Bayesian networks and does not need to form a triangulated model as an intermediate step. Unfortunately, the approach of van Engelen (1997) is restricted to directed graphical models (also known as Bayesian networks) and is therefore inapplicable for the majority of large-scale relational PGMs, which are undirected (c.f. Taskar et al., 2002; Richardson and Domingos, 2006). Moreover, the approach of van Engelen (1997) does not have a notion of query and is thus only able to optimize approximation quality of the full joint distribution $P(X)$ instead of the query marginal $P(Q)$.

A different line of work (Choi and Darwiche, 2008, 2006a, 2009, 2006b) focuses on the idea of (a) replacing a pairwise factor $\psi_{ij}(x_i x_j)$ with a pair of single-variable factors $\psi_i(x_i)$ and $\psi_j(x_j)$, (b) establishing conditions on the values of the new factors $\psi_i(x_i)$ and $\psi_j(x_j)$ that would minimize the approximation error and (c) using the optimality conditions in (b) as fixed point equations. For the case of replacing every pairwise factor with single-variable factors (i.e., fully disconnecting the variables) in such a manner, the fixed point equations yield exactly the belief propagation updates. Moreover, after recovering a connected tractable submodel of $(\{X, \mathcal{F}\}, \mathbb{T})$, such as a spanning tree, Choi and Darwiche (2006a) show ways to heuristically estimate the overcounting of influence by BP updates due to loops in $(\{X, \mathcal{F}\}, \mathbb{T})$. As Choi and Darwiche (2006a) have shown experimentally, further recovering the edges with the "most overcounting" significantly improves approximation quality of generalized belief propagation (interleaved exact inference on the recovered part of the factor graph and BP updates for the edges that were not recovered). In (Choi and Darwiche, 2008), a query-specific heuristic was introduced to focus on recovering the edges that have the largest impact on the query. The key differences of the approach of Choi and Darwiche (2008) (and other related papers) and our query-specific inference algorithms is the computational complexity: although the heuristics of Choi and Darwiche (2008) may be more accurate than ours, their computational complexity is much higher: while running belief propagation on the full model $(\{X, \mathcal{F}\}, \mathbb{T})$ is the highest possible complexity procedure for our approach, Choi and Darwiche (2008) need the results for the generalized belief propagation on the full model just to evaluate the importance of different edges to the query. After some of the edges are recovered, another run of GBP for the full model is needed to improve the approximation $\tilde{P}(Q)$. Therefore, our query-specific inference approach and that of Choi and Darwiche (2008) occupy non-overlapping intervals on the computational complexity spectrum, and the choice between the two would be dictated by the amount of resources available for a

given application.

4.6.3 Convergence analysis for belief propagation

Because belief propagation is not guaranteed to converge in general loopy models (Mooij and Kappen, 2005), an important problem is to predict, without running inference itself, whether BP will converge *given a particular model* $(\{X, \mathcal{F}\}, \mathbb{T})$. Similarly to the problem of optimal updates prioritization, convergence analysis requires one to consider the long-term impact of the message updates. In fact, the upper bound (4.12) on the immediate impact of an update on the neighbor messages was developed by Mooij and Kappen (2007) exactly for the problem of convergence analysis.

As Mooij and Kappen (2007) have shown, synchronous BP is guaranteed to converge whenever the matrix A of size $|\mathbb{T}| \times |\mathbb{T}|$ such that $A_{(\alpha-i),(\beta=j)} = \left\| \frac{\partial \nu_{\alpha-i}}{\partial \nu_{\beta-j}} \right\|$ has the spectral radius $\rho(A) < 1$. Moreover, Mooij and Kappen (2007) have shown their spectral radius criterion to outperform similar in spirit convergence conditions from the earlier literature, such as those of Tatikonda and Jordan (2002) and Georgii (1988). An equivalent to guarantee of Mooij and Kappen (2007) based on the spectral radius of the edge matrix, but restricted only to models with pairwise factors, was independently introduced by Ihler et al. (2005).

Both our notion of edge importance and the sufficient conditions for BP convergence from Mooij and Kappen (2007) are based on the upper bound (4.12) and a worst-case analysis of the message interactions in the model. As a result, the two approaches have many important properties in common. First, neither approach requires iterative updates with uncertain convergence horizon (as would be the case if one had to run belief propagation on the model first). As a result, both approaches are not only computationally efficient, but also are *predictable* in terms of the required computation. Second, both approaches share the drawback of somewhat crude results, which stems from the worst-case assumptions that (a) every message update results in the largest possible changes for the neighbor messages and (b) the messages themselves can be arbitrary. Additionally, our notion of maximum sensitivity edge importance only takes into account one possible path of the influence flow in the model. As a result of these simplifications, our edge importance values are only heuristic estimates, which can both over-estimate the edge influence on the query (in a tree-like model) and underestimate edge importance in a densely connected model. Similarly, the sufficient conditions for BP convergence of Mooij and Kappen (2007) are often too strict, and fail to hold for many models where belief propagation is actually well-behaved. Moreover, for cases when the conditions of Mooij and Kappen (2007) do not hold and convergence is not guaranteed, nothing can be said about the possible errors of BP beliefs.

In the next section, we discuss how one can obtain more detailed information about the behavior of belief propagation at the expense of larger computational cost.

4.6.4 Bounds on belief propagation beliefs

Instead of analyzing the model $(\{X, \mathcal{F}\}, \mathbb{T})$ before running belief propagation, one can propagate the error information on the BP beliefs *during the process of belief propagation itself* using upper bounds similar to (4.12) on the sensitivity of the outgoing messages with respect to the incoming messages. Suppose at some point during the inference process, for every factor graph edge $(\alpha - i)$ a multi-dimensional "box" (i.e., a Cartesian product of intervals for every dimension) is known that is guaranteed to contain the fixed point message $\nu^*_{\alpha-i}$. Then (4.12) or a similar bound can be used to iteratively tighten the "boxes"

with updates analogous to the standard BP updates (4.4). Fortunately, suitable initial boxes are easy to obtain using, for example, the upper bound (4.19) or even vacuous boxes that contain all possible messages.

Interval estimates of the fixed point BP messages can be used in two different ways. The first way is more passive: one can simply replace the standard belief propagation messages $\nu_{\alpha-i}$, which are essentially point estimates of the fixed point messages, with the bounding boxes, and use "box propagation" updates instead of (4.12) to iteratively tighten the box estimates. Importantly, this first group of approaches does not use the information about bounds to prioritize updates. For example, Mooij and Kappen (2008) proposed update rules such that the box estimates of the beliefs $\tilde{P}(x_i)$ are guaranteed to contain both fixed point BP beliefs $\tilde{P}^*(x_i)$ and the true marginals $P(x_i)$. Ihler (2007) use bounds propagation to refine the upper bounds on the difference between the fixed point beliefs and the true marginals.

The second group of approaches not only uses box propagation to obtain *a posteriori* error bounds on the BP beliefs, but also relies on the error bounds to actively guide the prioritization of updates. For example, in a lifted setting, de Salvo Braz et al. (2009) use error propagation analysis to reduce the size of the lifted network by treating messages that are not exactly equal, but similar, as the same message. Finally, in (Nath and Domingos, 2010) error propagation analysis is used to speed up inference by avoiding BP updates that are guaranteed to have small eventual effect on the query marginals. In particular, Nath and Domingos (2010) formulate conditions under which it is possible to ignore a BP update without affecting the MAP assignment.

The latter group of approaches has a lot in common with our query-specific inference algorithms. Two their main common properties are (a) using (4.12) and similar bounds to estimate the *local* effects of the belief propagation updates, and (b) using the estimates of the update effects to guide the prioritization of updates. However, there are also important differences between our approach and box propagation-based approaches of de Salvo Braz et al. (2009) and Nath and Domingos (2010):

- First, our query-specific belief propagation only requires $O(|\mathbb{T}|(\log|\mathbb{T}|+d_f d_v))$ time to run Alg. 4.8 as a preprocessing step (c.f. Proposition 64), and constant time per update afterwards. Box propagation approaches, however, being a generalization of belief propagation, are in general not guaranteed to converge. Therefore, our approach has an advantage in computational complexity over box propagation algorithms.

- Second, while our notion of maximum sensitivity edge importance characterizes the strength of the *long-range* dependencies between an arbitrary message and the query, box propagation updates only describe the immediate dependencies between messages of the *directly adjacent* edges. As a result, in a query-specific setting it is not immediately clear how to adapt box propagation to characterize relative importance of message updates that are not directly adjacent to the query. In particular, both de Salvo Braz et al. (2009) and Nath and Domingos (2010) use a rather crude approach of updating all the messages where the box estimates have diameter larger than a fixed threshold.

- Finally, while maximum sensitivity edge importance measure of Def. 48 only estimates the eventual impact of a message update on the query belief heuristically and without any quality guarantees, box propagation-based approaches operate with actual upper and lower bounds on the BP messages. In other words, box propagation algorithms provide much more reliable information about the eventual inference errors than our approach.

One can see that neither query-specific inference of this chapter nor the existing box propagation-based approaches dominates the alternative. Our edge importance measures based on path sensitivity are more

efficient computationally and better suited to estimating long-term effects of a BP update, while box propagation techniques are significantly more informative for the immediate effects of an update on the neighboring messages. Therefore combining the advantages of the two to improve the quality of the estimates of long-term effects of an arbitrary BP update on the query beliefs is an important open problem. One possible direction towards such a unified approach is to develop a better measure of path strength to replace the maximum sensitivity of Def. 48. The more accurate path strength measure would need to take into account not only the largest possible sensitivity of local dependencies via the upper bound (4.12), but also the possible ranges of BP messages along that path. Taking the possible messages ranges along the path into account would make it possible to tighten the current upper bound on the update impact on the query along the path, which in turn would improve update scheduling.

4.7 Experiments

In this section, we empirically demonstrate the performance of query-specific inference on real-life relational probabilistic graphical models.

4.7.1 Datasets and models

We have evaluated the query-specific belief propagation approach on a grounded Markov logic network for the WebKB webpage classification problem and the high-treewidth CRFs for the image segmentation problem (see sections 3.6.2 and 3.6.3 for the detailed description of the two models). We have also applied our approach to a grounded Markov logic network describing the structure of an academic department, which is described below.

UW-CSE academic department model

This Markov logic network model, introduced in (Richardson and Domingos, 2006) and extended in (Singla and Domingos, 2005) describes the structure of a university department, and was successfully applied to predicting a "professor A is a an advisor of student B" relation for the CSE department at the University of Washington. UW-CSE MLN has 22 first-order predicates, 10 constant types and 94 hand-coded formulas encoding domain-specific knowledge such as "a student typically has only one advisor", or "a student and his advisor are likely to be coauthors on a paper". The data gathered for the CSE department at UW spans students and faculty in five distinct areas (AI, graphics, programming languages, theory and systems). The goal is to use the data from 4 areas to train a model, and then predict which faculty member advises which student for the remaining one area. The value of every predicate of the MLN except for `AdvisedBy` is observed at both train and test time, enabling discriminative weight learning. The resulting factor graph sizes are shown in Table 4.1. Observe that the UW-CSE models are much more densely connected than the WebKB and image segmentation models: UW-CSE models have about 50 factors per variable versus 2-3 factors per variable for WebKB and image segmentation.

Model	Variables	Factors	Edges
UW-AI	4790	$3.4 \cdot 10^5$	$6.8 \cdot 10^5$
UW-GRAPHICS	6678	$2.5 \cdot 10^5$	$4.9 \cdot 10^5$
UW-SYSTEMS	7951	$4.1 \cdot 10^5$	$8.0 \cdot 10^5$
UW-THEORY	2527	$1.3 \cdot 10^5$	$2.6 \cdot 10^5$

Table 4.1: Factor graph sizes for UW-CSE Markov logic network.

4.7.2 Query selection, inference problems and error measures

The motivation for developing the query-specific algorithms described in this chapter was to improve *convergence speed* of belief propagation, as opposed to *approximation accuracy* in the limit of plentiful computation time. The question of choice between belief propagation and fundamentally different alternative inference methods such Gibbs sampling (Geman and Geman, 1984) or variational inference (Jordan et al., 1999) is outside of the scope of this thesis. Here, we will assume that the set of approximations entailed by choosing BP for inference leads to a satisfactory end result (such as classification accuracy) for the models at hand and will only compare the inference approaches based on belief propagation. Such a restriction makes it possible to evaluate convergence speedups independently of the approximation quality of BP fixed points. Specifically, we will focus on the *distance from the BP fixed point* as the main quality measure of the set of dingle-variable beliefs:

$$\text{FPError}(\tilde{P}(X), Q) \equiv \frac{1}{|Q|} \sum_{x_q \in Q} KL(\tilde{P}^*(x_q) || \tilde{P}(x_q)),$$

where $\tilde{P}(\cdot)$ is the current set of beliefs and $\tilde{P}^*(\cdot)$ is a belief propagation fixed point for the complete factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$. We will also demonstrate the results for classification accuracy as the quality measure in section 4.7.4.

Fixed point detection. The exact fixed point beliefs are unknown for our models. Instead, we use approximate fixed points found by running belief propagation until the total residuals (4.28) decrease below 10^{-10} for every variable. A residual threshold of 10^{-10} is a very strict convergence criterion in practice - noticeable changes in beliefs stop long before such low residuals are achieved, so one can be quite confident that the resulting BP beliefs lie very close to an exact fixed point.

Multiple fixed points. We have found empirically that some of the models used in this section have multiple BP fixed points: depending on the starting point or update schedule (in particular, different queries lead to different update schedules for query-specific algorithms), belief propagation may converge to different beliefs and messages with an extremely low residual. Because our initial assumption was that BP approximation is an adequate one, we consider every BP fixed point to be equally good, and use the distance to the *closest fixed point* as quality measure for the beliefs:

$$\text{FPError}(\tilde{P}(X), Q) \equiv \min_{i=1,\ldots,k} \frac{1}{|Q|} \sum_{x_q \in Q} KL(\tilde{P}_i^*(x_q) || \tilde{P}(x_q)), \tag{4.37}$$

where $\tilde{P}_1^*(X), \ldots, \tilde{P}_k^*(X)$ are different BP fixed points. Observe that it is only possible to possible to compute the nearest distance (4.37) to the closest *known* fixed point. Therefore, the values one gets in practice form an *upper bound* on the nearest distance to the nearest existing fixed points, because there may be other fixed points that remained undetected.

Inference problems. To obtain a representative performance picture, instead of reporting the results of running inference algorithm for one fixed query, we will report averages over sets of many queries (the question of selecting interesting sets of queries will be discussed shortly). Let

$$\mathbb{Q} = \cup_i \{Q_i\}, Q_i \subseteq X \forall i$$

be a set of queries. For every $Q_i \in \mathbb{Q}$, we run an inference algorithm in question with Q as a query. For belief propagation-based algorithms, the belief $\tilde{P}(Q)$ is a function of time τ: $\tilde{P}(Q) = \tilde{P}(Q, \tau)$. It follows that the belief error (4.37) is also a function of time $\text{FPError}(\tilde{P}, Q) \equiv \text{FPError}(Q, \tau)$. To aggregate the information for multiple queries, we will focus on the evolution of *average query belief error* with time, where the averaging is all queries from $\mathbb{Q}$:

$$\text{FPError}(\mathbb{Q}, \tau) = \frac{1}{\sum_{Q \in \mathbb{Q}} |Q|} \sum_{Q \in \mathbb{Q}} |Q| \cdot \text{FPError}(Q, \tau). \tag{4.38}$$

The key thing to notice about the average error (4.38) is that it only includes the error component $\text{FPError}(Q, \tau)$ for the run of the inference algorithm with Q as query and disregards the belief errors for Q for all the other runs.

We will use the following query sets $\mathbb{Q}$ in our evaluation:

1. All possible single-variable queries: $\mathbb{Q} = \cup_{x_i \in X} \{x_i\}$.
2. A set of multivariable queries that together cover all the variables X. Here, we split X into non-intersecting subsets of the same size k. In our experiments, $k = 30$. The performance of query-specific inference depends on the way of grouping the variables into query sets. In general, performance is better when all the query variables are close to each other in the factor graph and worse if the query variables are spread out: when there are multiple queries far away from each other, the algorithm is unable to narrowly focus the computation, because different query variables are affected by different parts of the factor graph. We have used the following partitioning approaches:
   - For WebKB, the query sets were constructed using breadth-first search. This is the easiest multivariable query setting.
   - For image segmentation models, we subdivided every image into a uniform 3×3 grid, and grouped the variables corresponding to the superpixels from the same grid cell into the same query set, resulting in 9 query sets per image. Here, the query set size is somewhat smaller than 30 and depends on the number of superpixels in every grid cell.
   - For UW-CSE dataset, we grouped query variables by the full name of the corresponding grounding of a first-order predicate. This is the hardest multiquery setting, as the variables are spread out through the graph.
3. Single-variable queries that are the "hardest" for the non-query-specific residual BP. After running the standard residual belief to convergence, one can identify *in the hindsight* which variables took the longest time to compute fixed point beliefs for. There are different ways to define how hard a query is, we used two approaches. Denote τ^* to be the time it takes residual belief propagation to converge. Define the *total RBP error* of a variable to be:

$$\text{FPTotalError}(Q) = \int_{\tau=0}^{\tau^*} \text{FPError}(Q, \tau) d\tau, \tag{4.39}$$

Model	Effective damping level
WebKB	0.1
Segmentation	0.1
UW-CSE	0.5

Table 4.2: Message damping levels used for the respective models in the experiments. These are the smallest possible damping levels for which belief propagation converged for every update schedule.

where $FPError(Q, \tau)$ is computed for RBP beliefs. One can see that (4.39) considers to be hard not only query variables where the beliefs take a long time to converge, but also the variables where the belief converge relatively fast, but the instantaneous error values are high initially. The latter situation (relatively quick convergence with briefly spiking error in the beginning) occurs quite often when the immediate neighborhood of a variable x_i in the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ induces a significantly different belief $\tilde{P}(x_i)$ from the full factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$. As will be shown in section 4.7.4, such a conflict of local subgraph of x_i and the full $(\{X, \mathcal{F}\}, \mathbb{T})$ occurs quite often for the UW-CSE model. To filter out the error spikes during the initial stage of the inference, we also adopt a more robust measure of total belief error:

$$\text{FPTotalMinError}(Q) = \int_{\tau=0}^{\tau^*} \left(\min_{\tau' \in [0,\tau]} \text{FPError}(Q, \tau') \right) d\tau. \tag{4.40}$$

One can see that the total measure (4.40), unlike (4.39), disregards the temporary increases in the instantaneous error $\text{FPError}(Q, \tau')$. Effectively, (4.40) uses the smallest distance from the fixed point achieved *in the hindsight* within the first τ time units as the instantaneous error measure.

For our experiments, for every factor graph we first filtered out the variables $x \in X$ with fixed point entropy $H_{\tilde{P}^*}(x) < 0.1$, and from the remaining variables selected 30 query variables with the largest total errors (4.39) and (4.40). We will denote the corresponding query sets as Q_{total} and $Q_{total\text{-}min}$. Low fixed point entropy $H_{\tilde{P}^*}(x)$ corresponds to having almost all of the probability mass of $\tilde{P}^*(x)$ to be on the single value of x. We filter out such variables from the set of candidate queries for two reasons. First, a sharply peaked fixed point belief is likely to be an artifact of belief propagation, which is known in loopy models to produce more concentrated beliefs $\tilde{P}^*(x)$ compared to the true marginals $P(x)$. Second, for problems such as classification, one can be quite confident in the maximum probability assignment $\arg\max_{\boldsymbol{x}} \tilde{P}^*(\boldsymbol{x})$ even with relatively large KL divergence $KL(\tilde{P}^*(x) \,||\, \tilde{P}(x))$. In other words, for variables with low fixed point entropy, a disproportionately large share of the error (4.37) comes from the question "how sharply peaked is the fixed point?" as opposed to "what does the fixed point roughly look like?". We argue that the latter question is typically more important in practice, and therefore concentrate on queries with relatively large fixed point entropies.

Choosing parameters

In practice, to achieve convergence of belief propagation one needs to *damp* message updates. For an old message $m_{\alpha-i}$ and a new message $\widehat{m}_{\alpha-i}$, instead of simply replacing the old value with the new one, an affine combination of the two is used:

$$m_{\alpha-i} \leftarrow d \cdot m_{\alpha-i} + (1-d) \cdot \widehat{m}_{\alpha-i}. \tag{4.41}$$

The parameter $d \in (0, 1)$ is called damping (c.f., Mooij and Kappen, 2005; Elidan et al., 2006; Heskes, 2002). Typically, smaller damping values lead to faster, but more brittle convergence: depending on the model $(\{X, \mathcal{F}\}, \mathbb{T})$ the messages either converge quickly, or enter an oscillation and do not converge at all. Correspondingly, larger damping values lead to slower, but more robust convergence. In our experiments, we used for every model the smallest damping value for which BP messages converged for *every update schedule.* The differences in update schedules for the same model stem from various factors:

1. The choice between edge updates of Alg. 4.5 versus variable updates of Alg. 4.6.

2. Different prioritization criteria: scheduling by maximum edge residual $\max_{(\alpha-i)\in\mathbb{T}} r_{\alpha-i}$, total variable residual (4.28) and total variable residual normalized by the update complexity (4.30).

3. Different variable importance weights induced by different queries.

The resulting damping values are shown in Table 4.2.

4.7.3 Validating non-query-specific heuristics

Before proceeding to the results of query-specific inference, we need to establish the baselines to compare against. As we have discussed in section 4.7.2, we will regard BP fixed points as the ground truth and will therefore concentrate only on belief propagation-based baselines. Moreover, we will also exclude approaches that perform query-specific model simplification, such as those of Pentney et al. (2006), because simplifying a model almost always changes the fixed points of the query beliefs. Because we consider computing a BP fixed point to be the ultimate goal of the inference, results on the simplified models cannot be directly compared with the results on the full model. However, even within the framework of belief propagation on the full model $(\{X, \mathcal{F}\}, \mathbb{T})$, there exists a wide variety of baselines corresponding to different update prioritization schemes. In this section, we will validate three heuristics for improving the performance of the standard residual belief propagation of Elidan et al. (2006) that were discussed in this chapter:

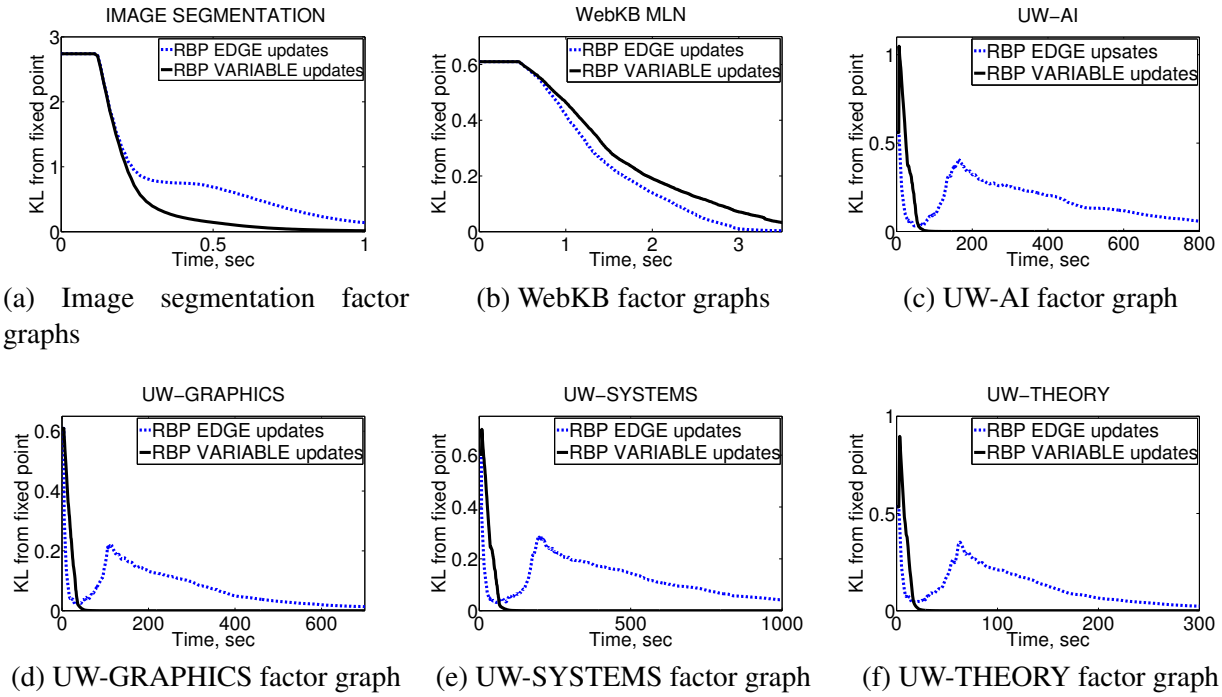
1. Replacing single-edge message updates of Alg. 4.5 with batch message updates of Alg. 4.6 for all the edges connected to the same variable.

2. Instead of prioritizing the updates by the maximum edge residual, with the priority corresponding to x_i equal to $\mu_i = \max_{\alpha\in\Gamma_i} r_{\alpha-i}$, use the total residual (4.28) of all the edges connected to x_i.

3. Finally, normalize the total variable residual (4.28) by the update complexity (4.29). As a result, prioritize the updates by the normalized residual (4.30).

We will start with the standard RBP of Elidan et al. (2006), which corresponds to Alg. 4.2 with uniform edge importance weights, and show that sequentially enabling every heuristic in the order they are listed above results in an inference speedup. As a result, we will demonstrate that residual belief propagation with batch variable updates and prioritization by total variable residual normalized by update complexity is the best performing baseline against which to compare our query-specific approaches.

For all the experiments in this section, we will plot the evolution over time of the KL divergence of single-variable beliefs from the BP fixed point, averaged over all the model variables X. Equivalently, we look at fixed point error (4.37) for $Q = X$.



(a) Image segmentation factor graphs (b) WebKB factor graphs (c) UW-AI factor graph

(d) UW-GRAPHICS factor graph (e) UW-SYSTEMS factor graph (f) UW-THEORY factor graph

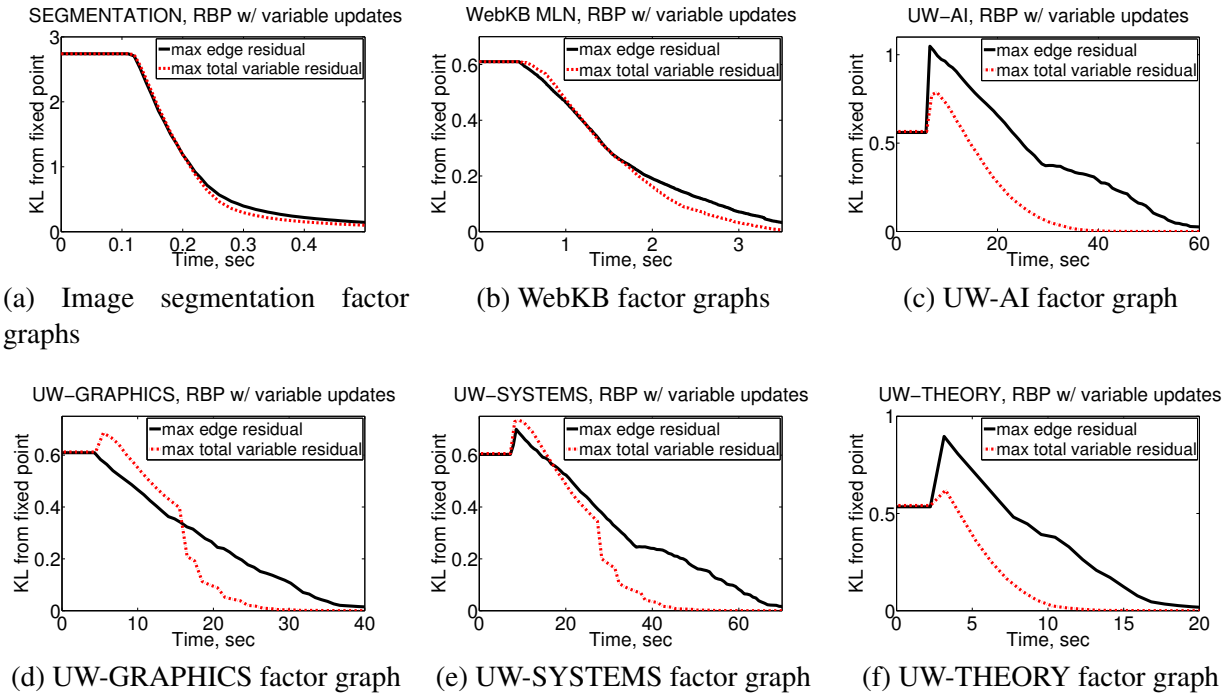
Figure 4.9: Comparison of belief propagation convergence speed for edge updates of Alg. 4.5 (dashed blue lines) and batch variable updates of Alg. 4.6 (solid black lines). Plotted is the average single-variable belief error of belief propagation (averaged over all factor graph variables X) versus time. Batch variable updates yield dramatically faster convergence for the densely connected UW-CSE models. The effect on the sparsely connected models is either less pronounced (SEGMENTATION) or slightly detrimental (WebKB).

Batch incoming messages updates

For the densely connected UW-CSE models, by far the largest speedup compared to the standard residual belief propagation is obtained by replacing the edge updates of Alg. 4.5 with the batch variable updates of Alg. 4.6. The dependence of average belief error on the runtime of the algorithm is shown in Fig. 4.9. As we have discussed in section 4.5, batch variables updates are on the order of d_v times more efficient than single edge updates, where d_v is the number of edges per variable in the model. One can see from Tables 3.1 and 4.1 that the factor graphs used in these experiments have very different connectivity densities: from 4-6 edges per variable for WebKB and image segmentation to 10^2 edges per variable for UW-CSE. The results in Fig. 4.9 confirm the dependence of the speedup of the inference convergence on the model connectivity: for sparsely connected models, speedups are either moderate (segmentation) or nonexistent (WebKB). Still, the speedup penalty for WebKB is much smaller than the advantages of variable updates for other models, justifying the choice of variable updates.

Prioritization by total residual of incoming edges

Having switched to a more efficient update scheme, we turn to improving the prioritization heuristic. Here, we compare prioritization by the maximum edge residual $\mu_i = \max_{\alpha \in \Gamma_i} r_{\alpha - i}$, with the total resid-



(a) Image segmentation factor graphs (b) WebKB factor graphs (c) UW-AI factor graph

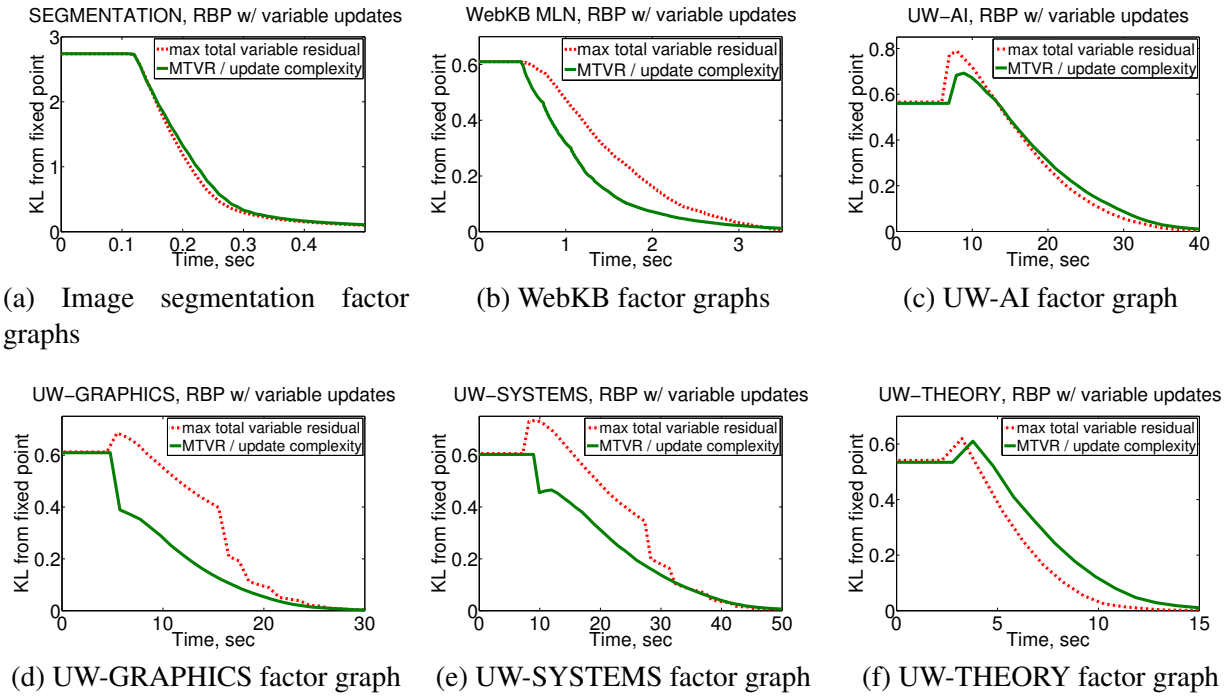
(d) UW-GRAPHICS factor graph (e) UW-SYSTEMS factor graph (f) UW-THEORY factor graph

Figure 4.10: Comparison of belief propagation convergence speed for update prioritization by maximum edge residual (solid black lines) and by total variable residual (dashed red lines).

ual (4.28) of all the edges connected to the variable x_i. The dependence of belief errors on time is shown in Fig. 4.10. One can see that prioritization by the total residual (4.28) again yields faster convergence for the densely connected models, although sometimes with higher errors in the initial stages of the inference (Fig. 4.10d and 4.10e). For sparsely connected models, there is no difference between the two prioritization schemes.

Normalizing by update complexity

Finally, we investigate the impact of normalizing the update priorities by the total update complexity. In Fig. 4.11, we plot the average beliefs error versus time for the prioritization by the total residual (4.28) and total residual (4.29) normalized by the update complexity. One can see that taking update complexity into account does not always lead to faster convergence. Still, normalizing by update complexity dominates the unnormalized total residual for 3 models out of 6 (WebKB, UW-GRAPHICS, UW-SYSTEMS), leads to slower convergence for UW-THEORY, makes no difference for UW-AI and image segmentation. On average over the datasets in question, batch variable updates prioritized by the total residual divided by update complexity (4.29) form the best non-query-specific RBP baseline, and we will use the complexity normalization (4.29) in the remaining experiments.



(a) Image segmentation factor graphs (b) WebKB factor graphs (c) UW-AI factor graph

(d) UW-GRAPHICS factor graph (e) UW-SYSTEMS factor graph (f) UW-THEORY factor graph

Figure 4.11: Comparison of belief propagation convergence speed for update prioritization by total variable residual (Eq. 4.28, dashed red lines) versus the total residual normalized by the update complexity (Eq. 4.30, solid green lines).

4.7.4 Results

Here, we evaluate the convergence speed of the inference algorithms for two error metrics: average KL distance from the fixed point (4.38) and its analogous counterpart for the accuracy:

$$\text{FPRelativeAccuracy}(\mathbb{Q}, \tau) = \frac{1}{\sum_{Q \in \mathbb{Q}} |Q|} \sum_{Q \in \mathbb{Q}} \min_{i=1,\ldots,k} \frac{1}{|Q|} \sum_{x_q \in Q} \mathcal{I}(\arg\max \tilde{P}_i^* (x_q) = \arg\max \tilde{P} (x_q)). \tag{4.42}$$

One can see that FPRelativeAccuracy$(\cdot)$ is the measure of prediction accuracy of current beliefs $\tilde{P}(x_q)$ under the assumption that the prediction corresponding to maximizing the fixed-point beliefs $\tilde{P}_i^*(x_q)$ comprise the ground truth of the labeling. Just like FPError$(\cdot)$, the relative accuracy FPRelativeAccuracy$(\cdot)$ lets one sidestep the question of the quality of the BP fixed points during the evaluation and concentrate on the convergence speed. We compare the results for the query-independent RBP-V and our query-specific approaches QSBP-V and pessimistic anytime QSBP-V. In the remaining of this section, all mentions of the anytime QSBP refer to the pessimistic version (Alg. 4.10).

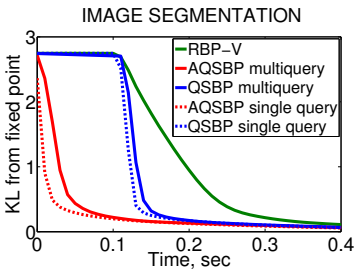
KL divergence

Consider the evolution over time of the KL distance of query belief from the BP fixed points averaged over all possible single-variable queries, and also for one partitioning of all variables into multivariable queries of size 30. The corresponding plots are in Fig. 4.12. We make the following observations:

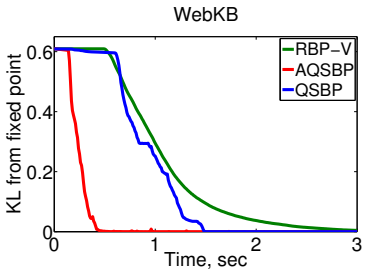
- In the setting of single-variable queries, our query-specific algorithms converge significantly faster than the baseline residual belief propagation throughout the spectrum of the models.
- Also, in the setting of single-variable queries, the anytime modifications of query-specific algorithms outperform the baseline versions with upfront initialization of the full set of messages and edge importance computation.
- In the multivariable query setting, the anytime approach either outperforms the baseline QSBP (on the easy WebKB and image segmentation problems) or provide the same performance (UW-CSE, except for the GRAPHICS dataset).
- In all settings except for one (AQSBP on UW-GRAPHICS) every query-specific approach converges faster than the baseline RBP-V.
- Query-specific inference yields speedups compared to the baseline residual BP not only in the settings where the factors near the query induce a query belief that is largely consistent with the BP fixed point on the full graph (WebKB, segmentation), but also when local potentials induce dramatically different beliefs from the fixed point. In figures 4.12c-4.12f, one can see that the query beliefs initially move *away* from the fixed point, indicating that the potentials closest to the query induce a significantly different belief from the full factor graph. However, over the full duration of inference, the query-specific approach converges to the fixed point faster even in the presence of conflicting potentials.
- When the queries are close by in the graph (WebKB, SEGMENTATION), the performance of query-specific inference is almost identical for multiple queries and a single query variable. In other words, by including a single variable into the query, one gets the results for the variables close by "for free". On the other hand, when the queries are more spread out through the graph (UW-CSE), the performance degradation in the multivariable query setting compared to a single-query setting can be substantial.
- The initialization times of QSBP and RBP-V (corresponding to the initial horizontal parts of the plots, where the messages are initialized, residuals are computed and beliefs are uniform) are almost the same, so running the Dijkstra's algorithm on the factor graph is not a significant overhead.
- It follows that the performance gains of the anytime QSBP compared to the baseline QSBP with upfront initialization result from deferring the computation of the new BP messages and their residuals for the far away parts of the factor graph.

One can conclude that first, for moderate-sized query sets the query-specific inference approach of this chapter provides a clear advantage over the standard residual belief propagation in terms of convergence speed. Second, the anytime modifications of Alg. 4.10 should be always used in practice, because, with very rare exceptions, the anytime algorithm will match, and often significantly exceed, the performance of the baseline QSBP.

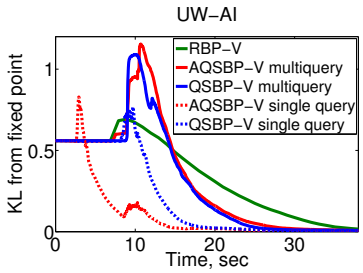
Difficult queries. In addition to the average case, we investigated the behavior of query-specific inference on variables that are especially problematic for the baseline residual BP. We have selected 30 variables for every grounding of the UW-CSE models with the largest values of cumulative inference errors FPTotalMinError$(\cdot)$and FPTotalError$(\cdot)$ (c.f. equations (4.40) and (4.39)). The corresponding results for average KL divergence from the fixed points are in Fig. 4.13 and Fig. 4.14. One can see that, same as with the average case, for single-variable queries, our query-specific algorithms provide significant speedups



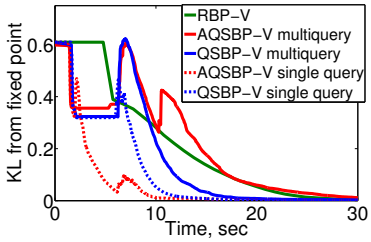
(a) Image segmentation



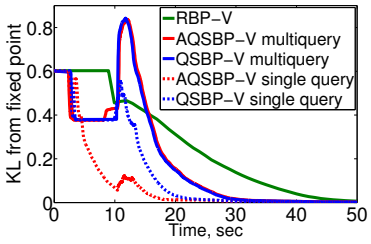
(b) WebKB average KL from fixed point. Results for single-query runs are not shown because they are the same as for the corresponding multiquery runs.



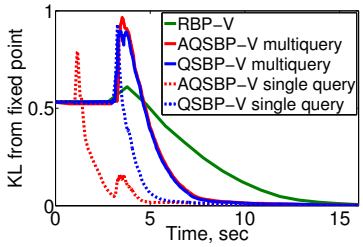
(c) UW-CSE model, AI area dataset.



(d) UW-CSE model, graphics area dataset.



(e) UW-CSE model, systems area dataset.

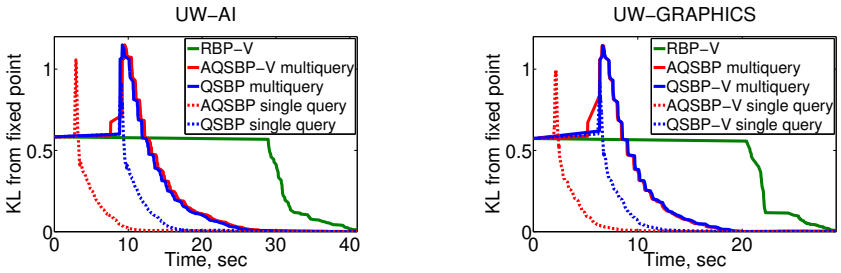


(f) UW-CSE model, theory area dataset.

Figure 4.12: Average per-variable KL divergence of BP beliefs from the fixed points over all the variables X of the respective models.

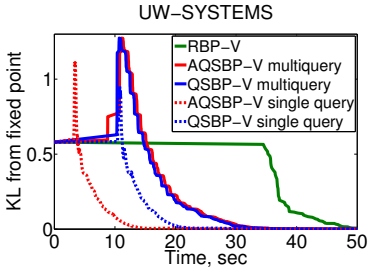
over the baseline residual BP, and also anytime QSBP perform as good as or better than the baseline QSBP.

However, there are also important differences in the behavior of errors over time that illustrate the different sources of difficulties in arriving at a fixed point of belief propagation and show how the effectiveness of QSBP may vary. From Fig. 4.13, one can see that the hardest queries in terms of the error measure FPTotalMinError (Eq. 4.40), are those that residual BP touches only late in the inference process, which is indicated by the long initial time when the beliefs do not change for RBP in Fig. 4.13. In other words, the absolute residual values for those variables are typically smaller than for other variables in the factor graph. Query-specific prioritization of updates for such queries yields robust convergence speedups, both in single query and in the multiquery settings. However, observe that with the query-specific inference approaches, the query beliefs initially move away from the fixed point, which is indicated by the upward spikes in the plot. At the same time, there are no upward spikes in the KL divergence plot for the baseline RBP. It follows that arriving at the BP fixed points for the query variables depends substantially on computing accurate beliefs for other, non-query variables, and simply restricting the consideration to the parts of the model closest to the queries will not produce the same beliefs. The soft prioritization of our query-specific approach is essential for achieving the correct beliefs.

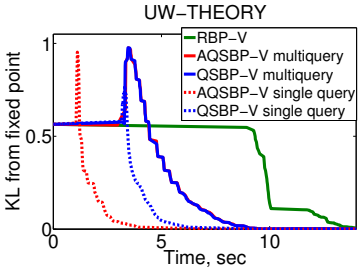


(a) UW-AI

(b) UW-GRAPHICS



(c) UW-SYSTEMS

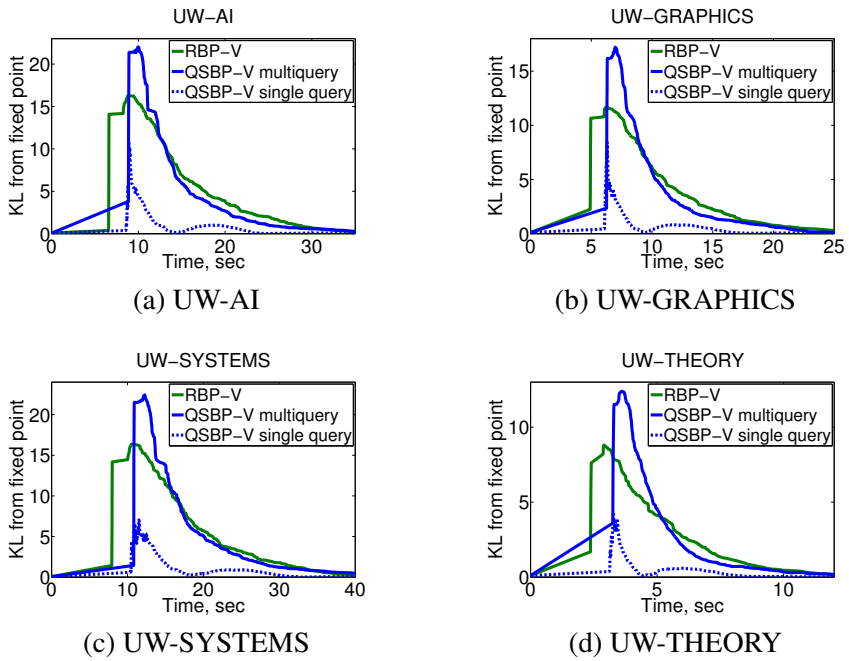


(d) UW-THEORY

Figure 4.13: Average per-variable KL divergence of BP beliefs from the fixed point over the 30 *hard queries* – variables with the largest hindsight error FPTotalMinError (Eq. 4.40) for residual BP.

Next, we consider the difficult queries as measured by the largest error measure FPTotalError (Eq. 4.39), which, unlike Eq. 4.40 does not take into account how close the query beliefs were to the fixed point in the past. One can see that the evolution of KL divergence from the fixed point of such queries, shown in Fig. 4.14, is very different from the queries with the largest cumulative error (4.40), which are discussed above and shown in Fig. 4.13. As Fig. 4.14 shows, the queries with large error measure (4.39) are difficult not because RBP spends computation resources on other areas of the factor graphs with larger residuals (the first BP updates are applied to these queries right after the initialization of messages), but because the part of the factor graph closest to the query induces drastically different query beliefs than the full factor graph. This phenomenon is illustrated in Fig. 4.14 by the sharp increase in the KL distance from the fixed point resulting from the initial BP updates. Moreover, the quality of beliefs for these query variables depends substantially on the convergence of non-query beliefs for a large portion of the full model. This dependence is indicated in Fig. 4.14 in three ways:

- In the non-query-specific case, the query error decreases relatively slowly.
- The anytime version of query-specific belief propagation (Alg. 4.10) has the same performance as the non-anytime QSBP (Alg. 4.9), which means that the upper bound (4.18) on the importance of the edges with not yet known exact importance weight is unable to guarantee that the parts of the factor graph away from the query have small enough impact on the query belief.
- In the multi-query setting, QSBP convergence speed is the same as the baseline RBP-V, which means that in order to compute accurate beliefs for the 30 query variables, one also needs to compute accurate beliefs for almost the remaining variables in the model.



(a) UW-AI (b) UW-GRAPHICS

(c) UW-SYSTEMS (d) UW-THEORY

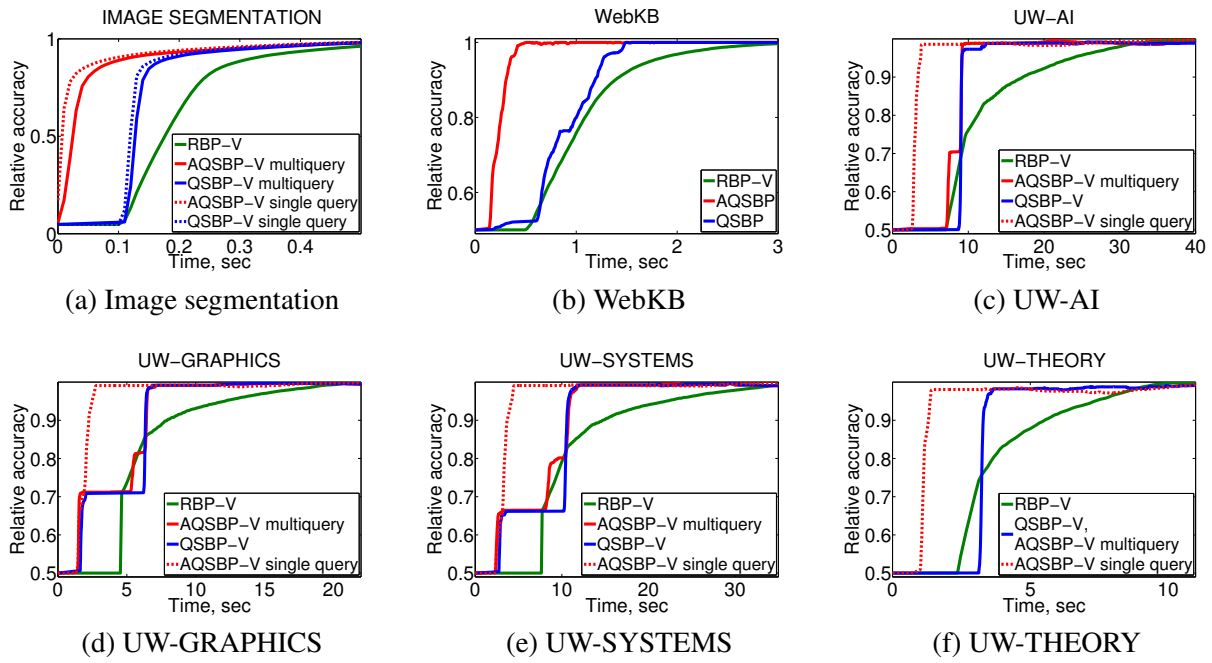
Figure 4.14: Average per-variable KL divergence of BP beliefs from the fixed point over the 30 *hard queries* – variables with the largest hindsight error FPTotalError (Eq. 4.39) for residual BP. Plots for the anytime modifications are not shown, because they are the same as for the non-anytime versions.

Because, for this particular choice of query variables, query beliefs substantially depend on a large share of the full factor graph, performance degradation of the multivariable query setting compared to the single variable query setting is more pronounced, with 30-variable queries yielding the same convergence speed as the baseline RBP-V. However, in the single-query setting query-specific belief propagation provides much faster convergence compared to RBP even in the presence of long-range dependencies.

Classification accuracy

To investigate the performance of query-specific inference for structured prediction, in this section we consider the dependence of the relative classification accuracy (in other words, the share of variables where the modes of BP beliefs match the modes of the fixed point beliefs, c.f. equation 4.42) on time. The results, plotted in Fig. 4.15, show the relative classification accuracy averaged over all single-variable queries and the same multi-variable queries as the KL divergence results of Fig. 4.12.

The problem of finding the modes of the fixed point beliefs is somewhat easier than the problem of approximating the beliefs themselves well, because it is sufficient for the belief to be in a certain neighborhood of the fixed point to have the same mode. The results of Fig. 4.15, while generally matching the results for KL divergence in Fig. 4.12, confirm this reduced problem difficulty. For "easy" models (WebKB, segmentation), where nearby factors mostly determine the fixed point beliefs, the relative accuracy behaves the same as the KL divergence. For the more difficult UW-CSE model, however, the relative accuracy evolves in a much "better behaved" way, with monotonic convergence both for the baseline RBP-V and for the query-specific inference. Moreover, for UW-CSE models, the advantage of query-specific inference over



(a) Image segmentation (b) WebKB (c) UW-AI

(d) UW-GRAPHICS (e) UW-SYSTEMS (f) UW-THEORY

Figure 4.15: Average prediction accuracy relative to the BP fixed points of the respective models.

the standard residual BP is more pronounced in terms of relative accuracy than in terms of KL divergence from the fixed points.

The difference in convergence speeds for UW-CSE is explained by the following: while the magnitude of impact of any single variable on the average accuracy (4.42) is at most $\frac{1}{\sum_{Q \in \mathbb{Q}} |Q|}$, a sharply peaked belief with an incorrect mode can in principle have an arbitrary large impact on the average KL divergence from the fixed point (4.42). Moreover, as Fig. 4.39 shows, there are variables in the UW-CSE MLNs that have sharply peaked BP beliefs *with wrong mode* during the course of inference, resulting in large KL divergence values from the fixed point, on the order of 10. Such variables have a disproportionately large effect on the average KL divergence compared to the "well-behaved" variables that remain near the BP fixed points. For the average accuracy, however, the effect of such hard queries is limited, because the single-variable accuracy is bounded to be between 0 and 1. Therefore, the average accuracy as the quality measure simply places less importance on the few difficult queries compared to the KL divergence.

To summarize, in terms of both average KL divergence from the fixed point beliefs and the relative accuracy with respect to fixed point beliefs, our query-specific inference approach provides significantly faster convergence in both single-query and multiquery settings compared to residual belief propagation. Moreover, the anytime QSBP (Alg. 4.10) yields as good or better performance compared to the baseline QSBP (Alg. 4.9) throughout the spectrum of the models we have experimented with, indicating that the anytime QSBP approach is the optimal choice in practice for the problem of query-specific inference.

4.8 Discussion and future work

In this chapter we have developed the following main contribution:

1. A general framework of importance-weighted residual belief propagation (query-specific BP, Alg. 4.2 and its more much efficient modification, Alg. 4.9) that allows one to significantly speed up convergence for query variables by focusing computation appropriately. Importantly, unlike the previous approaches, ours does not involve any simplification of the model $(\{X, \mathcal{F}\}, \mathbb{T})$, and thus does not change either the true distribution induced by $(\{X, \mathcal{F}\}, \mathbb{T})$ or the set of BP fixed point.

2. A principled measure of edge message importance (4.13) and variable belief importance (4.34) with respect to a given query that can be computed efficiently and yield significant convergence speedups for the query beliefs in practice.

3. Anytime modification of query-specific belief propagation (algorithms 4.3, 4.4 and 4.10) that allows one to postpone examining parts of the full model $(\{X, \mathcal{F}\}, \mathbb{T})$ that have little influence on the query until later stages of the inference process, leading to additional convergence speedups for the query beliefs compared to the standard query-specific BP. Importantly, despite deferring some of the initialization and edge weighting until after some BP updates are done, the order in which the updates are applied is guaranteed to be the same as for QSBP with full initialization. It follows that the anytime modifications do not change the inference results.

Next, we describe some of the directions for future future work, which fall into three broad categories: further improving the performance of belief propagation in the query-specific setting, adapting the ideas of this chapter to the non-query-specific setting (in other words, having all of the unknown variables X to be the query), and finally applying similar ideas to speed up inference techniques other than belief propagation.

4.8.1 Further improvements in the query-specific setting

Lifted inference and weighting. Most of the large scale probabilistic graphical models are groundings of relational models, where variables represent certain *types* of entities or properties, and factors represent relations between classes of entities. Because the factors model the interactions on the level of types of variables, as opposed to concrete variables (also called ground atoms), the same factor parameters are replicated multiple times throughout the model. For example, in the image labeling model, often a contrast-dependent factor encoding the smoothness of labels for neighboring pixels is used: $\psi_{xy}(x_i, x_j) = \exp\{\boldsymbol{u} \cdot \|color_i - color_j)\|\}$. The weight $\boldsymbol{u}$ is learned on the level of a relation ("neighbor pixels with the same color tend to correspond to the same object"), and then copied throughout the model as necessary. The same approach is taken by the Markov logic networks (Richardson and Domingos, 2006) and other relational model formalisms.

Large-scale replication of only a small number of distinct potential introduces a lot of symmetries into the grounded model. One consequence of those symmetries is that BP messages for many edges may be the same: if the factors are the same (because they are just copies of the same factor and link variables of the same type) and the incoming messages are the same, then the outgoing message will also be the same. Although typically such symmetries would be imperfect, because of different evidence for different ground atoms, for many models exploiting the remaining symmetries may dramatically speed up inference. Inference algorithms that aim exploit symmetries by working on the level of types of variables, without grounding the model, are called *lifted.* For Markov logic networks, Singla and Domingos (2008) developed a lifted version of belief propagation and demonstrated significant speedups over the standard BP on a grounded model.

In a query-specific setting, it is desirable to combine the speedups of lifted BP with speedups of the query-specific BP described in this chapter. A corresponding extensions looks to be straightforward, exploiting the organization of lifted BP algorithm. Lifted BP processes in two stages: first, the progress of BP is emulated to trace the effects of the evidence and find out which edges are guaranteed to have the same BP messages (initially, all the edges of the same kind are assumed to have the same message, as the effects of evidence propagate, sets of edges that may have different messages are split up as necessary). The result of the first stage is a *lifted network*: a factor graph where every edge represents a group of edges of the grounded model that are guaranteed to have the same BP message on every iteration. The second stage of lifted BP simply runs belief propagation, with somewhat modified update equations to account for effects of multiple copies of the same message, on the lifted network. It is the second stage of lifted BP that should be possible to speed up using edge importance weighting similar to the approach of this chapter.

The resulting query-specific lifted BP would first build the lifted network (on this step the information about the query should not be introduce to preserve the symmetries as much as possible). The second step would be to identify the variables of the lifted network corresponding to the query and run the importance weighting algorithm like Alg. 4.1. Finally, the third stage would be to run residual BP with importance-weighted residuals on the lifted network to compute the marginals of interest. The only challenge of such an approach lies on the second step and does not look like a significant bottleneck. One would have to adjust the derivative upper bound (4.12) to correspond to the lifted BP message updates so that the importance weights correspond to the structure and magnitude of BP message dependencies in the lifted network.

Larger update units and parallelization. As we have discussed in section 4.5 of this chapter, and demonstrated empirically in section 4.7, replacing single-edge message updates (Alg. 4.5) used in QSBP (Alg. 4.2) with batch updates of all messages connected to a certain variable (Alg. 4.6) leads to dramatic speedups of inference. Moreover, prioritizing the variable updates by the total residual (4.35) of incoming messages instead of the maximal single-edge residual improves performance even further.

The batch approach to message updates is not new to this work. In fact, updating all incoming messages for a single variable is a special case of a more general approach - performing *exact inference* for a *subtree* of the full factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$ given the current values of all the messages not in that subtree. Prior research has exploited different advantages of exact inference on subtrees over single-edge updates. Tree-reweighted belief propagation (Wainwright et al., 2003) uses exact inference on subtrees as a means of making larger and "more informed" steps to the BP fixed point, because an update to a tree brings residuals for all the message of the tree to zero simultaneously. Residual splash BP of Gonzalez et al. (2009), aiming to perform multiple BP updates simultaneously to take advantage of multiprocessor systems, adopts subtree updates as a way to reduce the contention for the global update priority queue and the corresponding locking overhead. Because a subtree update, which Gonzalez et al. (2009) call a *splash*, requires more computation in absolute terms than updating a single edge[1], every processor spends more time actually performing BP updates and less time waiting for a lock on the priority queue compared to a naïve parallelization of residual BP.

Although both tree-reweighted BP (Wainwright et al., 2003) and residual splash BP (Gonzalez et al., 2009) have demonstrated significant speedups over standard belief propagation, the questions of constructing good subtrees (for tree-reweighted BP) and choosing which subtree to update next (for both

[1]It is important to distinguish here the absolute complexity of an update, which is roughly linear in the tree size, from complexity per updated edge, which is roughly constant regardless of the tree size.

tree-reweighted BP and residual splash BP) are approached in a rather ad hoc manner. Wainwright et al. (2003) only require that every edge of the model is updated frequently enough. Gonzalez et al. (2009) use a reasonable heuristic for constructing the subtrees (greedily grow a tree choosing variables with the largest residual), but prioritize the subtrees only by the total residual of the root variable. Such a prioritization scheme (a) leads to redundancies in the elements of the update priority queue, because the subtrees grown from two adjacent root variables would have a lot of common elements, and (b) ignores the residuals of the elements of subtree beyond the root, leading to suboptimal accuracy of the priority score. Therefore, we conjecture that there is potential for improving the performance of residual splash BP (and tree-reweighted BP, since TRBP can be seen as a special case of residual splash BP) by (a) using importance-weighted edge residuals in the query-specific setting instead of currently used unweighted edge residuals, (b) explicitly scheduling the subtrees to be updated as opposed to root variables, which would allow to better estimate the update impact (c.f. max-edge score $\max_{(\alpha-i)\in\mathbb{T}} w_{\alpha-i}r_{\alpha-i}$ versus total variable residual score (4.35)) and eliminate the redundancies in the updates priority queue.

Using the query-specific edge importance weights in the residual splash BP is straightforward for the sequential implementation of Dijkstra's algorithm. Moreover, any advances in exploiting parallelism for Dijkstra's algorithm or any other shortest-path algorithms can be immediately adopted for speeding up the computation of the edge importance weights. The issue of directly scheduling subtree updates (instead of variables that act as roots of dynamically grown subtrees) is a more complicated one. On the one hand, ignoring edge residuals beyond the subtree root leads to a relatively non-informative priority heuristic. On the other hand, greedily growing a subtree by adding the neighbor variables with the largest residual makes it possible to choose from a very large number of possible subtrees without explicitly representing all the possibilities. The residual-driven approach to subtree construction is an important factor to residual splash BP performance, especially in the late stages of inference, when most of the edges have low residual and it is important to prune the update subtrees to avoid wasting computation on low-residual edges. On the opposite end of the tradeoff between the amount of information available for prioritizing updates and flexibility in constructing the shape of best BP update for the current values of the messages would be an approach where a set of "representative" subtrees that would cover all of the edges of the model is selected a priori. Ideally, every edge would be covered by only a few of the representative subtrees, so that changing the residual of an edge would lead to re-prioritization of only a small number of updates. Such an approach is likely to work well in the initial stages of the inference, when most edges have significant residuals, but is likely to be wasteful in the later stages, when it is important to concentrate computation on the small number of high-residual edges. Therefore, one needs a compromise solution to combine the advantages of the two approaches. For example, one can use low-depth subtrees as representative updates and dynamically grow those subtrees further by adding high-residual neighbor variables at runtime. Another option is to use large representative subtrees, but dynamically prune the low-residual elements at runtime and only update the high-residual forests. It remains to be seen what the optimal solution is.

4.8.2 Dealing with non-query-specific inference

The approach of this chapter is built on the idea of estimating the eventual effect of BP updates on the query more accurately than existing techniques do, and to use those more accurate estimates to better prioritize computation. In particular, we have concentrated on exploiting the information about the query to improve the accuracy of update effect estimates. However, the idea of better predicting the effects of a BP update can be applied more generally. Here, we discuss some directions of improving updates

prioritization in a non-query-specific setting.

Non-QS edge importance.

Maximum sensitivity edge importance value (4.13) is one possible approximate answer to the question "if the message for this edge is changed by a fixed amount, how much will the query beliefs eventually change?". In principle, one can cast the regular probabilistic inference problem where the query is not specified as a query-specific problem with every variable being in the query, $Q = X$. However, weighting the edge residuals by their maximum sensitivity importance value would not bring any advantages in such a setting. By definition 51, when $Q = X$ it holds that every edge of the model has the same importance of 1, so query-specific BP degenerates into standard residual BP. More generally, because of the maximization over the query variables in (4.13), maximum sensitivity importance value is better suited for relatively small query sets. As one could see in section 4.7, the performance advantage of QSBP over residual belief propagation decreases as the query set size grows. Therefore, in a non-query-specific setting a different approach to edge weighting is needed.

A natural alternative to the maximum sensitivity importance value of an edge (4.13) is, instead of maximizing over the query variables, to use the total sensitivity with respect to all of the other variables of the model:

$$\textit{total-max-sensitivity}(\alpha - i, X \setminus x_i) \equiv \sum_{x_q \in Q} \max_{\pi \in \Pi(\alpha - i, q)} \textit{sensitivity}(\pi).$$

Unfortunately, actually computing such a score is impractical, because it requires running Dijkstra's algorithm (more precisely, Alg. 4.1) $|X|$ times, once for every variable. Therefore, more tractable approximations are necessary. One way to improve tractability, similar to the idea behind anytime modifications such as Alg. 4.3, is instead of running Alg. 4.1) to completion for every variable $x_q \in X$, to stop as soon as the upper bound $\text{pt}(\mathbb{L})$ on the sensitivity of the remaining edges with respect to x_q is low enough.

Learning the effects of updates.

Although the upper bound (4.12) on the partial derivative of BP messages outgoing for a certain factor ψ_α with respect to the incoming messages typically provides a good indication of the strength of local dependencies, for models where the factors have extremely high dynamic ranges, such as protein side-chain prediction models (Yanover et al., 2007), the upper bound is "saturated" and takes values very close to 1. As a result, the sensitivity of every simple path in those models is also very close to 1. It follows that neither the query-specific approach of this chapter, nor the non-query-specific edge weighting proposed in the previous section would have any advantage over the standard residual BP, because the sensitivities of every edge with respect to every variable will be almost the same (close to 1).

One possible way to address the issue of saturation of the upper bound (4.12) is to attempt to *learn* the effects of the BP updates during the course of the inference process itself. Because the choice of the next update depends on the estimate of the magnitude of the eventual effects of that update, learning update effects essentially casts inference as a reinforcement learning problem (Kaelbling et al., 1996), where the set of actions is the set of available updates (for example, a set of all variables X for batch updates of Alg. 4.9), and the reward is, for example, the negative total residual of all the edges.

The main potential advantage of the reinforcement learning formulation is the more detailed information that can potentially be exploited. Because the actual residual changes and update effects can be observed during the initial stages of the inference, one does not have to resort to upper bounds such as (4.12) and instead attempt to better capture the average-case behavior of updates propagation, which is often a more useful estimate. Moreover, eliminating the reliance on (4.12) also lets one use more detailed information

about the state of the system, compared to just the residuals in L_∞ norm in log-messages space. Recall that the choice of the norm for the residual is largely dictated by the fact that for other choices of norms there are no known closed-form expressions like (4.12) for the upper bounds on the magnitude of partial derivatives of messages. For example, if the upper bound such as (4.12) is not needed, nothing prevents one from using features such as element-wise differences between new messages $\widehat{\nu}_{\alpha-i}$ and old messages $\nu_{\alpha-i}$ to describe the state of the system.

The main potential problem of a reinforcement learning approach to updates prioritization is computational complexity. Because residual BP updates are quite cheap computationally, any approach to updates prioritization can only be successful if its complexity is relatively low. Otherwise the standard RBP will have faster convergence simply via much higher throughput in terms of updates compared to a complicated "smart" prioritization technique. Therefore, controlling the complexity of the reinforcement learning technique, via selecting appropriate features to describe the state of the system, tractable value functions and learning rules, will be crucial to the overall performance on the inference.

4.8.3 Beyond belief propagation

A more open-ended direction of future work is adapting the ideas of this chapter to other inference approaches. In particular, **Gibbs sampling** (Geman and Geman, 1984) is in many ways similar to belief propagation:

1. **Local computation dependencies.** While belief propagation iteratively recomputes messages that only depend on neighbor messages in the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, Gibbs sampling iteratively updates the assignments to individual variables $x_i \in X$ by sampling from a conditional distribution $P(x_i \mid \boldsymbol{X}_{\Gamma_i})$ of x_i given the current assignment $\boldsymbol{X}_{\Gamma_i}$ to the neighbor variables of x_i.
2. **Large variety of valid schedules.** As Geman and Geman (1984) have shown, under reasonable assumptions on the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$, for any sequence in which the individual variables are sampled, as long as every variable is sampled infinitely often, it holds that the resulting distribution over X converges to the true distribution $P_{(\{X,\mathcal{F}\},\mathbb{T})}(X)$ defined by the factor graph $(\{X, \mathcal{F}\}, \mathbb{T})$. Similar to belief propagation, although practitioners typically use round-robin update schedules because of their simplicity, smart update prioritization has potential to significantly speed up convergence of Gibbs sampling for many models.

The key difference between the requirements to a belief propagation schedule and a Gibbs sampler schedule is that unlike BP schedule, a Gibbs sampler schedule cannot depend on the outcomes of the samples. A "residual Gibbs sampler" in general may converge to a wrong distribution (Gonzalez et al., 2011). However, even with the sampling schedule independent of the system state, there are still possibilities for focusing computation better. In a query-specific setting, one can adjust the frequency of sampling different variables according to those variables importance to the query, relying, for example, on variable importance weights similar to (4.34).

In fact, once we allow for the requirement that the Gibbs sampling schedule be independent of the actual sampling outcomes, most of the ideas discussed both in the main body of this chapter and in the future work section in the context of belief propagation can be also applied to the Gibbs sampling:

- **Larger update units.** Similar to the advantages of exact inference on model subtrees in belief propagation, the advantages of *block updates* in Gibbs sampling have been long recognized (Jensen et al., 1995; Barbu and Zhu, 2005). With block updates, groups of variables are sampled from their

joint conditional distribution given their Markov blanket, which significantly speeds up mixing in the presence of highly correlated variables. Sampling from the joint distribution $P(Y \mid \boldsymbol{X} \setminus Y)$ can be done at the same cost as exact inference in the submodel of $(\{X, \mathcal{F}\}, \mathbb{T})$ restricted to the variables of Y. Therefore, ideal Gibbs update blocks have low treewidth to achieve tractability and high intra-block dependencies to obtain a benefit from the sampling variables jointly instead of separately. These requirements are the same as for BP update subtrees discussed in section 4.8.1. Therefore, the advances in designing high-quality update subtrees for residual BP can be also applied to designing update blocks for Gibbs sampling.

- **Variable importance in the non-query-specific setting.** Estimating the share of a model that is substantially affected by changes to an assignment of a given variable x_i, similarly to an approach proposed in the first half of section 4.8.2, can be also used to adjust the resampling frequency of x_i.
- **Adjusting for update complexity.** The same argument as in section 4.5.1 advocating, with other factors being equal, updating more frequently the parts of the model where updates are cheaper computationally applies to Gibbs sampling as well.

The main challenge in adapting the formalisms based on belief propagation to be used in Gibbs sampling would be to replace the dependence between messages as a basis of the formalism with a more direct measure of dependence between assignments to different variables or between their respective conditional distributions given the Markov blankets. For example, mutual information between two neighbor variables x_i and x_j given the assignment to the rest of the variables $\boldsymbol{X} \setminus \{x_i, x_j\}$ is one such possible measure.

Chapter 5

Conclusions

In this thesis, we aimed to improve, in terms of both accuracy and efficiency, the quality of reasoning under uncertainty that can be achieved using the formalism of probabilistic graphical models. We explored two general directions: (a) learning accurate models that admit efficient exact inference, whereby the decrease in representation power compared to the common high-treewidth model is compensated for by the better inference accuracy, and (b) focusing the computation on the parts of the model relevant to the query. Our main contributions are a novel approach for learning low-treewidth models with quality guarantees in the generative setting, learning a novel class of tractable models with *evidence-specific* structure in the discriminative setting, and a principled approach to prioritize computation during inference so as to significantly speed up the computation of query marginals. Although quite different technically, all three main contribution share the property of significantly improving *test-time* computational efficiency, which makes it possible for new application areas that are sensitive to inference latency to benefit from the PGM formalism and techniques.

In terms of the schematic illustration of Fig. 1.1, our solutions make progress towards the optimum from both ends of the complexity spectrum. From the "simple models" end, we have achieved the accuracy of state of the art high-treewidth models with much more computationally efficient low-treewidth models. From the "complex models" end, we have reduced the effective inference complexity of the rich models with query-specific prioritization. Both general types of approaches can be developed further; we have outlined in the respective chapters some of the possible directions of improving the accuracy of tractable models and transferring the ideas of the query-specific inference into a more widely applicable setting with no nuisance variables.

We believe that speeding up approximate inference in high-treewidth model by improving the prioritization of computations is more likely to provide significant benefits for applications in the short term, because of the abundance of the existing high-treewidth models. In the longer term, however, we believe that high-accuracy tractable models (or "almost tractable" with few loops) provide a more promising avenue for development. Besides the guaranteed inference accuracy, a key advantage of tractable graphical models is the possibility to adjust complexity in a controlled manner by increasing the treewidth. In high-treewidth models, such an adjustment is complicated by the lack of guarantees on the convergence of iterative algorithms.

More generally, until recently, the research focus and the source of much of the success of probabilistic graphical models in practice was exploiting compact representation of probability distributions with

general-purpose approximate inference algorithms designed to handle an arbitrary compact structure. We believe that the next significant step in probabilistic improving graphical models can come from shifting the focus from the structure of representation to the structure of computation necessary for inference. This thesis presents one step towards fully exploiting the computational structure of probabilistic graphical models for efficient and accurate reasoning with high-dimensional distributions.

Appendix A

Proofs for chapter 2

A.1 Example: increasing treewidth does not preserve strong connectivity

Here, we present an example of how the "fattening" procedure of Section 2.2.5, applied to a ε-strongly connected junction tree may result in a 0-strongly connected JT. It follows that the "fattening" may decrease the strong connectivity of a structure by an arbitrary large amount. Consider a junction tree with the cliques $abc - bcd - cde - def$. Suppose we have added variable a to every clique of the JT to obtain $abcd - acde - adef$. Let us construct the distribution parameters such that the original clique is ε-strongly connected for some $\varepsilon > 0$, but $I\left(b, c \mid ade\right) = 0$ and thus the fattened structure is not strongly connected.

Let variable a have cardinality 4 and the remaining variables have cardinality 2. Let the marginal $P(abc)$ be as follows:

$$P(a) = \text{Uniform}(a), P(b|ac) = P(b|a), P(c|ab) = P(c|a)$$

and the conditional probabilities are

a	b	P(b — a)
0	0	1
0	1	0
1	0	1
1	1	0
2	0	0
2	1	1
3	0	0
3	1	1

a	c	P(c — a)
0	0	1
0	1	0
1	0	0
1	1	1
2	0	1
2	1	0
3	0	0
3	1	1

Let the conditional probability $P(d \mid bc)$ be a mixture of a deterministic distribution setting $d = (b + c) \mod 2$ and a uniform distribution, with both components having probability 0.5:

$$P(d \mid bc) = \begin{cases} 0.75, & d = (b + c) \mod 2, \\ 0.25, & d = (b + c) \mod 2, \end{cases}$$

Define the probabilities of e and f similarly:

$$P(e \mid cd) = \begin{cases} 0.75, & e = (c+d) \mod 2, \\ 0.25, & e = (c+d) \mod 2, \end{cases} \text{ and } P(f \mid de) = \begin{cases} 0.75, & f = (d+e) \mod 2, \\ 0.25, & f = (d+e) \mod 2, \end{cases}$$

One can see that knowing a fully determines the values of b and c. Therefore, $I(b, c \mid aX) = 0$ for any set X. In particular, $I(b, c \mid ade) = 0$ and thus the fattened structure is not strongly connected. On the other hand, one can check that the original junction tree with the parameters defined above is ε-strongly connected for $\varepsilon \approx 0.143$ (using logarithms with base 2). We have therefore shown that increasing the treewidth by even one variable can shift the strong connectivity arbitrarily close to zero.

A.2 Proofs

To prove our results, we will rely on the following useful properties of conditional mutual information: for disjoint sets A, B, C, D and every probability distribution $P(ABCD)$ the following holds:

- **Chain rule**:
$$I(A, BC \mid D) = I(A, B \mid D) + I(A, C \mid BD). \tag{A.1}$$
- **Nonnegativity**:
$$I(A, B \mid D) \geq 0. \tag{A.2}$$
- **Monotonicity**: for every $A' \subseteq A, B' \subseteq B$ it holds that
$$I(A', B' \mid D) \leq I(A, B \mid D). \tag{A.3}$$

Also, **monotonicity** holds for **conditional independence:** for every $A' \subseteq A, B' \subseteq B$ it holds that $(A \perp B \mid D) \Rightarrow (A' \perp B' \mid D)$.

Proof of Lemma 7 (page 22). **Factorization ⇒ projection.** The proof is by induction.

Base case: the junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a single clique: $\mathbb{T} = \emptyset, \mathbb{S} = \emptyset, \mathbb{C} = \{C\}$. Then $X = C$ and the lemma statement is true: $P(X) = P(C)$.

Induction step: For a junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, consider a separator $S \in \mathbb{S}$. Let $C_1, \ldots, C_{d_S}$ be the cliques connected directly to S. Consider a junction tree $(\mathbb{T}_{S \to C_i}, \mathbb{C}_{S \to C_i}, \mathbb{S}_{S \to C_i})$ that is a restriction of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ to cliques and separators reachable from C_i without using the edge $(C_i - S)$. It holds that $(\mathbb{T}_{S \to C_i}, \mathbb{C}_{S \to C_i}, \mathbb{S}_{S \to C_i})$ covers exactly the variables $SX_{S \to C_i}$. There are d_S such trees for separator S. From the monotonicity of conditional independence, $P(SX_{S \to C_i})$ factors over $(\mathbb{T}_{S \to C_i}, \mathbb{C}_{S \to C_i}, \mathbb{S}_{S \to C_i})$. The induction assumption is that the distribution $P(SX_{S \to C_i})$ is equal to its projection on $(\mathbb{T}_{S \to C_i}, \mathbb{C}_{S \to C_i}, \mathbb{S}_{S \to C_i})$:

$$P(SX_{S \to C_i}) = \frac{\prod_{C' \in \mathbb{C}_{S \to C_i}} P(C')}{\prod_{S' \in \mathbb{S}_{S \to C_i}} P(S')^{d_{S'}^{S \to C_i} - 1}} = \frac{\prod_{C' \in \mathbb{C}_{S \to C_i}} P(C')}{\prod_{S' \in \mathbb{S}_{S \to C_i}} P(S')^{d_{S'} - 1}}, \tag{A.4}$$

where $d_{S'}^{S \to C_i}$ is the degree of S' in $(\mathbb{T}_{S \to C_i}, \mathbb{C}_{S \to C_i}, \mathbb{S}_{S \to C_i})$ and $d_{S'}$ is the degree of S' in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Because all the edges from $\mathbb{T}$ incident to $S' \neq S$ are retained in $\mathbb{T}'$, it holds that $d_{S'}^{S \to C_i} = d_{S'}$.

Let us now prove the induction step. Observe that $X_{C_i \to S} \equiv \cup_{j=1 \ldots d_S, j \neq i} X_{S \to C_j}$. From the definition of factorization, for every $i = 1 \ldots (d_S - 1)$, it holds that

$$\begin{aligned} (X_{S \to C_i} \perp X_{C_i \to S} \mid S) &\Leftrightarrow (X_{S \to C_i} \perp \cup_{j=1 \ldots d_S, j \neq i} X_{S \to C_j} \mid S) \\ &\Rightarrow (X_{S \to C_i} \perp \cup_{j=(i+1) \ldots d_S} X_{S \to C_j} \mid S), \end{aligned} \tag{A.5}$$

Let us adopt the convention that fraction $\frac{0}{0} = 0$, instead of being undefined. Then, using the conditional independencies $(X_{S \to C_i} \perp X_{C_i \to S} \mid S)$, we can write the probability $P(X)$ as

$$\begin{aligned} P(X) &= P(X_{S \to C_1} \mid S X_{\text{-}S X_{S \to C_1}}) P(X_{\text{-}S X_{S \to C_1}} \mid S) P(S) \\ &\quad \left\{ \text{observe that } X_{\text{-}S X_{S \to C_1}} = \cup_{j=2 \ldots d_S} X_{S \to C_j} \right\} \\ &= P\left(X_{S \to C_1} \mid S \cup \left(\cup_{j=2 \ldots d_S} X_{S \to C_j}\right)\right) P(\cup_{j=2 \ldots d_S} X_{S \to C_j} \mid S) P(S) \\ &\quad \left\{ \text{from (A.5) it holds that } P\left(X_{S \to C_1} \mid S \cup \left(\cup_{j=2 \ldots d_S} X_{S \to C_j}\right)\right) = P(X_{S \to C_1} \mid S) \right\} \\ &= P(X_{S \to C_1} \mid S) P(\cup_{j=2 .. d_S} X_{S \to C_j} \mid S) P(S) \\ &\quad \left\{ \text{from (A.5), } P(\cup_{j=i .. d_S} X_{S \to C_j} \mid S) = P(X_{S \to C_i} \mid S) P(\cup_{j=(i+1) .. d_S} X_{S \to C_j} \mid S) \right\} \\ &= P(S) \prod_{i=1}^{d_S} P(X_{S \to C_i} \mid S) = P(S) \prod_{i=1}^{d_S} \frac{P(S X_{S \to C_i})}{P(S)} = \frac{\prod_{i=1}^{d_S} P(S X_{S \to C_i})}{P(S)^{d_S - 1}} \end{aligned} \tag{A.6}$$

Plugging the projections for every $P(S X_{S \to C_i})$ from the induction assumption (A.4) into (A.6), we get

$$P(X) = \frac{\prod_{C \in \mathbb{C}} P(C)}{\prod_{S \in \mathbb{S}} P(S)^{d_S - 1}}$$

and thus prove the induction step.

Projection $\Rightarrow$ factorization. From the definition of conditional independence, $(A \perp B \mid S) \Leftrightarrow P(AB \mid S) = P(A \mid S) P(B \mid S)$, it follows that

$$(A \perp B \mid S) \Leftrightarrow \exists \psi_{AS}(A, S), \psi_{BS}(B, S) \text{ s.t. } P(A, B, S) = \psi_{AS}(A, S) \psi_{BS}(B, S).$$

For any pair of a clique C and separator S directly connected to C, set

$$\psi_{S \to C}(S, X_{S \to C}) = \frac{\prod_{C' \in \mathbb{C}_{S \to C}} P(C')}{\prod_{S' \in \mathbb{S}_{S \to C}} P(S')^{d_{S'} - 1}}, \;\; \psi_{C \to S}(S, X_{C \to S}) = \frac{\prod_{C' \in \mathbb{C}_{C \to S}} P(C')}{\prod_{S' \in \mathbb{S}_{C \to S}} P(S')^{d_{S'} - 1}}$$

to get the required decomposition of $P(X)$ into two factors. □

Proof of Lemma 8 (page 22). From Lemma 7, it follows that any $P' \in \mathcal{P}(\mathbb{T}, \mathbb{C}, \mathbb{S})$ can be written down as

$$P'(X) = \frac{\prod_{C \in \mathbb{C}} P'(C)}{\prod_{S \in \mathbb{S}} P'(S)^{d_S - 1}}.$$

Write down the KL divergence from $P(X)$ to an arbitrary $P'(X) \in \mathcal{P}(\mathbb{T}, \mathbb{C}, \mathbb{S})$:

$$\begin{aligned}
KL(P||P') &= \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \log \frac{P(\boldsymbol{X})}{P'(\boldsymbol{X})} \\
&= \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \log \left(P(\boldsymbol{X}) \cdot \frac{\prod_{S \in \mathbb{S}} P'(\boldsymbol{S})^{d_S - 1}}{\prod_{C \in \mathbb{C}} P'(\boldsymbol{C})} \right) \\
&= \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \log \left(P(\boldsymbol{X}) \cdot \frac{\prod_{S \in \mathbb{S}} P(\boldsymbol{S})^{d_S - 1}}{\prod_{C \in \mathbb{C}} P(\boldsymbol{C})} \cdot \frac{\prod_{C \in \mathbb{C}} P(\boldsymbol{C})}{\prod_{S \in \mathbb{S}} P(\boldsymbol{S})^{d_S - 1}} \cdot \frac{\prod_{S \in \mathbb{S}} P'(\boldsymbol{S})^{d_S - 1}}{\prod_{C \in \mathbb{C}} P'(\boldsymbol{C})} \right) \\
&= KL(P||P_{(\mathbb{T},\mathbb{C},\mathbb{S})}) + \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \log \left(\frac{\prod_{C \in \mathbb{C}} P(\boldsymbol{C})}{\prod_{S \in \mathbb{S}} P(\boldsymbol{S})^{d_S - 1}} \cdot \frac{\prod_{S \in \mathbb{S}} P'(\boldsymbol{S})^{d_S - 1}}{\prod_{C \in \mathbb{C}} P'(\boldsymbol{C})} \right) \\
&= KL(P||P_{(\mathbb{T},\mathbb{C},\mathbb{S})}) + \sum_{C \in \mathbb{C}} \sum_{\boldsymbol{C}} P(\boldsymbol{C}) \log \frac{P(\boldsymbol{C})}{P'(\boldsymbol{C})} - \sum_{S \in \mathbb{S}} (d_S - 1) P(\boldsymbol{S}) \log \frac{P(\boldsymbol{S})}{P'(\boldsymbol{S})} \\
&= KL(P||P_{(\mathbb{T},\mathbb{C},\mathbb{S})}) + KL(P_{(\mathbb{T},\mathbb{C},\mathbb{S})}||P').
\end{aligned}$$

Observe that only the second component of the end expression depends on P'. From the fact that KL divergence is nonnegative and zero only if the two distributions are the same, we find that $KL(P||P')$ is minimized for $P' = P_{(\mathbb{T},\mathbb{C},\mathbb{S})}$. □

Proof of Lemma 9 (page 22). First, let us prove that whenever an AKU junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ exists, an equivalent Jensen and Jensen junction tree $(\mathbb{T}', \mathbb{C}')$ exists. The proof is by explicit construction. Set $\mathbb{C}' \equiv \mathbb{C}$. For every separator $S \in \mathbb{S}$ of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, let $C_{i_{S,1}}, \ldots, C_{i_{S,d_S}}$ be the set of cliques that are connected directly to S. Add to $\mathbb{T}'$ the following $d_S - 1$ edges:

$$(C_{i_{S,1}} - C_{i_{S,2}}), \ldots, (C_{i_{S,d_S - 1}} - C_{i_{S,d_S}}). \tag{A.7}$$

We need to prove that $(\mathbb{T}', \mathbb{C})$ is a tree, the running intersection property holds, and $(\mathbb{T}', \mathbb{C})$ has the same projection expression as $(\mathbb{T}, \mathbb{C}, \mathbb{S})$.

- $(\mathbb{T}', \mathbb{C})$ **is a tree.** It is known (Diestel, 2005) that a graph with n vertices is a tree iff it is connected and has $n - 1$ edges.

  $(\mathbb{T}', \mathbb{C})$ is connected: to obtain a path from C_1 to C_2, take a path from C_1 to C_2 in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ and replace every segment $C_{i_{S,k}} - S - C_{i_{S,m}}$ (the indexing here is the same as in construction of the edges in Equation A.7) with $C_{i_{S,k}} - C_{i_{S,k+1}} - \cdots - C_{i_{S,m-1}} - C_{i_{S,m}}$.

  By construction, $(\mathbb{T}', \mathbb{C})$ contains

  $$|\mathbb{T}'| = \sum_{S \in \mathbb{S}} (d_S - 1)$$

  edges. Because in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ every edge involves exactly one separator, it holds that $\sum_{S \in \mathbb{S}} d_S = |\mathbb{T}|$. Moreover, $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a tree with $\mathbb{C} \cup \mathbb{S}$ as vertices, so $|\mathbb{T}| = |\mathbb{S}| + |\mathbb{C}| - 1$, and therefore

  $$|\mathbb{T}'| = \sum_{S \in \mathbb{S}} (d_S - 1) = |\mathbb{S}| + |\mathbb{C}| - 1 - |\mathbb{S}| = |\mathbb{C}| - 1 = |\mathbb{C}'| - 1$$

  so $(\mathbb{T}', \mathbb{C})$ is indeed a tree.

Algorithm A.1: Auxiliary transformation for an Jensen and Jensen junction tree before construction of an equivalent AKU junction tree

Input: Jensen and Jensen junction tree $(\mathbb{T}', \mathbb{C})$

set $\mathbb{T}'' \equiv \mathbb{T}'$

while $\exists$ *a simple path in* $(\mathbb{T}'', \mathbb{C})$ *of the form* $C_1 - C_2 - C_3 - \cdots - C_{m-1} - C_m$ *s.t.* $(C_1 \cap C_2) = (C_{m-1} \cap C_m) = S$ *and* $C_2 \cap C_3 \neq S$ **do**

  remove $(C_1 - C_2)$ from $\mathbb{T}''$, add $(C_1 - C_m)$ to $\mathbb{T}''$

return $(\mathbb{T}'', \mathbb{C})$

Algorithm A.2: Transformation from an Jensen and Jensen junction to an equivalent AKU junction tree

Input: $(\mathbb{T}'', \mathbb{C}) =$ Alg. A.1$(\mathbb{T}', \mathbb{C})$ for a Jensen and Jensen junction tree $(\mathbb{T}', \mathbb{C})$

$\mathbb{T} = \emptyset, \mathbb{S} = \emptyset$

for *all maximal connected subsets* $C_{i_{S,1}}, \ldots, C_{i_{S,m}} \in \mathbb{C}$ *s.t.* $\exists S$ *s.t. for every edge* $(C_{i_{S,j}} - C_{i_{S,q}})$ *between them in* $\mathbb{T}''$ *it holds that* $C_{i_{S,j}} \cap C_{i_{S,q}} = S$ **do**

  add S to $\mathbb{S}$

  add $(S - C_{i_{S,1}}), \ldots, (S - C_{i_{S,m}})$ to $\mathbb{T}$

return $(\mathbb{T}, \mathbb{C}, \mathbb{S})$

- **RIP holds in** $(\mathbb{T}', \mathbb{C})$**.** For any two cliques C_1 and C_2, consider the simple path from C_1 to C_2 in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ and replace every segment $C_{i_{S,k}} - S - C_{i_{S,m}}$ (the indexing here is the same as in Equation A.7) with $C_{i_{S,k}} - C_{i_{S,k+1}} - \cdots - C_{i_{S,m-1}} - C_{i_{S,m}}$. By RIP in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, every S on that path contains $C_1 \cap C_2$. Also, by Def. 5, we have

$$C_{i_{S,k}}, C_{i_{S,k+1}}, \ldots C_{i_{S,m-1}}, C_{i_{S,m}} \supset S \supseteq (C_1 \cap C_2)\,.$$

  Thus every clique on a path between C_1 and C_2 in $(\mathbb{T}', \mathbb{C})$ contains $C_1 \cap C_2$, so every clique on the simple path also has to contain $C_1 \cap C_2$. Therefore, RIP holds in $(\mathbb{T}', \mathbb{C})$.

- **Same projection.** Write down the projection expression for $(\mathbb{T}', \mathbb{C})$:

$$P_{(\mathbb{T}', \mathbb{C})}(X) = \frac{\prod_{C \in \mathbb{C}} P(C)}{\prod_{(C_i - C_j) \in \mathbb{T}'} P(C_i \cap C_j)}$$

  Observe that every separator $S \in \mathbb{S}$ by construction has produced $d_S - 1$ edges $(C_{i_{S,1}} - C_{i_{S,2}}), \ldots,$ $(C_{i_{S,m-1}} - C_{i_{S,d_S}})$ in $\mathbb{T}'$ such that $C_{i_{S,k-1}} \cap C_{i_{S,k}} = S$. Therefore,

$$\prod_{(C_i - C_j) \in \mathbb{T}'} P(C_i \cap C_j) = \prod_{S \in \mathbb{S}} P(S)^{d_S - 1}$$

  and

$$P_{(\mathbb{T}', \mathbb{C})}(X) = \frac{\prod_{C \in \mathbb{C}} P(C)}{\prod_{(C_i - C_j) \in \mathbb{T}'} P(C_i \cap C_j)} = \frac{\prod_{C \in \mathbb{C}} P(C)}{\prod_{S \in \mathbb{S}} P(S)^{d_S - 1}} = P_{(\mathbb{T}, \mathbb{C}, \mathbb{S})}(X).$$

Now let us show that for every Jensen and Jensen junction tree $(\mathbb{T}', \mathbb{C})$ there exists an equivalent AKU junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. First, apply Alg. A.1 to $(\mathbb{T}', \mathbb{C})$. Alg. A.1, while preserving the projection expression of $(\mathbb{T}', \mathbb{C})$, will change the edges of $(\mathbb{T}', \mathbb{C})$ so that edges associated with the same separator S are next to each other in the resulting junction tree, denoted $(\mathbb{T}'', \mathbb{C})$.

As the second step, we can simply replace every subtree of $(\mathbb{T}'', \mathbb{C})$ associated with separator S with a star-shaped tree with one separator node S and the corresponding cliques connected directly to S, which is exactly what Alg. A.2 does. We claim that $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, the result of Alg. A.2, is then an AKU junction tree equivalent to $(\mathbb{T}', \mathbb{C})$. The proof consists of two parts: first, prove that $(\mathbb{T}'', \mathbb{C})$ is a junction tree with the same projection expression as $(\mathbb{T}', \mathbb{C})$, second that $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an AKU junction tree with the same projection as $(\mathbb{T}'', \mathbb{C})$.

- $(\mathbb{T}'', \mathbb{C})$ **is a junction tree with the same projection expression as** $(\mathbb{T}', \mathbb{C})$**.**
  - $(\mathbb{T}'', \mathbb{C})$ **is a tree.** The edge replacements on line 3 of Alg. A.1 preserve connectivity of the graph and the number of edges it has, so $(\mathbb{T}'', \mathbb{C})$ is a connected graph with $|\mathbb{C}| - 1$ edges and thus a tree (Diestel, 2005).
  - $(\mathbb{T}'', \mathbb{C})$ **RIP and projection.** Let $\mathbb{C}$ to be a set of cliques of a junction tree. Consider a fully connected graph over $\mathbb{C}$, called **junction graph**, and assign a weight to every edge $C_i - C_j$ of the junction graph equal to $|C_i \cap C_j|$. It is known (Jensen and Jensen, 1994) that every maximum spanning tree of a junction graph is a junction tree, and all junction trees over $\mathbb{C}$ have the same set of separators, (taking into account node multiplicity). All junction trees with the same set of cliques and the multiset of separators have the same projection. Therefore, if we show that the edge replacements on line 3 of Alg. A.1 do not change the multiset of separators, it will follow that (a) the weight of the spanning tree $(\mathbb{T}', \mathbb{C})$ is preserved and every intermediate spanning tree in Alg. A.1 is a junction tree and (b) every intermediate spanning tree in Alg. A.1, including the end result $(\mathbb{T}'', \mathbb{C})$, has the same projection expression as $(\mathbb{T}', \mathbb{C})$.

    For every edge replacement on line 3 of Alg. A.1, it holds that

    $$(C_1 \cap C_m) \subseteq (C_1 \cap C_2) = S \text{ from the RIP},$$

    but also

    $$C_1 \supset (C_1 \cap C_2) = S, \text{ and } C_m \supset (C_{m-1} \cap C_m) = S$$

    so

    $$(C_1 \cap C_m) \supseteq S \Rightarrow (C_1 \cap C_m) = S = (C_1 \cap C_2)$$

    and the new edge $(C_1 - C_m)$ is associated with the same separator as the old one $(C_1 - C_2)$. Thus the edge replacement on line 3 does not change the multiset of separators, so every intermediate graph in the Alg. A.1 is a junction tree with the same projection expression as the initial JT $(\mathbb{T}', \mathbb{C})$.

- $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ **is an AKU junction tree with the same projection as** $(\mathbb{T}'', \mathbb{C})$**.**
  - **The nodes in** $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ **correspond to unique sets.** The cliques are already unique. For the separators, observe that for every $S \in \mathbb{S}$ there is only one maximal connected subset $C_{i_{S,1}}, \ldots, C_{i_{S,m}} \in \mathbb{C}$ s.t. for every edge $(C_{i_{S,j}} - C_{i_{S,q}})$ between $C_{i_{S,1}}, \ldots, C_{i_{S,m}}$ in $\mathbb{T}''$ it holds that $C_{i_{S,j}} \cap C_{i_{S,q}} = S$. Otherwise the termination condition on line 2 of Alg. A.1 would not hold and $(\mathbb{T}'', \mathbb{C})$ would not be the result of Alg. A.1. Thus every distinct set S is only added to $\mathbb{S}$ once.
  - $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ **is a tree.** $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is connected: for every path $C_1 - C_2 - \cdots - C_m$ in $(\mathbb{T}'', \mathbb{C})$, there exists a corresponding path $C_1 - S_{12} - C_2 - \cdots - C_m$ in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$.

Let us show that $|\mathbb{T}| = |\mathbb{C}| + |\mathbb{S}| - 1$. Consider for some $S \in \mathbb{S}$ the maximal connected subset $C_{i_{S,1}}, \ldots, C_{i_{S,m}} \in \mathbb{C}$ s.t. for every edge $(C_{i_{S,j}} - C_{i_{S,q}})$ between $C_{i_{S,1}}, \ldots, C_{i_{S,m}}$ in $\mathbb{T}''$ it holds that $C_{i_{S,j}} \cap C_{i_{S,q}} = S$. For this set of cliques, Alg. A.2 adds m edges and one separator to $\mathbb{T}$ and $\mathbb{S}$. At the same time, in $(\mathbb{T}'', \mathbb{C})$ there is a subtree with $m-1$ edges over $C_{i_{S,1}}, \ldots, C_{i_{S,m}}$. Thus the total number of edges is

$$\begin{aligned}\sum_{\text{sets } C_{i_{S,1}},\ldots,C_{i_{S,m}} \text{ as in Alg. A.2}} m &= \sum_{\text{sets } C_{i_{S,1}},\ldots,C_{i_{S,m}} \text{ as in Alg. A.2}} (m-1) \\ &\quad + \sum_{\text{sets } C_{i_{S,1}},\ldots,C_{i_{S,m}} \text{ as in Alg. A.2}} 1 \\ &= |\mathbb{T}''| + |\mathbb{S}| = |\mathbb{C}| - 1 + |\mathbb{S}|\end{aligned}$$

so $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a tree.

- **RIP.** For every pair of cliques $C_1, C_2 \in \mathbb{C}$, consider the simple path between C_1 and C_2 in $(\mathbb{T}'', \mathbb{C})$: $C_1 - C_2 - \cdots - C_{m-1} - C_m$ It corresponds, to a path $\Pi \equiv C_1 - S_{12} - C_2 - \cdots - C_{m-1} - S_{m-1,m} - C_m$ in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Π is not necessarily simple. Since RIP holds for $(\mathbb{T}'', \mathbb{C})$, it also holds for path Π. A simple path in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a subpath of Π, so the RIP holds for the simple path as well.
- $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ **has same projection as** $(\mathbb{T}'', \mathbb{C})$. The sets of cliques are the same. Also, for every separator S from $(\mathbb{T}'', \mathbb{C})$, if the $C_{i_{S,1}}, \ldots, C_{i_{S,m}} \in \mathbb{C}$ is the maximal connected subset s.t. for every edge $(C_{i_{S,j}} - C_{i_{S,q}})$ between them in $\mathbb{T}''$ it holds that $C_{i_{S,j}} \cap C_{i_{S,q}} = S$, then there are exactly $m-1$ edges in $\mathbb{T}''$ that are associated with separator S. Alg. A.2 adds a separator node S of degree m to $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, so in both $(\mathbb{T}'', \mathbb{C})$ and $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ it holds that $P(S)$ has degree $m-1$ in the denominator of the projection expression.

□

Proof of Theorem 11 (page 23). First, let us prove that whenever two AKU junction trees, $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1)$ and $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$, have the same set of cliques, $\mathbb{C}_1 = \mathbb{C}_2 = \mathbb{C}$, it follows that the two junction trees are exactly the same: $\mathbb{S}_1 = \mathbb{S}_2$ and $\mathbb{T}_1 = \mathbb{T}_2$. Suppose that $\mathbb{C}_1 = \mathbb{C}_2 = \mathbb{C}$. From Lemma 9, there exist Jensen and Jensen junction trees $(\mathbb{T}'_1, \mathbb{C})$ and $(\mathbb{T}'_2, \mathbb{C})$ with the same projection expressions as, correspondingly, $(\mathbb{T}_1, \mathbb{C}, \mathbb{S}_1)$ and $(\mathbb{T}_2, \mathbb{C}, \mathbb{S}_2)$. From Jensen and Jensen (1994), we know that the projection expression of a Jensen and Jensen JT is completely determined by the set of cliques of that JT, so $(\mathbb{T}'_1, \mathbb{C})$ and $(\mathbb{T}'_2, \mathbb{C})$ have the same projection expression. Therefore,

- $(\mathbb{T}_1, \mathbb{C}, \mathbb{S}_1)$ and $(\mathbb{T}_2, \mathbb{C}, \mathbb{S}_2)$ have the same projection expression.
- $(\mathbb{T}_1, \mathbb{C}, \mathbb{S}_1)$ and $(\mathbb{T}_2, \mathbb{C}, \mathbb{S}_2)$ have the same set of separators and their cardinalities.

To prove that $(\mathbb{T}_1, \mathbb{C}, \mathbb{S}_1)$ and $(\mathbb{T}_2, \mathbb{C}, \mathbb{S}_2)$ are the same, we now only need to prove

$$\mathbb{C}_1 = \mathbb{C}_2 = \mathbb{C} \text{ and } \mathbb{S}_1 = \mathbb{S}_2 \Rightarrow \mathbb{T}_1 = \mathbb{T}_2.$$

Consider a path $C' - S - C''$ from $(\mathbb{T}_1, \mathbb{C}, \mathbb{S})$ and a simple path between C' and C'' in $(\mathbb{T}_2, \mathbb{C}, \mathbb{S})$. From RIP, we have $S \subseteq C' \cap C''$. Since $C' \neq C''$, $|C'| = |C''| = k+1$, $S \supseteq C' \cap C''$ and $|S| = k$, it holds that $S = C' \cap C''$. Every separator S' between C' and C'' in $(\mathbb{T}_2, \mathbb{C}, \mathbb{S})$, by RIP, has to contain $C' \cap C'' = S$. But $|S'| = |S| = k$, so $S' = S$ is the only separator that may exist in the path between

C' and C''. Therefore, the path $C' - S - C''$ is also present in $(\mathbb{T}_2, \mathbb{C}, \mathbb{S})$. Repeating the above reasoning for every pair of neighboring cliques, we get $\mathbb{T}_1 \subseteq \mathbb{T}_2$ and symmetrically $\mathbb{T}_2 \subseteq \mathbb{T}_1$, so $\mathbb{T}_1 = \mathbb{T}_2$ and $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1) = (\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$.

We have proved that whenever two AKU junction trees, $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1)$ and $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$, have the same set of cliques, the two junction trees are exactly the same. Therefore, whenever two AKU junction trees $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1)$ and $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$ are different, it follows that $\mathbb{C}_1 \neq \mathbb{C}_2$. We now only need to prove that whenever $\mathbb{C}_1 \neq \mathbb{C}_2$, there exists a distribution that factors according to $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1)$, but not to $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$. Take a clique $C \in \mathbb{C}_1 \setminus \mathbb{C}_2$. Without loss of generality, assume that all of the variables in C have domains of cardinality r with values $0, \ldots, r-1$, such that r is even. Take the distribution

$$P_1(X) = \text{Uniform}(X_{\text{-}C}) P_1^C(C),$$

where $P_1^C(C)$ assigns zero probability to all assignments $\boldsymbol{C}$ such that $\sum_{x \in C} \boldsymbol{x}$ is odd and uniform probability to all assignments $\boldsymbol{C}$ such that $\sum_{x \in C} \boldsymbol{x}$ is even:

$$P_1^C(\boldsymbol{C}) = \frac{2}{r^C} 1_{\{\sum_{x \in C} \boldsymbol{x} \text{ is even}\}}.$$

We will show that $P_1(X)$ factors according to $(\mathbb{T}_1, \mathbb{C}_1, \mathbb{S}_1)$, but not to $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$. Because every variable $y \in C$ has the same number of even and odd values, for any assignment $\boldsymbol{C}_{\text{-}\boldsymbol{y}}$ it holds that

$$P_1(\boldsymbol{C}_{\text{-}\boldsymbol{y}}) = \sum_{\boldsymbol{y}} P_1^C(C_{\text{-}y} = \boldsymbol{C}_{\text{-}\boldsymbol{y}}, y = \boldsymbol{y}) = \frac{2}{r^{|C|}} \sum_{\boldsymbol{y}} 1_{\{\boldsymbol{y} + \sum_{x \in C_{\text{-}y}} \boldsymbol{x} \text{ is even}\}} = \frac{2}{r^{|C|}} \cdot \frac{r}{2} = \frac{1}{r^{|C|-1}}$$

which does not depend on the value of $\boldsymbol{C}_{\text{-}\boldsymbol{y}}$, so $P_1(C_{\text{-}y}) = \text{Uniform}(C_{\text{-}y})$. Similarly, $P_1(X_{\text{-}y}) = \text{Uniform}(X_{\text{-}y})$ and the $P_1(X)$ can be written as

$$P_1(X) = \frac{P_1(C) \prod_{C' \in \mathbb{C}_1 \setminus C} \text{Uniform}(C')}{\prod_{S \in \mathbb{S}_1} \text{Uniform}(S)^{d_S - 1}}.$$

Observe that no clique or separator of $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$ contains C. Therefore, for every clique $C'' \in \mathbb{C}_2$ it holds that $P_1(C'') = \text{Uniform}(C'')$ and similarly for separators. Therefore, the projection of $P_1(X)$ on $(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)$ is

$$P_{1,(\mathbb{T}_2, \mathbb{C}_2, \mathbb{S}_2)}(X) = \frac{\prod_{C'' \in \mathbb{C}_2} P_1(C'')}{\prod_{S \in \mathbb{S}_1} P_1(S)^{d_S - 1}} = \frac{\prod_{C'' \in \mathbb{C}_2} \text{Uniform}(C'')}{\prod_{S \in \mathbb{S}_1} \text{Uniform}(S)^{d_S - 1}} = \text{Uniform}(X) \neq P_1(X),$$

qed. □

For some of the remaining proofs, for a junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ and a leaf clique $C \in \mathbb{C}$ let us denote $(T_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ to be $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ with leaf clique C removed. Formally, we have

Definition 66. Let $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ be a junction tree, and $C \in \mathbb{C}$ be a leaf clique of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Define $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ to be the following modification of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$:

- Remove C from $\mathbb{C}$ and edge $S - C$ from $\mathbb{T}$.
- If S becomes a leaf, remove S from $\mathbb{S}$, and the edge involving S from $\mathbb{T}$.

The convenient property of $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is that, because of monotonicity of conditional mutual information, whenever $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an ε-junction tree, $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is also an ε-JT:

Lemma 67. *If $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an ε-junction tree for $P(X)$, and $C \in \mathbb{C}$ is a leaf clique of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ connected to a separator $S \in \mathbb{S}$, then $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is an ε-junction tree for $P(X_{\text{-}(C \setminus S)})$.*

Proof. First, observe that $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ involves exactly the variables $X_{\text{-}(C \setminus S)}$:

- Every variable $x \notin C$ is contained in some $C' \in \mathbb{C}_{\text{-}C}$, so x is involved in $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$.
- From RIP, no variable $y \in C_{\text{-}S}$ occurs in any clique $C' \in \mathbb{C}_{\text{-}C}$, because for every C' the simple path between C and C' includes S. Thus $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ does not involve any variables from $C_{\text{-}S}$.
- Finally, every variable $z \in S$ also occurs in some clique $C' \in \mathbb{C}_{\text{-}C}$, because, from Def. 5, S in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ was connected to at least one clique $C' \neq C$, and $S \subset C'$.

Every simple path from $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ that does not involve C is also a simple path in $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$. Moreover, because C is a leaf clique, only simple paths that have C as one of the endpoints involve C. Therefore, $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is connected (and thus a tree). Every simple path from $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is also a simple path in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, so the running intersection property holds in $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$. Therefore, $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is a junction tree.

By analogy with $X_{S \to C}$, define $X'_{S' \to C'}$ to be the variables other than S' that are reachable from C' in $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ without using the edge $S' - C'$. Every path in $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is also present in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, so $X'_{S' \to C'} \subseteq X_{S' \to C'}$ Therefore, from monotonicity of conditional mutual information,

$$I\left(X'_{S' \to C'}, X'_{C' \to S'} \mid S'\right) \leq I\left(X_{S' \to C'}, X_{C' \to S'} \mid S'\right) \leq \varepsilon,$$

so $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ is indeed an ε-junction tree. □

Proof of Lemma 14 (page 23). Write down the divergence between a distribution $P(X)$ and its projection on the junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Using the projection expression from (2.2), we get

$$\begin{aligned}
KL\left(P, P_{(\mathbb{T},\mathbb{C},\mathbb{S})}\right) &\equiv \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \log \frac{P(\boldsymbol{X})}{P_{(\mathbb{T},\mathbb{C},\mathbb{S})}(\boldsymbol{X})} \\
&= \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \log P(\boldsymbol{X}) - \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \log \frac{\prod_{C \in \mathbb{C}} P(\boldsymbol{C})}{\prod_{S \in \mathbb{S}} P(\boldsymbol{S})^{d_S - 1}} \\
&= -H(X) - \sum_{\boldsymbol{X}} P(\boldsymbol{X}) \left(\sum_{C \in \mathbb{C}} \log P(\boldsymbol{C}) - (d_S - 1) \sum_{S \in \mathbb{S}} \log P(\boldsymbol{S}) \right) \\
&\left\{ \text{observe that } \textstyle\sum_{\boldsymbol{X}_{\text{-}\boldsymbol{C}}} P(\boldsymbol{X}) = \sum_{\boldsymbol{X}_{\text{-}\boldsymbol{C}}} P(\boldsymbol{C}\boldsymbol{X}_{\text{-}\boldsymbol{C}}) = P(\boldsymbol{C}) \right\} \\
&= -H(X) - \sum_{C \in \mathbb{C}} \sum_{\boldsymbol{C}} P(\boldsymbol{C}) \log P(\boldsymbol{C}) + \sum_{S \in \mathbb{S}} (d_S - 1) \sum_{\boldsymbol{S}} P(\boldsymbol{S}) \log P(\boldsymbol{S}) \\
&= -H(X) + \sum_{C \in \mathbb{C}} H(C) - \sum_{S \in \mathbb{S}} (d_S - 1) H(S) \qquad \text{(A.8)}
\end{aligned}$$

The remainder of the proof is by induction. Consider a leaf clique C' (there is always a leaf clique in a junction tree).

Base case: C' is the only clique in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$. Then $C' = X$ and from (A.8) we have

$$KL\left(P, P_{(\mathbb{T},\mathbb{C},\mathbb{S})}\right) = -H(X) + H(C') = -H(X) + H(X) = 0 \leq n\varepsilon.$$

Induction step: C' is not the only clique. Then there is exactly one separator S' directly connected to C': $C' - S' \in \mathbb{T}$. We have

$$\begin{aligned} KL\left(P, P_{(\mathbb{T},\mathbb{C},\mathbb{S})}\right) &= -H(X) + \sum_{C\in\mathbb{C}} H(C) - \sum_{S\in\mathbb{S}} (d_S - 1)H(S) \\ &= -H(X) + H(C') - H(S') + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) \end{aligned}$$

where

$$H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) = \sum_{C\in\mathbb{C}\setminus C'} H(C) - (d_{S'} - 2)H(S') - \sum_{S\in\mathbb{S}\setminus S'} (d_S - 1)H(S).$$

Let us get back to $KL\left(P, P_{(\mathbb{T},\mathbb{C},\mathbb{S})}\right)$. Denote $B \equiv C'_{\text{-}S'}$. Then we have (using underlines to mark terms that either cancel out or are transformed at the next step)

$$\begin{aligned} KL\left(P, P_{(\mathbb{T},\mathbb{C},\mathbb{S})}\right) &= \underline{-H(X) + H(C')} - H(S) + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) \\ &= -H(X_{\text{-}BS'} \mid BS') - H(S') + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) \\ &= \underline{-H(X_{\text{-}BS'} \mid BS') + H(X_{\text{-}BS'} \mid S')} \\ &\quad -H(X_{\text{-}BS'} \mid S') - H(S') + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) \\ &= I\left(X_{\text{-}BS'}, B|S'\right) - H(X_{\text{-}BS'} \mid S') - H(S') + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) \\ &\quad \{\, B \equiv X_{S'\to C'}, X_{\text{-}BS'} \equiv X_{C'\to S'}, \\ &\qquad (\mathbb{T}, \mathbb{C}, \mathbb{S}) \text{ is an } \varepsilon\text{-JT} \Rightarrow I\left(X_{\text{-}BS'}, B|S'\right) \leq \varepsilon\} \\ &\leq \varepsilon - \underline{H(X_{\text{-}BS'} \mid S') - H(S')} + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) \\ &\quad \{H(X_{\text{-}BS'} \mid S') + H(S') = H(X_{\text{-}B})\} \\ &\leq \varepsilon - H(X_{\text{-}B}) + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}). \end{aligned} \tag{A.9}$$

Now notice that $H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'})$ is exactly the entropy of a projection of $P(X_{\text{-}B})$ on $(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'})$. Therefore,

$$-H(X_{\text{-}B}) + H(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'}) = KL\left(P(X_{\text{-}B}), P_{(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'})}(X_{\text{-}B})\right)$$

By Lemma 67, $(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'})$ is an ε-junction tree. Therefore, from induction assumption,

$$KL\left(P(X_{\text{-}B}), P_{(\mathbb{T}_{\text{-}C'}, \mathbb{C}_{\text{-}C'}, \mathbb{S}_{\text{-}C'})}(X_{\text{-}B})\right) \leq |X_{\text{-}B}| \cdot \varepsilon,$$

and plugging it into (A.9) concludes the proof of the induction step. □

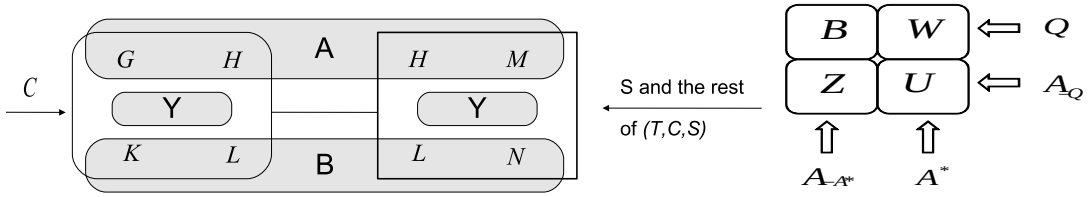
Proof of Lemma 16 (page 23). We will prove slightly more general statement, allowing the distribution P to be in certain cases approximately representable by a junction tree of treewidth higher than k:

Lemma 68. *Let $P(X)$ be a probability distribution. Let A, B, Y be a partitioning of X. Suppose there exists an ε-junction tree $(\mathbb{T}^*, \mathbb{C}^*, \mathbb{S}^*)$ for $P(X)$ s.t. for every $C^* \in \mathbb{C}^*$ it holds that $|C^* \cap AB| \leq k+1$. If $\forall W \subseteq AB$ s.t. $|W| \leq k+1$ it holds that*

$$I(A \cap W, B \cap W \mid Y) \leq \delta,$$

then

$$I(A, B \mid Y) \leq |AB|(\varepsilon + \delta).$$



(a) Illustration of notation in the proof of Lemma 16.

(b) Illustration of notation in the proof of Lemma 69.

Figure A.1: Illustration of notation in the proofs of Lemma 16 and Lemma 69.

Note that in this formulation a clique C^* of a junction tree $(\mathbb{T}^*, \mathbb{C}^*, \mathbb{S}^*)$ may have size greater than $k+1$, up to $k+1+|Y|$.

First, let us construct a new junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ from $(\mathbb{T}^*, \mathbb{C}^*, \mathbb{S}^*)$ by adding all variables of Y to every separator $S^* \in \mathbb{S}^*$ and clique $C^* \in \mathbb{C}^*$. Let us show that $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is also an ε-junction tree. Consider any neighboring pair of a separator S^* and clique C^* from $(\mathbb{T}^*, \mathbb{C}^*, \mathbb{S}^*)$. By definition of an ε-junction tree,

$$I\left(X_{C^* \to S^*}, X_{S^* \to C^*} \mid S^*\right) \leq \varepsilon.$$

We need to show for the corresponding pair S, C from $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ that $I\left(X_{C \to S}, X_{S \to C} \mid S\right) \leq \varepsilon$. Denote

$$Y_1 \equiv X_{C^* \to S^*} \cap Y, \;\; Y_2 \equiv X_{S^* \to C^*} \cap Y.$$

By construction of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, $Y_1 Y_2$ was added to S, so that $Y \subseteq S$. Because $X_{S \to C}$ and $X_{C \to S}$ do not include the variables of S, *adding* variables *to a separator* S leads to *removing* those variables *from* the sets $X_{S^* \to C^*}$. Therefore, it holds that

$$X_{C \to S} = X_{C^* \to S^*} \setminus Y_1, \;\; X_{S \to C} = X_{S^* \to C^*} \setminus Y_2, \;\; S = S^* \cup Y_1 \cup Y_2.$$

Using the chain rule, we have

$$\begin{aligned} I\left(X_{C \to S}, X_{S \to C} \mid S\right) &= I\left(X_{C \to S}, X_{S \to C} \mid S^* Y_1 Y_2\right) \\ &= I\left(X_{C \to S} Y_1, X_{S \to C} \mid S^* Y_2\right) - I\left(Y_1, X_{S \to C} \mid S^* Y_2\right) \\ &= I\left(X_{C \to S} Y_1, X_{S \to C} Y_2 \mid S^*\right) - I\left(Y_1, X_{S \to C} \mid S^* Y_2\right) \\ &\quad - I\left(Y_2, X_{C \to S} Y_1 \mid S^*\right) \\ &\leq I\left(X_{C \to S} Y_1, X_{S \to C} Y_2 \mid S^*\right) \equiv I\left(X_{C^* \to S^*}, X_{S^* \to C^*} \mid S^*\right) \leq \varepsilon, \end{aligned}$$

so $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is indeed an ε-junction tree.

Also, since we only added variables from Y to cliques and separators of $(\mathbb{T}^*, \mathbb{C}^*, \mathbb{S}^*)$, it still holds for $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ that $|C \cap AB| \leq k+1$ for every $C \in \mathbb{C}$.

Now let us prove the main statement of the lemma. The proof is by induction.

Base case. $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ contains only one clique, so $C = ABY$. Taking $W \equiv AB$, we have

$$I(A \cap AB, B \cap AB \mid Y) \leq \delta \Leftrightarrow I(A, B \mid Y) \leq \delta \Rightarrow I(A, B \mid Y) \leq |AB|(\varepsilon + \delta)$$

(the last transition is valid, because $|AB| \geq 2$), so the theorem statement is true.

Induction step. Suppose $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ contains more than one clique. Consider a leaf clique $C \in \mathbb{C}$ that is connected to only one separator S. If a tree is not empty, a leaf clique always exists. Define (see Fig. A.1a for illustration)

$$G \equiv C_{\text{-}S} \cap A, \;\; H \equiv S \cap A, \;\; M \equiv A_{\text{-}GH} \equiv X_{C \to S} \cap A$$
$$K \equiv C_{\text{-}S} \cap B, \;\; L \equiv S \cap B, \;\; N \equiv B_{\text{-}KL} \equiv X_{C \to S} \cap B.$$

On a high level, the proof shows the following three facts:

- $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$ satisfies lemma conditions, so $I\left(A_{\text{-}G}, B_{\text{-}K} \mid Y\right) \leq |A_{\text{-}G}B_{\text{-}K}|(\varepsilon + \delta)$ by induction assumption.
- The following upper bound holds:

$$I\left(A, B \mid Y\right) \leq I\left(A_{\text{-}G}, B_{\text{-}K} \mid Y\right) + I\left(A_{\text{-}G}, K \mid YB_{\text{-}K}\right) + I\left(B, G \mid YA_{\text{-}G}\right).$$

- The last two terms in the above upper bound can in turn be bounded from above by $\varepsilon + \delta$.

Let us show that $A_{\text{-}G}, B_{\text{-}K}$ and Y satisfy the conditions of this lemma. Observe that $A_{\text{-}G}B_{\text{-}K}Y = X_{\text{-}C \setminus S}$. Take $(\mathbb{T}_{\text{-}C}, \mathbb{C}_{\text{-}C}, \mathbb{S}_{\text{-}C})$, which is an ε-junction tree for $P(X_{\text{-}C \setminus S})$ by Lemma 67. Because $A_{\text{-}G}B_{\text{-}K} \subset AB$ and $\mathbb{C}_{\text{-}C} \subset \mathbb{C}$, for every $C' \in \mathbb{C}_{\text{-}C}$ it holds that

$$|A_{\text{-}G}B_{\text{-}K} \cap C'| \leq |AB \cap C'| \leq k + 1.$$

Also, for every $W \subset A_{\text{-}G}B_{\text{-}K}$ such that $|W| \leq k + 1$ it holds that $W \cap A_{\text{-}G} = W \cap A$ and $W \cap B_{\text{-}K} = W \cap B$, so

$$I\left(W \cap A_{\text{-}G}, W \cap B_{\text{-}K} \mid Y\right) = I\left(W \cap A, W \cap B \mid Y\right) \leq \delta.$$

Therefore, the conditions of this lemma hold for $A_{\text{-}G}, B_{\text{-}K}$ and Y and by induction assumption it holds that

$$I\left(HM, LN \mid Y\right) \equiv I\left(A_{\text{-}G}, B_{\text{-}K} \mid Y\right) \leq |A_{\text{-}G}B_{\text{-}K}|(\varepsilon + \delta).$$

Let us proceed to proving the induction step. Without loss of generality, assume that both G and K are not empty. Using the chain rule, we have

$$\begin{aligned}
I(A, B \mid Y) &= I(GHM, KLN \mid Y) \\
&= I(HM, KLN \mid Y) + I(G, KLN \mid YHM) \\
&= I(HM, LN \mid Y) + I(HM, K \mid YLN) + I(G, KLN \mid YHM) \\
&\leq (|AB| - |GK|)(\varepsilon + \delta) + I(HM, K \mid YLN) + I(G, KLN \mid YHM) \qquad \text{(A.10)}
\end{aligned}$$

Unroll the last term using the chain rule:

$$\begin{aligned}
I(G, KLN \mid YHM) &= I(G, MKLN \mid YH) - I(G, M \mid YH) \\
I(G, MKLN \mid YH) &= I(G, MN \mid KLYH) + I(G, KL \mid YH) \\
I(G, KL \mid YH) &= I(GH, KL \mid Y) - I(H, KL \mid Y) \\
I(G, MN \mid KLYH) &= I(GK, MN \mid LYH) - I(K, MN \mid LYH) \\
&\Downarrow \\
I(G, KLN \mid YHM) &= I(GK, MN \mid LYH) - I(K, MN \mid LYH) + \\
&\quad + I(GH, KL \mid Y) - I(H, KL \mid Y) - I(G, M \mid YH). \qquad \text{(A.11)}
\end{aligned}$$

Plugging the following mapping: $(G, K, L, H, N, M) \to (K, \emptyset, H, L, M, N)$ into (A.11), we get

$$I(HM, K \mid YLN) = I(H, KL \mid Y) - I(H, L \mid Y) + I(MN, K \mid HYL) - I(N, K \mid YL) \quad \text{(A.12)}$$

and plugging (A.11) and (A.12) back into (A.10) (again, underscores indicate terms that will be transformed on the current step of the derivation):

$$\begin{aligned}
I(A, B \mid Y) &\leq (|AB| - |GK|)(\varepsilon + \delta) + I(HM, K \mid YLN) + I(G, KLN \mid YHM) \\
&= (|AB| - |GK|)(\varepsilon + \delta) \\
&\quad \underline{+I(H, KL \mid Y)} - I(H, L \mid Y) \underline{+I(MN, K \mid HYL)} - I(N, K \mid YL) \\
&\quad \underline{+I(GK, MN \mid LYH)} \underline{-I(K, MN \mid LYH)} + \\
&\quad +I(GH, KL \mid Y) \underline{-I(H, KL \mid Y)} - I(G, M \mid YH) \\
&= (|AB| - |GK|)(\varepsilon + \delta) + I(GK, MN \mid LYH) + I(GH, KL \mid Y) \\
&\quad -I(H, L \mid Y) - I(N, K \mid YL) - I(G, M \mid YH) \\
&\leq (|AB| - |GK|)(\varepsilon + \delta) + I(GK, MN \mid LYH) + I(GH, KL \mid Y) \\
&\quad \{\text{note that } GH = A \cap C, KL = B \cap C, \text{ so } |GHKL| \leq k + 1\} \\
&\leq (|AB| - |GK|)(\varepsilon + \delta) + I(GK, MN \mid LYH) + \delta \\
&\quad \{\text{note that } YLH = S,\ GK = X_{S \to C},\ MN = X_{C \to S}\} \\
&\leq (|AB| - |GK|)(\varepsilon + \delta) + \varepsilon + \delta \\
&\quad \{\text{note that } |GK| \geq 1\} \\
&\leq |AB|(\varepsilon + \delta),
\end{aligned}$$

qed. □

Proof of Corollary 17 (page 27). Because $X_{C \to S}$, $X_{S \to C}$ and S partition the set X, from Lemma 16, for every separator S it holds that

$$I\left(X_{S \to C}, X_{C \to S} \mid S\right) \leq n(\epsilon + \delta),$$

so by definition $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an $n(\epsilon + \delta)$-junction tree for $P(X)$. □

Proof of Lemma 18 (page 29). The algorithm cycles through all subsets $W \subset X$ of size at most q on line 2. This results in $\sum_{i=2}^{q} \binom{n}{i} = O\left(n^q\right)$ iterations. For each iteration, it runs Queyranne's algorithm to find

$$\min_{U} I\left(U, W_{\text{-}U} \mid S\right).$$

Queyranne's algorithm takes $O\left(|W|^3\right) = O\left(q^3\right)$ mutual information oracle calls to find this minimum. Thus the time spent on it for every subset W is $O\left(q^3 J^{MI}_{|W|+|S|}\right) = O\left(q^3 J^{MI}_{k+q}\right)$. Finally, for every W, finding the matching components $Q \in \mathbb{Q}_S, Q \cap W \neq \emptyset$ and merging them takes at most $O(n)$ time. Therefore, the total complexity is $O\left(n^q \left(q^3 J^{MI}_{k+q} + n\right)\right)$. □

To prove Lemma 21, we need the following auxiliary result:

Lemma 69. *Suppose sets A, B, S are such that $|A| = m$, $B \subset A$, $S \cap A = \emptyset$. If for every $W \subseteq A$ s.t. $W \cap B \neq \emptyset$ and $W \cap A_{\text{-}B} \neq \emptyset$ it holds that*

$$\min_{U \subset W} I\left(U, W_{\text{-}U} \mid S\right) \leq \delta, \quad \text{(A.13)}$$

then it holds that

$$I\left(B, A_{\text{-}B} \mid S\right) \leq (m - 1)\delta.$$

Proof. The proof is by induction.

Base case: $m = 2$ trivially holds.

Induction step. Suppose the lemma statement holds for all sizes of A up to m. Let us show that it has to hold for $m + 1$ as well.

Denote

$$A^* \equiv \arg\min_{U \subset A} I\left(U, A_{\text{-}U} \mid S\right)$$

Notice that it is possible to have $A^* \subseteq B$ or $A^* \cap B = \emptyset$. Denote also (this notation is illustrated in Fig. A.1b)

$$D \equiv A^* \cap B, \;\; F \equiv A^* \cap A_{\text{-}B}, \;\; G \equiv A_{\text{-}A^*} \cap B, \;\; Z \equiv A_{\text{-}A^*} \cap A_{\text{-}B}.$$

Because $A^* \subset A$, it holds that $|DF| = |A^*| \leq m$ and $|GZ| = |A_{\text{-}A^*}| \leq m$, so by induction hypothesis we get

$$I\left(D, F \mid S\right) \leq (|A^*| - 1)\delta \text{ and } I\left(G, Z \mid S\right) \leq (|A_{\text{-}A^*}| - 1)\delta = (m - |A^*|)\delta$$

and also from (A.13)

$$I\left(DF, GZ \mid S\right) \leq \delta.$$

Observe that some of the sets D, F, G, Z may be empty, in which case the corresponding mutual information is 0: for example, if $D = \emptyset$, then $I\left(D, F \mid S\right) = 0$. The above three inequalities still hold for the case of some of D, F, G, Z being empty, and the remainder of the proof goes through as well. Let us show that

$$I\left(D, F \mid S\right) + I\left(G, Z \mid S\right) + I\left(DF, GZ \mid S\right) \geq I\left(DG, FZ \mid S\right)$$

Write down the difference in terms of entropies:

$$\begin{aligned}
& I\left(D, F \mid S\right) + I\left(G, Z \mid S\right) + I\left(DF, GZ \mid S\right) - I\left(DG, FZ \mid S\right) \\
= \;& H(D \mid S) \underline{-H(D \mid FS)} + H(G \mid S) - H(G \mid ZS) \\
& \underline{+H(DF \mid S)} - H(DF \mid GZS) - H(DG \mid S) + H(DG \mid FZS) \\
& \{\text{note that } H(DF \mid S) - H(D \mid FS) = H(F \mid S)\} \\
= \;& H(D \mid S) + H(F \mid S) + H(G \mid S) \underline{-H(G \mid ZS)} \\
& \underline{-H(DF \mid GZS)} - H(DG \mid S) + H(DG \mid FZS) \\
& \{\text{note that } H(DF \mid GZS) + H(G \mid ZS) = H(DFG \mid ZS)\} \\
= \;& H(D \mid S) + H(F \mid S) + H(G \mid S) \underline{-H(DFG \mid ZS)} - H(DG \mid S) \underline{+H(DG \mid FZS)} \\
& \{\text{note that } H(DFG \mid ZS) - H(DG \mid FZS) = H(F \mid ZS)\} \\
= \;& H(D \mid S) + H(F \mid S) \underline{+H(G \mid S)} - H(F \mid ZS) \underline{-H(DG \mid S)} \\
= \;& H(D \mid S) + H(F \mid S) - H(D|GS) - H(F \mid ZS) \\
= \;& I(D, G \mid S) + I(F, Z \mid S) \geq 0
\end{aligned}$$

Therefore,

$$\begin{aligned}
I\left(DG, FZ \mid S\right) &\leq I\left(D, F \mid S\right) + I\left(G, Z \mid S\right) + I\left(DF, GZ \mid S\right) \\
&\leq (|A^*| - 1)\delta + (m - |A^*|)\delta + \delta \\
&= m\delta = ((m + 1) - 1)\delta.
\end{aligned}$$

But $I\left(DG, FZ \mid S\right) \equiv I\left(A, A_{\text{-}B} \mid S\right)$ so the induction step is proved. □

Proof of Lemma 21 (page 30). Denote $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ the "true" ε-junction tree for $P(X)$ and consider a separator $S\ inseps$. The proof of **correctness** is by induction.

Base case. Initially, each variable $x \in X_{\text{-}S}$ forms its own partition $Q \equiv \{x\}$, so property

$$\exists C \in \mathbb{C} \text{ s.t. } (S - C) \in T \text{ and } Q \subseteq X_{S \to C} \tag{A.14}$$

holds. That is, the initial partitioning placing every variable in its own partition is correct. We will now prove that correctness is maintained throughout the execution of Alg. 2.2.

Induction step. Suppose (A.14) holds just before a subset W is tested on line 3 of Alg. 2.2. If W lies within $X_{S \to C}$ for some C, then by the induction assumption for every $Q \in \mathbb{Q}_S$ such that $Q \cap W \neq \emptyset$ it holds that $Q \subseteq X_{S \to C}$. Therefore,

$$D \equiv \left(\bigcup_{Q \in \mathbb{Q}_S \text{ s.t. } Q \cap X \neq \emptyset} Q \right) \subseteq X_{S \to C}.$$

This means that property (A.14) can only be violated after processing W if W intersects both $X_{S \to C}$ and $X_{C \to S}$. In this case, from the definition of an ε-junction tree $I\left(X_{S \to C}, X_{C \to S} \mid S\right) \leq \varepsilon$, and therefore

$$\begin{aligned}
\varepsilon &\geq I\left(X_{S \to C}, X_{C \to S} | S\right) \\
&\quad \{\text{monotonicity of conditional mutual information}\} \\
\varepsilon &\geq I\left(W \cap X_{S \to C}, W \cap X_{C \to S} \mid S\right) \\
\varepsilon &\geq \min_{U \subset W} I\left(U, W_{\text{-}U} \mid S\right) \\
&\quad \{\delta = \varepsilon\} \\
\delta &\geq \min_{U \subset W} I\left(U, W_{\text{-}U} \mid S\right),
\end{aligned}$$

which means Alg. 2.2 skips line 4 for W and the partitioning $\mathbb{Q}_S$ does not change upon processing W. Therefore, (A.14) holds right after W is processed. By induction, (A.14) holds throughout the runtime of Alg. 2.2 and in the end result.

Let us now prove $n(\varepsilon + k\delta)$**-weakness**. Consider the final partitioning $\mathbb{Q}_S$ and take an arbitrary set $Q \in \mathbb{Q}_S$. Because $\mathbb{Q}_S$ is the final result, we know that Q was not merged with any other subset $Q' \subset X_{\text{-}QS}$, which means that for every $W \subset X_{\text{-}S}$ s.t. $|W| \leq k+1$, $W \cap Q \neq \emptyset$ and $W \cap X_{\text{-}QS} \neq \emptyset$ the condition on line 3 was false, meaning

$$\min_{U \subset W} I\left(U, W_{\text{-}U} \mid S\right) \leq \delta.$$

By Lemma 69, it follows that $I\left(W \cap Q, W \cap X_{\text{-}QS} \mid S\right) \leq k\delta$ and we can apply Lemma 16 to obtain $n(\varepsilon + k\delta)$-weakness directly. □

Proof of Lemma 23 (page 32). Let us show by induction that, under the lemma conditions, each call to $GetSubtree(S, Q)$ returns an AKU junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ over variables $S \cup Q$, and a clique $C \in \mathbb{C}$.

Base case. $|Q| = 1$ and $D(Q, S) = \emptyset$. Then on line 1 Alg. 2.4 sets $x = Q$ and returns a junction tree consisting of a single clique SQ, and also returns the same SQ as a clique C.

Induction step. We need to prove the following properties of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ on line 8 of Alg. 2.4:

1. $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ involves exactly the variables $S \cup Q$. By induction assumption, for every $(S', Q') \in \mathbb{D}(S, Q)$ it holds that $GetSubtree$ on line 6 of Alg. 2.4 returns $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ that involves exactly the variables $S' \cup Q'$.

   By construction on line 7 of Alg. 2.4, $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ includes all the cliques and separators of the results $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ of Alg. 2.4 for every $(S', Q') \in \mathbb{D}(S, Q)$. The only other vertices Alg. 2.4 adds to $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ are the clique Sx on line 2 and S' for every $(S', Q') \in \mathbb{D}(S, Q)$ on line 5. Therefore, the result involves exactly the variables

$$\begin{aligned} \left(\cup_{(S',Q')\in\mathbb{D}(S,Q)}(S' \cup Q')\right) \cup Sx &= \left\{\text{by Def. 22, } S' \subset Sx\right\} = \left(\cup_{(S',Q')\in\mathbb{D}(S,Q)} Q'\right) \cup Sx \\ &= \left\{\text{by Def. 22, } \cup_{(S',Q')\in\mathbb{D}(S,Q)} Q' = Q_{\text{-}x}\right\} = QS. \quad \text{(A.15)} \end{aligned}$$

2. There are no duplicate nodes, that is no two cliques $C', C'' \in \mathbb{C}$ contain exactly the same variables, and similarly for the separators. The proof is again by induction.

   WLOG, suppose $\mathbb{D}(S, Q) = \{(S', Q'), (S'', Q'')\}$. Denote the subtrees $(C', (\mathbb{T}', \mathbb{C}', \mathbb{S}')) = \text{Alg. 2.4}(S', Q')$ and $(C'', (\mathbb{T}'', \mathbb{C}'', \mathbb{S}'')) = \text{Alg. 2.4}(S'', Q'')$. By induction assumption, the nodes in the subtrees are unique within the respective subtree. By construction, Alg. 2.4 will return a graph $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ with $\mathbb{S} = \mathbb{S}' \cup \mathbb{S}'' \cup \{S'\} \cup \{S''\}$ and $\mathbb{C} = \mathbb{C}' \cup \mathbb{C}'' \cup \{Sx\}$. Since all the separators are of size k and cliques of size $k+1$, we have $\mathbb{S} \cap \mathbb{C} = \emptyset$. Therefore, the only three ways to get a duplicate separator is to have $S' \in \mathbb{S}'$, $S'' \in \mathbb{S}'$, or $\mathbb{S}' \cap \mathbb{S}'' \neq \emptyset$. Analogously, the only two ways to get a duplicate clique is to have $Sx \in \mathbb{C}'$ or $\mathbb{C}' \cap \mathbb{C}'' = \emptyset$. Let us show that none of these five situations is possible.

   - $S' \in \mathbb{S}'$. Alg. 2.4 only adds separators that are present in the components of the decomposition. Therefore, $S' \in \mathbb{S}'$ would mean that separator S' was added to the subtree of (S', Q') at some level of recursion, which implies that there exists $(S', Q''') \in \mathbb{L}$ such that $Q''' \subset Q', |Q'''| > 0$. However, conditions of the lemma state that such (S', Q''') does not exist in $\mathbb{L}$, a contradiction.

   - $S'' \in \mathbb{S}'$. We have just shown that $S' \notin \mathbb{S}'$, the only remaining option is $S'' \neq S'$, $S'' \in \mathbb{S}'$. From (A.15), we know that $\cup_{S'''\in\mathbb{S}'} S''' \subseteq S'Q'$. Also, from Def. 22 we have

$$S'' \subset Sx, Sx \cap Q' = \emptyset \Rightarrow S'' \cap Q' = \emptyset \Rightarrow S'' \subseteq S'.$$

     We assumed $S'' \neq S'$, so $S'' \subset S'$, but from conditions of the lemma it holds that $|S''| = |S'| = k$, a contradiction.

   - $\mathbb{S}' \cap \mathbb{S}'' \neq \emptyset$ or $\mathbb{C}' \cap \mathbb{C}'' = \emptyset$. From (A.15), we know that the subtrees are exactly over the variables of (S', Q') and (S'', Q'') correspondingly. Let us look at the common variables of the two subtrees:

$$\begin{aligned} (S' \cup Q') \cap (S'' \cup Q'') &= (S' \cap S'') \cup (S' \cap Q'') \cup (S'' \cap Q') \cup (Q' \cap Q'') \\ &\quad \{Q' \cap Q'' = \emptyset \text{ by Def. 22}\} \\ &= (S' \cap S'') \cup (S' \cap Q'') \cup (S'' \cap Q') \\ &\quad \{S' \subset Sx, Q'' \cap Sx = \emptyset \text{ by Def. 22}\} \\ &= S' \cap S'' \end{aligned}$$

     From conditions of the lemma, $|S'| = |S''| = k$, and all the cliques in $\mathbb{C}'$ and $\mathbb{C}''$ are of form $S'''y$ for some $(S''', Q''') \in \mathbb{L}$, so all the subtree cliques are of size $k+1$. Because $|S' \cap S''| \leq k$, a common clique is impossible, and the only possible common separator is S', provided $S' = S''$. However, we have already shown that S' cannot be contained in $\mathbb{S}'$.

- $Sx \in \mathbb{C}'$. From (A.15), we know that $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ involves exactly the variables $S' \cup Q'$. From Def. 22 it holds that $Q' \cap Sx = \emptyset$ and $S' \subset Sx$, so $Sx \not\subseteq S' \cup Q'$, a contradiction with the assumption $Sx \in \mathbb{C}' \Rightarrow Sx \subseteq S' \cup Q'$.

We have shown that all the options for a duplicate node to appear are impossible, so the nodes of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ are unique.

3. $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a tree. A graph is a tree iff it is connected and has one more node than it has edges. The number of edges in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is (all line numbers refer to Alg. 2.4)

Edges of components obtained on line 6.	$\sum_{(S',Q') \in \mathbb{D}(S,Q)} (\lvert\mathbb{C}'\rvert + \lvert\mathbb{S}'\rvert - 1)$
Edges of form $S' - C'$ (line 7)	$\lvert\mathbb{D}\rvert$ (one for every $(S', Q') \in \mathbb{D}(S, Q)$)
Edges of form $Sx - S'$ (line 5)	m (number of unique S' in $\mathbb{D}(S, Q)$)

The total number of edges $\mathbb{T}$ is thus

$$|\mathbb{T}| = \sum_{(S',Q') \in \mathbb{D}(S,Q)} \big(|\mathbb{C}'| + |\mathbb{S}'| - 1\big) + |\mathbb{D}| + m = \sum_{(S',Q') \in \mathbb{D}(S,Q)} \big(|\mathbb{C}'| + |\mathbb{S}'|\big) + m$$

Let us now count the nodes of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$:

Nodes of components obtained on line 6.	$\sum_{(S',Q') \in \mathbb{D}(S,Q)} (\lvert\mathbb{C}'\rvert + \lvert\mathbb{S}'\rvert)$
Nodes S' (line 5)	m
Node Sx (line 2)	1

Therefore,

$$|\mathbb{S}| + |\mathbb{C}| = \sum_{(S',Q') \in \mathbb{D}(S,Q)} \big(|\mathbb{C}'| + |\mathbb{S}'|\big) + m + 1 = |\mathbb{T}| + 1.$$

Finally, $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is connected:, the subcomponents $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ are connected by induction assumption, and any two subcomponents $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ and $(\mathbb{T}'', \mathbb{C}'', \mathbb{S}'')$ are connected via edges $S' - Sx - S''$. Therefore, $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a tree.

4. Running intersection property holds.

It is sufficient to show for every S^*, C_1, C_2 s.t. $(S^* - C_1) \in \mathbb{T}$ and $(S^* - C_2) \in \mathbb{T}$ that

$$X_{S^* \to C_1} \cap X_{S^* \to C_2} = \emptyset. \tag{A.16}$$

The induction assumption is that RIP holds for the subtrees $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ and $(\mathbb{T}'', \mathbb{C}'', \mathbb{S}'')$ for (S', Q'), $(S'', Q'') \in \mathbb{D}(S, Q)$. Then three cases are possible:

- $C_1 = C', C_2 = C'', S' = S'' = S^*$. Because $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ involves exactly the variables $S'Q'$, we have

  $$X_{S^* \to C_1} = \big(S'Q'\big) \setminus S' = Q'$$

  and analogously $X_{S^* \to C_2} = Q''$. From Def. 22, it holds that

  $$Q' \cap Q'' = \emptyset \Rightarrow X_{S' \to C'} \cap X_{S'' \to C''} = \emptyset.$$

- $C_1 = C', C_2 = Sx, S^* = S'$. In this case,

  $$X_{S^* \to C_2} = \big(Sx \cup \cup_{(S''',Q''') \in \mathbb{D}(S,Q) \text{ s.t. } S''' \neq S'} \big(S''' \cup Q'''\big)\big) \setminus S'$$

  From Def. 22, it holds that $S''' \subset Sx$ and $Sx \cap Q' = \emptyset$. Also, it holds that $Q''' \cap Q' = \emptyset$, so $X_{S' \to C''} \cap Q' = \emptyset$. Because $X_{S' \to C'} = Q'$, it holds that $X_{S' \to C'} \cap X_{S' \to C''} = \emptyset$.

- $C_1, C_2 \in \mathbb{C}', S^* \in \mathbb{S}', S^* \neq S'$. In other words, we need to show that adding tree components on a higher level of recursion does not invalidate RIP for separators already added on lower levels of recursion. Suppose the new clique Sx is on the same side of S^* as C_1. Then $X^{(\mathbb{T},\mathbb{C},\mathbb{S})}_{S^*\to C_2} = X^{(\mathbb{T}',\mathbb{C}',\mathbb{S}')}_{S^*\to C_2} \subset ((S'Q') \setminus S^*)$ and

$$X^{(\mathbb{T},\mathbb{C},\mathbb{S})}_{S^*\to C_1} = X^{(\mathbb{T}',\mathbb{C}',\mathbb{S}')}_{S^*\to C_1} \cup \left(Sx \cup \bigcup_{(S''',Q''')\in\mathbb{D}(S,Q)\setminus(S',Q')} \left(S''' \cup Q'''\right) \right) \setminus S^*.$$

  Again, from Def. 22, $S''' \subset Sx$, so removing S''' from the above expression does not change the result and we can rewrite it as

$$X^{(\mathbb{T},\mathbb{C},\mathbb{S})}_{S^*\to C_1} = X^{(\mathbb{T}',\mathbb{C}',\mathbb{S}')}_{S^*\to C_1} \cup \left(S' \cup (Sx \setminus S') \cup \bigcup_{(S''',Q''')\in\mathbb{D}(S,Q)\setminus(S',Q')} Q''' \right) \setminus S^*.$$

  Observe that $S' \setminus S^*$ was already in $X^{(\mathbb{T}',\mathbb{C}',\mathbb{S}')}_{S^*\to C_1}$, so adding $S' \setminus S^*$ does not change anything. Also, $Q' \subset Q_{\text{-}x}$ and $Q \cap S = \emptyset$, so $(Sx \setminus S') \cap (S'Q') = \emptyset$. Finally, $Q''' \subset Q_{\text{-}x}, Q''' \cap Q' = \emptyset$ and $S' \subset Sx$, so $Q''' \cap (S'Q') = \emptyset$. We have shown that the variables added to $X_{S^*\to C_1}$ on the current step are absent from $S'Q' \supset X_{S^*\to C_2}$, so the RIP w.r.t. separator S^* is preserved.

The proof of the main statement of the lemma is obtained by taking int account that each call to GetSubtree(S, Q) returns an AKU junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ over variables $S \cup Q$ and applying the same reasoning as in the proof that Alg. 2.4 returns a junction tree to lines 4–8 of Alg. 2.3. □

Proof of Lemma 24 (page 34). First, consider the complexity of $FindDecompositionGreedy$ (Alg. 2.5). Line 5 cycles through $|Q|$ different values of variable x. For every x, there are $\binom{k+1}{k} = k + 1$ separators S' s.t. $S' \subseteq Sx$. For every such S', it takes $O(n)$ time to check for all components (S', Q') whether they satisfy the condition on line 8, and add them to the decomposition $\mathbb{D}(S, Q)$ if they do. Thus the complexity of $FindDecompositionGreedy$ is $O(|Q|(k + 1)n) = O(|Q|nk)$.

Consider now the complexity of $FindConsistentTreeDPGreedy$ (Alg. 2.3). There are $\binom{n}{k}$ separators and for each separator S, the components (S, Q) involve exactly $n - k$ variables:

$$\sum_{Q:(S,Q)\in\mathbb{L}} |Q| = n - k.$$

Therefore, for every S, line 2 takes time

$$\sum_{Q:(S,Q)\in\mathbb{L}} O(|Q|nk) = O(n^2k).$$

Therefore, the loop on lines 1–2 takes time $O(n^k \times n^2k) = O(n^{k+2}k)$.

For every S, the check of line 3 can be done in $O(n)$ time (since there are at most $n - k$ sets $Q \in \mathbb{Q}_S$), so the total complexity of line 3 is $O(n^{k+1})$.

The recursive unrolling of the junction tree on lines 4–8 takes $O(n)$ time because it involves n components (S', Q').

Therefore, the total complexity of Alg. 2.3 is dominated by the loop on lines 1–2 and is equal to $O(n^{k+2}k)$. □

Proof of Proposition 25 (page 34). Alg. 2.6 runs LTCI (Alg. 2.2) for every candidate separator S of size k, for a total of $O(n^k)$ invocations. The complexity of every LTCI invocation, from Lemma 18 is

$$O\left(n^q\left(q^3 J^{MI}_{k+q}+n\right)\right).$$

In Alg. 2.6, $q=k+2$, therefore

$$O\left(n^q\left(q^3 J^{MI}_{k+q}+n\right)\right)=O\left(n^{k+2}\left((k+2)^3 J^{MI}_{2k+2}+n\right)\right)=O\left(n^{k+2}\left(k^3 J^{MI}_{2k+2}+n\right)\right).$$

Since there are $O(n^k)$ invocations, the total time complexity of lines 1–3 of Alg. 2.6 is $O\left(n^{2k+2}\left(k^3 J^{MI}_{2k+2}+n\right)\right)$. The complexity of the following dynamic programming is $O(n^{k+2}k)$, so the total complexity is

$$O\left(n^{2k+2}\left(k^3 J^{MI}_{2k+2}+n\right)\right)+O(n^{k+2}k)=O\left(n^{2k+2}\left(k^3 J^{MI}_{2k+2}+n\right)\right).$$

□

Let us now prove two lemmas that will be useful in the proof of Thm. 27.

Lemma 70. *For non-intersecting sets A,B,S,Y,W, and for every probability distribution $P(ABSYW)$ it holds that*

$$I\left(A,BY\mid SW\right)+I\left(AB,W\mid SY\right)\geq I\left(A,B\mid SY\right)$$

Proof. Write down the difference:

$$\begin{aligned}
& I\left(A,BY\mid SW\right)+I\left(AB,W\mid SY\right)-I\left(A,B\mid SY\right)\\
= \;& H(ASW)-H(SW)-H(ABSWY)+H(BSWY)+\\
& +H(ABSY)-\underline{H(SY)}-H(ABSWY)+H(SWY)-\\
& -H(ASY)+\underline{H(SY)}+H(ABSY)-H(BSY)\\
= \;& \underline{H(ASW)}-H(SW)-H(ABSWY)+H(BSWY)+H(ABSY)\\
& -H(ABSWY)+\underline{H(SWY)}-H(ASY)+H(ABSY)-H(BSY)\\
& \{\text{entropy is submodular, so } H(ASW)+H(SWY)\geq H(ASWY)+H(SW)\}\\
\geq \;& \underline{H(ASWY)}+\underline{H(SW)-H(SW)}-H(ABSWY)+H(BSWY)\\
& +\underline{H(ABSY)}-H(ABSWY)-H(ASY)+H(ABSY)-H(BSY)\\
& \{\text{entropy is submodular, so } H(ASWY)+H(ABSY)\geq H(ABSWY)+H(ASY)\}\\
\geq \;& \underline{H(ABSWY)+H(ASY)-H(ABSWY)}+H(BSWY)\\
& -H(ABSWY)\underline{-H(ASY)}+H(ABSY)-H(BSY)\\
= \;& H(BSWY)-H(ABSWY)+H(ABSY)-H(BSY)=I(A,W\mid BSY)\geq 0.
\end{aligned}$$

□

Lemma 71. *Let $(\mathbb{T},\mathbb{C},\mathbb{S})$ be a maximal AKU junction tree. For every separator $S\in\mathbb{S}$ and clique $C\in\mathbb{C}$ such that $(S-C)\in\mathbb{T}$, and every set $Q\subset X$ such that*

$$X_{S\to C}\cap Q\neq\emptyset \text{ and } X_{S\to C}\not\subseteq Q$$

there exists a clique $C'\in\mathbb{C}_{S\to C}$ such that

$$C'_{\text{-}S}\cap Q\neq\emptyset \text{ and } C'_{\text{-}S}\not\subseteq Q. \tag{A.17}$$

Proof. Consider $(\mathbb{T}_{S\to C}, \mathbb{C}_{S\to C}, \mathbb{S}_{S\to C})$, the restriction of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ to $X_{S\to C}$ (notice that $S \notin \mathbb{S}_{S\to C}$). It is straightforward to check that $(\mathbb{T}_{S\to C}, \mathbb{C}_{S\to C}, \mathbb{S}_{S\to C})$ is a junction tree over $X_{S\to C}$.

The proof is by contradiction. Assume that for every clique $C' \in \mathbb{C}_{S\to C}$ it holds that

$$\text{either (a) } C'_{\text{-}S} \subseteq Q \text{ or (b) } C'_{\text{-}S} \subseteq X_{\text{-}Q}. \tag{A.18}$$

Because $X_{S\to C}$, by conditions of the lemma, contains both variables from Q and from $X_{\text{-}Q}$, it holds that cliques of both type (a) and type (b) are present in $(\mathbb{T}_{S\to C}, \mathbb{C}_{S\to C}, \mathbb{S}_{S\to C})$. Because $(\mathbb{T}_{S\to C}, \mathbb{C}_{S\to C}, \mathbb{S}_{S\to C})$ is connected, there has to be a pair of cliques $C^1, C^2 \in \mathbb{C}_{S\to C}$ that share a separator S_{12} and such that $C^1{}_{\text{-}S} \subseteq Q$ and $C^2{}_{\text{-}S} \cap Q = \emptyset$. For maximal AKU junction trees it holds that $C_1 \cap C_2 = S_{12}$ and therefore

$$S_{12} \setminus S = (C_1 \cap C_2) \setminus S = C^1{}_{\text{-}S} \cap C^2{}_{\text{-}S} \subseteq Q \cap X_{\text{-}Q} = \emptyset$$

But because $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is maximal, $|S_{12}| = |S| = k$, so it holds that $S_{12} = S$, a contradiction with the fact $S \notin \mathbb{S}_{S\to C}$. Therefore, the assumption that for every clique $C' \in \mathbb{C}_{S\to C}$ either $C'_{\text{-}S} \subseteq Q$ or $C'_{\text{-}S} \subseteq X_{\text{-}Q}$ is incorrect, which proves the statement of the lemma. □

Proof of Theorem 27 (page 36). Although Theorem 27 is (almost) a special case of Theorem 28 and has a similar proof, we present the proof of Theorem 27 separately, because dealing with several technical issues can be avoided here and the general idea behind the result is better exposed.

From the $n(\varepsilon + k\delta)$-weakness property of Lemma 21, setting $\delta = \varepsilon$, we get that for every component $(S, Q) \in \mathbb{L}$ it holds that

$$I\left(Q, X_{\text{-}QS} \mid S\right) \leq n(k+1)\varepsilon.$$

Therefore, if Alg. 2.6 returns a junction tree, it will be a $n(k+1)\varepsilon$-junction tree.

Let $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ be the junction tree that satisfies the assumptions of Thm. 27. To show that Alg. 2.6 is guaranteed to return a tree, it is sufficient to show that for every pair of a separator $S \in \mathbb{S}$ and clique $C \in \mathbb{C}$ that are directly connected in $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ it holds that

1. LTCI puts $(S, X_{S\to C})$ in $\mathbb{L}$, and
2. *FindDecompositionGreedy* will find a decomposition for $(S, X_{S\to C})$.

Let us prove **part 1**. $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is an ε-junction tree, so from the correctness property of Lemma 21, we get that for every Q s.t. $(S, Q) \in \mathbb{L}$ there exists a clique $C' \in \mathbb{C}$ such that $Q \subseteq X_{S\to C'}$. Without loss of generality, assume that $C' = C$. The rest of the proof is by contradiction. Suppose $Q \neq X_{S\to C}$. From Lemma 71, there exists $C'' \in \mathbb{C}_{S\to C}$ such that

$$C''{}_{\text{-}S} \cap Q \neq \emptyset \text{ and } C''{}_{\text{-}S} \cap X_{\text{-}Q} \neq \emptyset. \tag{A.19}$$

However, $(k+2)\varepsilon$-strong connectivity of $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ tells us that

$$\min_{U \subset C''{}_{\text{-}S}} I\left(U, C''{}_{\text{-}SU} \mid S\right) > (k+2)\varepsilon \geq \varepsilon,$$

so Alg. 2.2 had to merge all variables of $C''{}_{\text{-}S}$ into a single component, and thus

$$\text{either } C''{}_{\text{-}S} \subseteq Q \text{ or } C''{}_{\text{-}S} \subseteq X_{\text{-}Q},$$

a contradiction with (A.19). Thus $Q = X_{S\to C}$.

Now let us prove **part 2** of the sufficient conditions. The proof is by induction on the size $|Q|$ of components (S, Q). Consider separators $S_1, S_2 \in \mathbb{S}$ and cliques $C_1, C_2 \in \mathbb{C}$ from $(\mathbb{T}, \mathbb{C})$ such that the edges $S_1 - C_1 - S_2 - C_2$ are in $\mathbb{T}$. Notice that

$$X_{S_1 \to C_1} \supset X_{S_2 \to C_2} \Rightarrow |X_{S_1 \to C_1}| > |X_{S_2 \to C_2}|.$$

Induction base. If C_2 is a leaf clique, then $|X_{S_2 \to C_2}| = 1$ and the empty decomposition is trivially found. The induction base is thus proved.

Induction step. Suppose a decomposition for $(S_2, X_{S_2 \to C_2})$ has been found. When $FindDecompositionGreedy$ (Alg. 2.5) checks for a decomposition of $(S_1, X_{S_1 \to C_1})$, it will iterate through all variables x of $X_{S_1 \to C_1}$ (see line 5). Therefore, at some point it will try $x = C_1 \setminus S_1$ and form a corresponding candidate clique $S_1 \cup x = C_1$. In the corresponding inner loop (lines 7–9), $FindDecompositionGreedy$ will iterate through the components (S', Q') "compatible" with clique C_1 that can be added to the decomposition of $(S_1, X_{S_1 \to C_1})$. Component $(S_2, X_{S_2 \to C_2})$ is one such component that is also present in the "true" decomposition. To prove the induction step, it is sufficient to show that one of the following happens:

- $FindDecompositionGreedy$ adds $(S_2, X_{S_2 \to C_2})$ to $\mathbb{D}(S_1, X_{S_1 \to C_1})$, or
- $\mathbb{D}(S_1, X_{S_1 \to C_1})$ already contains another component (S', Q') such that $X_{S_2 \to C_2} \subseteq Q'$. Intuitively, it means that all the variables of $X_{S_2 \to C_2}$ are already covered in the decomposition, so the component $(S_2, X_{S_2 \to C_2})$ is not needed.

Suppose Alg. 2.5 does not add $(S_2, X_{S_2 \to C_2})$ to $\mathbb{D}(S_1, X_{S_1 \to C_1})$. This means that $(S_2, X_{S_2 \to C_2})$ fails the condition on line 8. Since $S_2 \subset C_1$, this means that

$$X_{S_2 \to C_2} \not\subseteq X_{S_1 \to C_1} \setminus \left(x \cup \bigcup_{(S', Q') \in \mathbb{D}(X_{S_1 \to C_1})} Q' \right),$$

and consequently that

$$\exists (S', Q') \in \mathbb{D}(X_{S_1 \to C_1}) \text{ such that } Q' \cap X_{S_2 \to C_2} \neq \emptyset.$$

We will now show that $X_{S_2 \to C_2} \subseteq Q'$. Suppose this is not the case. Then from Lemma 71, there exists a clique $C' \in \mathbb{C}_{S_2 \to C_2}$ such that

$$A \equiv \left(C' \text{-}_{S_2} \cap Q' \right) \neq \emptyset \text{ and } B \equiv \left(C' \text{-}_{S_2} \cap X_{\text{-}Q'} \right) \neq \emptyset \tag{A.20}$$

Using the fact that $|C_1| = k + 1$, $|S_1| = |S_2| = |S'| = k$ and $S_1, S_2, S' \subset C_1$, denote

$$S_1 = Syz, S_2 = Sxy, S' = Sxz.$$

A and B are in different components of the partitioning $\mathbb{Q}_{S'}$ for separator S'. Without loss of generality, assume that A and y are in different components for separator S' (otherwise B and y are in different components). From Lemma 70, we have

$$\begin{aligned} I(A, By \mid Sxz) &\geq I(A, B \mid Sxy) - I(AB, z \mid Sxy) \\ &> (k+2)\varepsilon - I(AB, z \mid Sxy) \\ &\quad \{S_2 = Sxy \text{ separates } z \text{ and } AB \text{ in } (\mathbb{T}, \mathbb{C}, \mathbb{S}), \\ &\quad \text{and } (\mathbb{T}, \mathbb{C}, \mathbb{S}) \text{ is an } \varepsilon\text{-JT}\} \\ &\geq (k+2)\varepsilon - \varepsilon = (k+1)\varepsilon \end{aligned} \tag{A.21}$$

But Alg. 2.2 did not put A and By in the same partition. Because $|ABy| \leq k+2$ and Alg. 2.2 was called with max-set parameter $q = k+2$ and $\delta = \varepsilon$, from Lemma 69 we get

$$I\left(A, By \mid Sxz\right) \leq (k+1)\varepsilon,$$

a contradiction. Analogously, B and y cannot be in different components, so $ABy \subseteq Q'$, which contradicts with $B \subseteq X_{\text{-}Q'}$ that follows from the initial assumption that $X_{S_2 \to C_2} \not\subseteq Q'$.

We have proved that when $FindDecompositionGreedy$ tries to add a true component $(S_2, X_{S_2 \to C_2})$ to the decomposition of $(S_1, X_{S_1 \to C_1})$, either it succeeds, or all the variables from $X_{S_1 \to C_1}$ are already covered by another component. Therefore, eventually all variables of $X_{S_1 \to C_1} \setminus x$ will be covered, and a decomposition of $(S_1, X_{S_1 \to C_1})$ will thus be found. This holds for every separator from $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, so a junction tree will eventually be found $\square$

Proof of Theorem 28 (page 37). We will first show that whenever Alg. 2.6 returns a junction tree, it will be a $n(k+1)\varepsilon$-junction tree. Then we will show that Alg. 2.6 is guaranteed to return a junction tree.

First, starting with the given "true" maximal ε-junction tree $(\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)$ of treewidth m, let us construct a "fattened" maximal ε-junction tree $(\mathbb{T}^k, \mathbb{C}^k, \mathbb{S}^k)$ of treewidth k. Take the separator S^{m*} and cliques $C_1^m, \ldots, C_j^m$ directly connected to S^{m*} such that $\sum_{i=1}^{j} |X_{S^{m*} \to C_i^m}| = k - m$. Denote $A = \cup_{i=1}^{j} X_{S^{m*} \to C_i^m}$. The new junction tree $(\mathbb{T}^k, \mathbb{C}^k, \mathbb{S}^k)$ is obtained by adding A to every clique and separator of $(\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)$ and removing the cliques $\mathbb{C}^m_{S^{m*} \to C_i^m}$ for $i = 1, \ldots, j$ and corresponding separators:

$$\begin{aligned}
\mathbb{C}^k &= \left\{ C^m A \mid C^m \in \mathbb{C}^m \setminus \cup_{i=1}^{j} \mathbb{C}^m_{S^{m*} \to C_i^m} \right\}, \\
\mathbb{S}^k &= \left\{ S^m A \mid S^m \in \mathbb{S}^m \setminus \cup_{i=1}^{j} \mathbb{S}^m_{S^{m*} \to C_i^m} \right\}, \\
\mathbb{T}^k &= \left\{ (S^m A - C^m A) \mid (S^m - C^m) \in \mathbb{T}^m,\ C^m \notin \cup_{i=1}^{j} \mathbb{C}^m_{S^{m*} \to C_i^m} \right\}.
\end{aligned}$$

For a clique C^m or separator S^m from $(\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)$, we will denote C^k and S^k their corresponding counterpart in $(\mathbb{T}^k, \mathbb{C}^k, \mathbb{S}^k)$. For every edge $(S^k - C^k) \in \mathbb{T}^k$, we have

$$\begin{aligned}
I\left(X_{S^k \to C^k}, X_{C^k \to S^k} \mid S^k\right) &= I\left(X_{S^m \to C^m} \setminus A, X_{C^m \to S^m} \setminus A \mid S^m A\right) \\
&\leq I\left(X_{S^m \to C^m}, X_{C^m \to S^m} \mid S^m\right) \quad \text{(by the chain rule)} \\
&\leq \varepsilon \quad \text{(since } (\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m) \text{ is an } \varepsilon\text{-junction tree).}
\end{aligned}$$

Therefore, $(\mathbb{T}^k, \mathbb{C}^k, \mathbb{S}^k)$ is an ε-junction tree. From the $n(\varepsilon + k\delta)$-weakness property of Lemma 21, setting $\delta = \varepsilon$, we get that for every component $(S, Q) \in \mathbb{L}$ it holds that

$$I\left(Q, X_{\text{-}QS} \mid S\right) \leq n(k+1)\varepsilon.$$

Therefore, if Alg. 2.6 returns a junction tree, it will be a $n(k+1)\varepsilon$-junction tree.

Let us now show that Alg. 2.6 is guaranteed to find a junction tree. Let $\overrightarrow{\mathbb{T}}^k$ be the edges of $\mathbb{T}^k$ *directed away from* S^{k*}. It is then sufficient to show for every $(S^k \to C^k) \in \overrightarrow{\mathbb{T}}^k$ that

1. LTCI puts $(S^k, X_{S^k \to C^k})$ in $\mathbb{L}$, and
2. *FindDecompositionGreedy* will find a decomposition for $(S^k, X_{S^k \to C^k})$.

Notice that, unlike the proof of Theorem 27, here we are only concerned with discovering and decomposing components $X_{S^k \to C^k}$ such that C^k and the separator S^{k*} are on the different sides of separator S^k. In other words, here we will show properties 1 and 2 only for the case when the junction tree edges are oriented away from one specific separator, namely S^{k*}, whereas the proof of Theorem 27 shows that for $k = m$, 1 and 2 hold for every $S^k \in \mathbb{S}^k$ in the role of S^{k*}.

Let us prove **part 1**. For any fixed $(S^k \to C^k) \in \overrightarrow{\mathbb{T}}^k$, consider a component $(S^k, Q) \in \mathbb{L}$ such that $Q \cap X_{S^k \to C^k} \neq \emptyset$. We need to show that $Q = X_{S^k \to C^k}$.

From the correctness property of Lemma 21 applied to the "fattened" ε-junction tree $(\mathbb{T}^k, \mathbb{C}^k, \mathbb{S}^k)$, it follows that $Q \subseteq X_{S^k \to C^k}$.

Suppose $Q \neq X_{S^k \to C^k}$, then from Lemma 71, there exists $C^{k''} \in \mathbb{C}_{S^k \to C^k}$ such that

$$C^{k''}{}_{\text{-}S^k} \cap Q \neq \emptyset \text{ and } C^{k''}{}_{\text{-}S^k} \cap X_{\text{-}Q} \neq \emptyset.$$

We have

$$\begin{aligned}
\min_{U \subset C^{k''}{}_{\text{-}S^k}} I\left(U, C^{k''}{}_{\text{-}S^k U} \mid S^k\right) &\equiv \min_{U \subset (C^{m''}A)_{\text{-}S^m A}} I\left(U, (C^{m''}A)_{\text{-}S^m AU} \mid S^m A\right) \\
&= \min_{U \subset C^{m''}{}_{\text{-}S^m}} I\left(U, C^{m''}{}_{\text{-}S^m U} \mid S^m A\right) \\
&= \min_{U \subset C^{m''}{}_{\text{-}S^m}} \left(I\left(U, AC^{m''}{}_{\text{-}S^m U} \mid S^m\right) - I\left(A, U \mid S^m\right)\right) \\
&\quad \{\text{apply monotonicity of } I(\cdot, \cdot \mid \cdot)\} \\
&\geq \min_{U \subset C^{m''}{}_{\text{-}S^m}} \left(I\left(U, C^{m''}{}_{\text{-}S^m U} \mid S^m\right) - I\left(A, U \mid S^m\right)\right) \\
&\quad \{(\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m) \text{ is } (k+3)\varepsilon\text{-strongly connected}\} \\
&\geq \min_{U \subset C^{m''}{}_{\text{-}S^m}} \left((k+3)\varepsilon - I\left(A, U \mid S^m\right)\right) \\
&\quad \{S^m \text{ separates } U \text{ and } A \text{ in an } \varepsilon\text{-JT } (\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)\} \\
&\geq (k+3)\varepsilon - \varepsilon = (k+2)\varepsilon > \varepsilon = \delta.
\end{aligned}$$

Therefore, contrary to our assumption, all the variables of $C^{k''}{}_{\text{-}S^k}$ will be placed into the same component (S^k, Q'). For any fixed separator S^k, every variable belongs to exactly one component, so $Q = Q' = X_{S^k \to C^k}$ and thus $(S^k, X_{S^k \to C^k}) \in \mathbb{L}$.

Let us now prove **part 2**. The proof by induction on the component size is analogous to that of Theorem 27. The key fact we we need to show is that for every tuple $(S^k_a, C^k_a, S^k_b, C^k_b)$ such that $(S^k_a \to C^k_a \to S^k_b \to C^k_b) \in \overrightarrow{\mathbb{T}}^k$, after Alg. 2.5 selects $x = C^k_a \setminus S^k_a$ on line 5, Alg. 2.5 will include into the candidate decomposition $\mathbb{D}(S^k_a, X_{S^k_a \to C^k_a})$ either $(S^k_b, X_{S^k_b \to C^k_b})$ or some other component $(S^{k'}, Q^{k'})$ such that $Q^{k'} \supseteq X_{S^k_b \to C^k_b}$.

The only way for Alg. 2.5 to not add $(S^k_b, X_{S^k_b \to C^k_b})$ to $\mathbb{D}(S^k_a, X_{S^k_a \to C^k_a})$ is for $(S^k_b, X_{S^k_b \to C^k_b})$ to fail the condition on line 8, which will entail that there exists $(S^{k'}, Q^{k'}) \in \mathbb{D}(S^k_a, X_{S^k_a \to C^k_a})$ such that $Q^{k'} \cap X_{S^k_b \to C^k_b} \neq \emptyset$. Suppose $X_{S^k_b \to C^k_b} \not\subseteq Q'$, then from Lemma 71, there exists a clique $C^{k'} \in \mathbb{C}_{S^k_b \to C^k_b}$ such that

$$F^k \equiv \left(C^{k'}{}_{\text{-}S^k_b} \cap Q^{k'}\right) \neq \emptyset \text{ and } B^k \equiv \left(C^{k'}{}_{\text{-}S^k_b} \cap X_{\text{-}Q^{k'}}\right) \neq \emptyset$$

Because $C^{k'} = C^{m'}A$ and $S_b^k = S_b^m A$, it holds that ${C^{k'}}_{\text{-}S_b^k} = {C^{m'}}_{\text{-}S_b^m}$ and thus F^k and B^k partition ${C^{m'}}_{\text{-}S_b^m}$:

$$F^m \equiv \left({C^{m'}}_{\text{-}S_b^m} \cap Q^{k'}\right) = F^k \text{ and } B^m \equiv \left({C^{m'}}_{\text{-}S_b^m} \cap X_{\text{-}Q^{k'}}\right) = B^k.$$

Denote $F^k = F^m = F, B^k = B^m = B$. Using the fact that $|C_a^k| = k+1$, $|S_a^k| = |S_b^k| = |S^{k'}| = k$ and $S_a^k, S_b^k, S^{k'} \subset C_a^k$, denote

$$S_a^k = S^k yz, S_b^k = S^k xy, S^{k'} = S^k xz.$$

A and B are in different components of the partitioning $\mathbb{Q}_{S^{k'}}$ for separator $S^{k'}$. Without loss of generality, assume that A and y are in different components for $S^{k'}$. From Lemma 70, we have

$$\begin{aligned}
I\left(F, By \mid S^k xz\right) &\geq I\left(F, B \mid S^k xy\right) - I\left(FB, z \mid S^k xy\right) \\
&= I\left(F, B \mid S_b^m A\right) - I\left(FB, z \mid S_b^m A\right) \\
&\geq I\left(F, B \mid S_b^m A\right) - I\left(FB, zA \mid S_b^m\right) \\
&\quad \{S_b^m \text{ separates } zA \text{ and } FB \text{ in an } \varepsilon\text{-JT } (\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)\} \\
&\geq I\left(F, B \mid S_b^m A\right) - \varepsilon \\
&= I\left(F, BA \mid S_b^m\right) - I\left(F, A \mid S_b^m\right) - \varepsilon \\
&\quad \{S_b^m \text{ separates } A \text{ and } F \text{ in an } \varepsilon\text{-JT } (\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m)\} \\
&\geq I\left(F, B \mid S_b^m\right) - 2\varepsilon = I\left(F, {C^{m'}}_{\text{-}FS_b^m} \mid S_b^m\right) - 2\varepsilon \\
&\quad \{(\mathbb{T}^m, \mathbb{C}^m, \mathbb{S}^m) \text{ is } (k+3)\varepsilon\text{-strongly connected}\} \\
&> (k+3)\varepsilon - 2\varepsilon = (k+1)\varepsilon
\end{aligned}$$

But Alg. 2.2 did not put F and By in the same partition. Because $|FBy| \leq k+2$ and Alg. 2.2 was called with max-set parameter $q = k+2$ and $\delta = \varepsilon$, from Lemma 69 we get

$$I\left(F, By \mid Sxz\right) \leq (k+1)\varepsilon,$$

a contradiction. Analogously, B and y cannot be in different components, so $FBy \subseteq Q'$, which contradicts with $B \subseteq X_{\text{-}Q'}$ that follows from the initial assumption that $X_{S_b \to C_b} \not\subseteq Q'$. □

Proof of Theorem 31 (page 39). Let us compute every mutual information estimate with the accuracy Δ with probability $1 - \frac{\gamma}{O(n^{2k+2})}$ using the technique of Höffgen (1993). We will prove that in this case the theorem statement holds.

The **time complexity** follows directly from Proposition 25 and the fact that from Thm. 30 the individual mutual information computation complexity is

$$J_{2k+2}^{MI} = f(2k+2, r, \delta, \frac{\gamma}{n^{2k+2}}).$$

We now need to show the **sample complexity** and **result quality**. Observe that Alg. 2.6 makes a total of $O(n^{2k+2})$ conditional mutual information computations. From Thm. 30, every such computation is within $\pm\Delta$ of the true value with probability $1 - \frac{\gamma}{O(n^{2k+2})}$. By union bound, all of these mutual information estimates will be within $\pm\Delta$ of their true values *simultaneously* with probability at least $1 - \gamma$.

For the rest of the proof, we can concentrate on the favorable case of all estimates being Δ-accurate, because the alternative has probability less than γ. Let us prove that Alg. 2.6 will return an $n(k+1)(\varepsilon +$

2Δ)-junction tree. The proof is essentially the same as that of Thm. 27. We will only highlight the necessary differences.

Denote the value of the estimates of $I(\cdot,\cdot \mid \cdot)$ from data to be $\hat{I}(\cdot,\cdot \mid \cdot)$. Then

$$\hat{I}(A,B \mid C) \leq \delta \Rightarrow I(A,B \mid C) \leq \delta + \Delta \tag{A.22}$$

$$I(A,B \mid C) \leq \varepsilon \Rightarrow \hat{I}(A,B \mid C) \leq \varepsilon + \Delta = \delta \tag{A.23}$$

From (A.22), replacing exact conditional mutual information computations with estimates from data results in replacing the $n(\varepsilon + k\delta)$-weakness of LTCI with $n(\varepsilon + k(\delta + \Delta))$-weakness. Since $\delta = \varepsilon + \Delta$, it holds that

$$n(\varepsilon + k(\delta + \Delta)) = n(\varepsilon + k(\varepsilon + 2\Delta)) \leq n(k+1)(\varepsilon + 2\Delta),$$

so any returned junction tree will be a $n(k+1)(\varepsilon + 2\Delta)$-junction tree.

Because of (A.23), LTCI will not mistakenly merge variables on the different sides of any true separator (that is, the partitioning will be **correct**). On the other hand, from strong connectivity, for every clique $C'' \in \mathbb{C}_{S \to C}$ it holds that

$$\begin{array}{rcl} \min\limits_{U \subset C''\text{-}_S} I\left(U, C''\text{-}_{SU} \mid S\right) & > & (k+2)(\varepsilon + 2\Delta) \\ & \Downarrow & \\ \min\limits_{U \subset C''\text{-}_S} \hat{I}\left(U, C''\text{-}_{SU} \mid S\right) & > & (k+2)(\varepsilon + 2\Delta) - \Delta > (\varepsilon + \Delta) = \delta \end{array}$$

so all of the variables of $C''\text{-}_S$ will be assigned to the same set Q in the partitioning $\mathbb{Q}_S$ corresponding to the separator S. Therefore, by the same reasoning as in the proof of Thm. 27, every "true" component $(S, X_{S \to C})$ will be in $\mathbb{L}$.

Finally, to show that a decomposition will be found for every $(S, X_{S \to C})$, we need to modify the part of the reasoning in the proof of Thm. 27 starting with Eqn. A.21

$$\begin{array}{rcl} I(A, By \mid Sxz) & \geq & I(A,B \mid Sxy) - I(AB, z \mid Sxy) \\ & > & (k+2)(\varepsilon + 2\Delta) - I(AB, z \mid Sxy) \\ & \geq & (k+2)(\varepsilon + 2\Delta) - \varepsilon > (k+1)(\varepsilon + 2\Delta) \end{array}$$

But LTCI did not put A and By in the same partition. This means that for every $W \subseteq ABy$ that intersects both A and By it holds that

$$\min_{U \subset W} \hat{I}(U, W\text{-}_U \mid Sxz) \leq \varepsilon + \Delta \Rightarrow \min_{U \subset W} I(U, W\text{-}_U \mid Sxz) \leq \varepsilon + 2\Delta$$

so applying Lemma 69 with $\delta = \varepsilon + 2\Delta$ we get

$$I(A, By \mid Sxz) \leq (k+1)(\varepsilon + 2\Delta),$$

the same contradiction as in the proof of Thm. 27. The rest of the proof of Thm. 27 goes without change to yield the result of this theorem. □

Proof of Corollary 32 (page 39). Select $\Delta = \frac{\beta}{2(k+1)n^2}$. Since $2(k+2)\Delta = \frac{\beta(k+2)}{n^2(k+1)} \leq \alpha$, an α-strongly connected junction tree is also $2(k+2)\Delta$-strongly connected, so from Thm. 31 the number of *samples* sufficient for Alg. 2.6 to learn a $2(k+1)n\Delta$-junction tree with probability at least $1-\gamma$ is

$$\begin{aligned} f\left(2k+2, r, \Delta, \frac{\gamma}{n^{2k+2}}\right) &= f\left(2k+2, r, \frac{\beta}{2(k+1)n^2}, \frac{\gamma}{n^{2k+2}}\right) \\ &= O\left(r^{4k+4}\frac{4(k+1)^2n^4}{\beta^2}\log^2\frac{r^{2k+2}2(k+1)n^2}{\beta}\log\frac{r^{2k+2}n^{2k+2}}{\gamma}\right) \\ &= O\left(\frac{n^4k^2r^{4k+4}}{\beta^2}\log^2\frac{n^2kr^k}{\beta}\log\frac{nr^k}{\gamma}\right) \end{aligned}$$

and the time complexity is

$$\begin{aligned} O\left(n^{2k+2}\left(f\left(2k+2, r, \Delta, \frac{\gamma}{n^{2k+2}}\right)k^3+n\right)\right) &= \\ O\left(n^{2k+2}\left(\frac{n^4k^2r^{4k+4}}{\beta^2}\left(\log^2\frac{n^2kr^k}{\beta}\log\frac{nr^k}{\gamma}\right)k^3+n\right)\right) &= \\ O\left(\frac{n^{2k+6}k^5r^{4k+4}}{\beta^2}\log^2\frac{n^2kr^k}{\beta}\log\frac{nr^k}{\gamma}\right). \end{aligned}$$

Denote $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ the resulting $2(k+1)n\Delta$-junction tree. Because $2(k+1)n\Delta \leq \frac{\beta}{n}$, it holds that $(\mathbb{T}', \mathbb{C}', \mathbb{S}')$ is also a $\frac{\beta}{n}$-junction tree. From Lemma 14, it holds that $KL\left(P, P_{(\mathbb{T}', \mathbb{C}', \mathbb{S}')}\right) \leq \beta$. □

Proof of Lemma 33 (page 41). The **time complexity** is detailed in the table right before the lemma statement in the text. Let us show that Alg. 2.7 will return an $\hat{\varepsilon}$-junction tree with $\hat{\varepsilon} \leq n(k+1)\varepsilon$. Observe that every time the line 9 is reached, for every S it holds that $\mathbb{Q}_S$ is exactly the same as LTCI$(S, I, \delta, k+2)$ would find, because the only values of δ' at which $\mathbb{Q}_S$ may possibly change are exactly those for which $(\delta'; S) \in \mathbb{A}$, and for every such $\delta' \leq \delta$ the corresponding $\mathbb{Q}_S$ has been recomputed by Alg. 2.7. The only difficulty that can arise concerns "degenerate" values:

$$(\delta; S'), (\delta; S'') \in \mathbb{A}, S' \neq S''.$$

In this case, $\mathbb{Q}_S$ for every S will become the same as LTCI$(S, I, \delta, k+2)$ would compute from scratch after Alg. 2.7 processes the last pair $(\delta; S'')$ containing the "degenerate" value δ.

Consider

$$\delta^* = \max_{(\delta;S)\in\mathbb{W} \text{ s.t. } \delta\leq\varepsilon} \delta. \tag{A.24}$$

By construction of $\mathbb{A}$ on lines 2–4 of Alg. 2.7, for every $W \subset X_{\text{-}S}$ s.t. $|W| \leq k+2$ it holds that

$$\min_{U\subset W} I\left(U, W_{\text{-}U} \mid S\right) > \delta^* \Leftrightarrow \min_{U\subset W} I\left(U, W_{\text{-}U} \mid S\right) > \varepsilon.$$

Therefore, if Alg. 2.7 gradually increases the value δ and reaches δ^*, the outcomes of all mutual information comparisons on line 3 will be the same as for $\delta = \varepsilon$. From Thm. 27, the *FindConsistentTreeDPGreedy* will find a $n(k+1)\varepsilon$-junction tree in this case.

The only other option is that δ does not reach the value of δ^*. It can only happen if *FindConsistentTreeDPGreedy* will find a junction tree before δ reaches δ^*, that is for some $\delta < \delta^*$. In that case, from the $n(\varepsilon + k\delta)$-weakness property of Lemma 21, we have (again using the fact that $\delta \leq \varepsilon$ from Equation A.24)

$$\hat{\varepsilon} = n(\varepsilon + k\delta) \leq n(\varepsilon + k\delta^*) \leq n(k+1)\varepsilon.$$

We have shown that in both cases an $\hat{\varepsilon}$-junction tree will be found, with $\hat{\varepsilon} \leq n(k+1)\varepsilon$. □

Proof of Lemma 35 (page 43). **Time complexity.** On line 2, Alg. 2.8 iterates through all subsets W of size at most q. There are $O(n^q)$ such subsets. For every W, $\mathrm{strength}_S(W)$ is computed, taking $O(q^3 J^{MI}_{q+k})$ time. As will be shown shortly, at any time there are at most $|X_{\text{-}S}|$ hyperedges in $\mathbb{W}_S$. Note also that every hyperedge $W \in \mathbb{W}_S$ has size at most q. Thus checking any edge for redundancy w.r.t. S and $\mathbb{W}_S$ (using depth-first search) takes $O(q|X_{\text{-}S}|) = O(qn)$ time. The total complexity of Alg. 2.8 is thus

$$O\left(n^q \left(q^3 J^{MI}_{q+k} + qn^2\right)\right).$$

Let us show that at every point in time in the execution of Alg. 2.8 $\mathbb{W}_S$ contains at most $|X_{\text{-}S}|$ elements. Observe that right before an edge W is added on line 3, none of the hyperedges $W \in \mathbb{W}_S$ are redundant w.r.t. S and $\mathbb{W}_S$. Thus, if we were to start with a graph over nodes $X_{\text{-}S}$ and no edges, and enable the hyperedges from $\mathbb{W}_S$ one by one, every edge would reduce the number of connected components by at least one. The graph with no edges contains $|X_{\text{-}S}|$ connected components, each consisting of a single variable. Because there cannot be less than one, $\mathbb{W}_S$ can contain at most $|X_{\text{-}S}| - 1$ hyperedges. One more hyperedge is then added on line 3.

Alg. 2.8 returns exactly the set of non-redundant edges. Observe that at any time throughout the execution of Alg. 2.8 it holds that $\mathbb{W}_S \subseteq \mathbb{W}_S[0]$, because Alg. 2.8 only considers hyperedges W of size at most q and only adds edges with nonzero strength to $\mathbb{W}_S$.

Suppose U is a non-redundant hyperedge w.r.t. S and $\mathbb{W}_S[0]$. Then $\mathrm{strength}_S(U) > 0$ and at some point $W = U$ will be added to $\mathbb{W}_S$ on line 3. Because $\mathbb{W}_S \subseteq \mathbb{W}_S[0]$, from monotonicity (2.13) it follows that U is not redundant w.r.t. S and $\mathbb{W}_S$. Therefore, U will be retained throughout the remainder of the execution of Alg. 2.8, so all the non-redundant hyperedges will be included in the result.

Suppose now W is redundant w.r.t. S and $\mathbb{W}_S[0]$. Let $\mathbb{R}(W \mid S, \mathbb{W}_S[0]) = \{W_1, \ldots, W_m\}$ to be a redundant set of hyperedges from Def. 34 (there may be many different redundant sets for W, we can choose one of them randomly). Let us construct a set $\overline{\mathbb{R}}(W \mid S, \mathbb{W}_S[0])$ as follows:

- Initialize $\overline{\mathbb{R}}(W \mid S, \mathbb{W}_S[0]) = \mathbb{R}(W \mid S, \mathbb{W}_S[0])$.
- While there is a redundant edge W' in $\overline{\mathbb{R}}(W \mid S, \mathbb{W}_S[0])$, replace W' with its redundant set $\mathbb{R}(W \mid S, \mathbb{W}_S[0])$.

Observe that the end result of such an iterative process $\overline{\mathbb{R}}(W \mid S, \mathbb{W}_S[0])$ consists only of non-redundant edges: $\overline{\mathbb{R}}(W \mid S, \mathbb{W}_S[0]) \subseteq \mathbb{N}_S$, because any redundant edge will be eventually selected for replacement with its redundant set and thus will be absent from the eventual result. Also, because a redundant set of a hyperedge W' keeps the variables of W' connected, $\overline{\mathbb{R}}(W \mid S, \mathbb{W}_S[0])$ remains a redundant set for W throughout the iterative replacements.

We have already shown that the result $\mathbb{W}_S$ found by Alg. 2.8 includes all the non-redundant edges. It follows that at the last iteration of the loop of line 2, after the hyperedge W has been added to $\mathbb{W}_S$, it holds that all non-redundant edges are in $\mathbb{W}_S$. Therefore, for any redundant hyperedge W that is still in $\mathbb{W}_S$ it holds that $\overline{\mathbb{R}}(W \mid S, \mathbb{W}_S[0]) \subset \mathbb{W}_S$ throughout the redundancy checks on lines 4–6, so W will be found to be redundant w.r.t. S and $\mathbb{W}_S$ and eliminated. Thus, none of the redundant edges will survive. □

Proof of Lemma 36 (page 44). The **time complexity** is detailed in the table right before the lemma statement in the text.

The proof of **result quality** is the same as in Lemma 33, except that the value $\delta = \delta^*$ with

$$\delta^* = \max_{(S;W)\in\mathbb{W} \text{ s.t. } \mathrm{strength}_S(W)\leq\varepsilon} \mathrm{strength}_S(W)$$

will lead not to the same *outcomes of conditional mutual information comparisons* by LTCI$(S, I, \delta, k+2)$ and LTCI$(S, I, \varepsilon, k+2)$ for every candidate separator S, but to the same partitionings $\mathbb{Q}_S$. □

Proof of Theorem 37 (page 46). **Time complexity.** Observe that throughout the execution of Alg. 2.10 and Alg. 2.11 every $\mathbb{W}_S$ consists of hyperedges not redundant w.r.t. S and $\mathbb{W}_S$, and at most one other hyperedge, right after it was added on line 4 of Alg. 2.11. From the proof of Lemma 35, we know that it follows that $|\mathbb{W}_S| \leq |X_{\text{-}S}|$ and the modification of adding a new hyperedge to $\mathbb{W}_S$ and clearing out the redundant w.r.t. S and $\mathbb{W}_S$ hyperedges has time complexity $O\left(J^{MI}_{2k+1}k^3 + n^2k\right)$.

Let us now show that for every pair of separator S and hyperedge W it is sufficient to check W on line 3 of Alg. 2.11 and attempt to insert W into $\mathbb{W}_S$ only once throughout the execution of Alg. 2.10. Let us enumerate the possible outcomes of the first attempt to insert W into $\mathbb{W}_S$ on lines line 3–4 of Alg. 2.11:

- $\mathrm{strength}_S(W) \leq \delta$. In this case, W is not added to $\mathbb{W}_S$. Because it holds that

  $$\delta = \min_S \min_{Y\in\mathbb{W}_S} \mathrm{strength}_S(Y),$$

  and only hyperedges with strengths greater than δ can be added to $\mathbb{W}_S$ on line 4 of Alg. 2.11, it holds that δ monotonically increases throughout the execution of Alg. 2.10. Therefore, for the rest of the runtime of Alg. 2.10 it will hold that $\mathrm{strength}_S(W) \leq \delta$ and W will never be added to $\mathbb{W}_S$ and there is no need to check the pair S, W more than once.

- $\mathrm{strength}_S(W) > \delta$, but W is redundant w.r.t. S and $\mathbb{W}_S$. In other words, $\mathbb{W}_S$ is a redundant set for W. Observe that Alg. 2.11 removes any hyperedge Y from $\mathbb{W}_S$ only when $\mathbb{W}_S$ is a redundant set for Y. As discussed in the proof of Lemma 35, such removals preserve the property of $\mathbb{W}_S$ being a redundant set for W. Therefore, W will be redundant w.r.t. S and $\mathbb{W}_S$ throughout the remainder of the runtime of Alg. 2.10 and does not need to be checked twice.

- $\mathrm{strength}_S(W) > \delta$, X is not redundant w.r.t. S and $\mathbb{W}_S$, so it is added to $\mathbb{W}_S$ and retained during the removal of redundant edges on line 4 of Alg. 2.11. Consider the result of checking the pair S, W again later on in the execution of Alg. 2.10. Three outcomes may happen: either $\mathrm{strength}_S(W) \leq \delta$, or W would still be in $\mathbb{W}_S$, or W would be replaced by some its redundant set in $\mathbb{W}_S$. In any of the three outcomes W will not be added to $\mathbb{W}_S$, so again there is no need to check the pair S, W again.

The enumeration above of the outcomes of the first attempt to insert W into $\mathbb{W}_S$ is exhaustive and shows that every pair S, W has to be checked at most once. Although not reflected in the pseudocode of Alg. 2.10 and Alg. 2.11, it is straightforward to exclude the already tested pairs S, W from being tested again on line 3 of Alg. 2.11 (for example, by picking the next hyperedge to be tested in some fixed order and recording the last tested hyperedge for every candidate separator S). With such a modification, the complexity of Alg. 2.10 due to maintaining and modifying the sets $\mathbb{W}_S$ is

$$O\left(n^{2k+1}\left(J^{MI}_{2k+1}k^3 + n^2k\right)\right).$$

In addition to modifying $\mathbb{W}_S$, Alg. 2.10 also runs $FindConsistentDPGreedy$ on the current partitionings $\mathbb{Q}_S$. One run of *FindConsistentDPGreedy* takes $O(n^{k+2})$ time. *FindConsistentDPGreedy* is run at

most as many times as there are pairs $(S; W)$, because, as we have just shown, every W is added to $\mathbb{W}_S$ at most once. Thus the dynamic programming takes at most

$$O(n^{2k+1} \times n^{k+1}) = O(n^{3k+3})$$

time, and the total time complexity of Alg. 2.10 is

$$O\left(n^{2k+1} J^{MI}_{2k+1} k^3 + k n^{2k+3} + n^{3k+3}\right) = O\left(n^{2k+1} J^{MI}_{2k+1} k^3 + n^{3k+3}\right).$$

Alg. 2.10 always returns a junction tree. Every time *FindConsistentDPGreedy* is run, a hyperedge W is removed from some $\mathbb{W}_S$ and is never inserted back. Unless the algorithm returns a junction tree earlier, at some point the strongest possible hyperedge,

$$W^* \arg\max_{S,W} \mathrm{strength}_S(W)$$

will be removed and δ will be set to $\mathrm{strength}_S(W*)$. At this point all $\mathbb{W}_S$ will be empty and no hyperedge will be possible to add to $\mathbb{W}_S$. Every $\mathbb{Q}_S$ will then consist of singleton sets, and *FindConsistentDPGreedy* will find a junction tree with some separator connected to all the cliques.

The returned junction tree is a $n(\varepsilon + k\delta^*)$-junction tree. If Alg. 2.10 returns a junction tree $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, Alg. 2.11 ensures that for every $S \in \mathbb{S}$ every set $W \subseteq X_{\text{-}S}$ such that $\exists Q \in \mathbb{Q}_S$ with $W \cap Q \neq \emptyset, W \cap X_{\text{-}SQ} \neq \emptyset$ has strength at most δ^*. Indeed, if there was such W with strength more than δ, it would be added to $\mathbb{W}_S$ on line 4 of Alg. 2.11. Moreover, W would not be eliminated by the subsequent redundancy check because it involves variables of Q and some other connected component of the graph with hyperedges $\mathbb{W}_S$. Consequently $\mathbb{Q}_S$ would change, Alg. 2.11 would return failure and Alg. 2.10 would not return $(\mathbb{T}, \mathbb{C}, \mathbb{S})$, a contradiction. Therefore, from Corollary 17, $(\mathbb{T}, \mathbb{C}, \mathbb{S})$ is a $n(\varepsilon + k\delta^*)$-junction tree. □

Bibliography

Pieter Abbeel, Daphne Koller, and Andrew Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7, 2006.

Russell Almond and Augustine Kong. Optimality issues in constructing a Markov tree from graphical models. Technical Report 329, University of Chicago, Department of Statistics, 1991.

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov support vector machines. In *International Conference on Machine Learning (ICML)*, 2003.

Galen Andrew and Jianfeng Gao. Scalable training of L_1-regularized log-linear models. In *International Conference on Machine Learning (ICML)*, 2007.

Stefan Arnborg, Derek Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2), 1987.

Francis Bach and Michael Jordan. Thin junction trees. In *Conference on Neural Information Processing Systems (NIPS)*, 2002.

Adrian Barbu and Song-Chun Zhu. Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 2005.

Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David Blei, and Michael Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3, 2003.

Ingo Beinlich, Jaap Suermondt, Martin Chavez, and Gregory Cooper. The ALARM monitoring system: A case study with two probablistic inference techniques for belief networks. In *European Conference on Artificial Intelligence in Medicine*, 1988.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2), 1974.

Christopher Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

Christian Borgelt and Rudolf Kruse. Probabilistic graphical models for the diagnosis of analog electrical circuits. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, 2005.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1996.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. URL `http://www.stanford.edu/~{}boyd/cvxbook/`.

Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

Joseph Bradley and Carlos Guestrin. Learning tree conditional random fields. In *International Conference on Machine Learning (ICML)*, 2010.

John Breese and David Heckerman. Decision-theoretic troubleshooting: A framework for repair and experiment. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1996.

Anton Chechetka and Carlos Guestrin. Efficient principled learning of thin junction trees. In *Conference on Neural Information Processing Systems (NIPS)*, 2007.

Anton Chechetka and Carlos Guestrin. Focused belief propagation for query-specific inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010a.

Anton Chechetka and Carlos Guestrin. Evidence-specific structures for rich tractable CRFs. In *Conference on Neural Information Processing Systems (NIPS)*, 2010b.

David Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*. Springer-Verlag, 1996.

David Maxwell Chickering and Christopher Meek. Finding optimal Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

Arthur Choi and Adnan Darwiche. An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *National Conference on Artificial Intelligence (AAAI)*, 2006a.

Arthur Choi and Adnan Darwiche. A variational approach for approximating Bayesian networks by edge deletion. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006b.

Arthur Choi and Adnan Darwiche. Focusing generalizations of belief propagation on targeted queries. In *National Conference on Artificial Intelligence (AAAI)*, 2008.

Arthur Choi and Adnan Darwiche. Relax then compensate: On max-product belief propagation and more. In *Conference on Neural Information Processing Systems (NIPS)*, 2009.

Arthur Choi, Hei Chan, and Adnan Darwiche. On Bayesian network approximation by edge deletion. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3), 1968.

Gregory Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3), 1990.

Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. McGraw-Hill, 2001.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3), 1995.

Robert Cowell, Philip Dawid, Steffen Lauritzen, and David Spiegelhalter. *Probabilistic Networks and Expert Systems (Information Science and Statistics)*. Springer, 2003.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2, 2002.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. In *National Conference on Artificial Intelligence (AAAI)*, 1998.

Antonio Criminisi. Microsoft research Cambridge object recognition image database, 2004. URL `http://research.microsoft.com/vision/cambridge/recognition/`.

Paul Dagum and Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1), 1993.

Adnan Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3), 2003.

Sanjoy Dasgupta. Learning polytrees. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999.

Rodrigo de Salvo Braz, Sriram Natarajan, Hung Bui, Jude Shavlik, and Stuart Russell. Anytime lifted belief propagation. In *International Workshop on Statistical Relational Learning*, 2009.

Rina Dechter and Robert Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3), 2007.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), 1997.

Amol Deshpande, Minos Garofalakis, and Michael Jordan. Efficient stepwise selection in decomposable models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

Amol Deshpande, Carlos Guestrin, Sam Madden, Joseph Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *International Conference on Very Large Data Bases (VLDB)*, 2004.

Marie desJardins, Priyang Rathod, and Lise Getoor. Bayesian network learning with abstraction hierarchies and context-specific independence. In *European Conference on Machine Learning (ECML)*, 2005.

Reinhard Diestel. *Graph Theory*. Springer-Verlag Heidelberg, 2005.

Edsger Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 1959.

Daniel Eaton and Kevin Murphy. Bayesian structure learning using dynamic programming and MCMC. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

Gal Elidan and Stephen Gould. Learning bounded treewidth Bayesian networks. *Journal of Machine Learning Research*, 9, 2008.

Gal Elidan, Ian McGraw, and Daphne Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

Markus Enzweiler and Dariu Gavrila. A multi-level mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 2011.

R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7 (2), 1964.

Ian Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995.

Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659), 2004.

Nir Friedman and Daphne Koller. Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1), 2003.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration

of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 1984.

Hans-Otto Georgii. *Gibbs Measures and Phase Transitions*. Walter De Gruyter Inc, 1988.

Lise Getoor and Benjamin Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

Basilis Gidas. Consistency of maximum likelihood and pseudo-likelihood estimators for Gibbs distributions. In *Stochastic Differential Systems, Stochastic Control Theory and Applications*. Springer, 1988.

Vibhav Gogate, William Webb, and Pedro Domingos. Learning efficient Markov networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2010.

Joseph Gonzalez, Yucheng Low, and Carlos Guestrin. Residual splash for optimally parallelizing belief propagation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel Gibbs sampling: From colored fields to thin junction trees. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3), 2008.

Stephen Gould, Olga Russakovsky, Ian Goodfellow, Paul Baumstarck, Andrew Ng, and Daphne Koller. The STAIR vision library (v2.4), 2010. URL `http://ai.stanford.edu/~sgould/svl`.

Asela Gunawardana, Milind Mahajan, Alex Acero, and John Platt. Hidden conditional random fields for phone classification. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2005.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 2003.

John Hammersley and Peter Clifford. Markov fields on finite graphs and lattices, 1971.

Wolfgang Härdle, Marlene Müller, Stefan Sperlich, and Axel Werwatz. *Nonparametric and Semiparametric Models*. Springer, 2004.

Tom Heskes. Stable fixed points of loopy belief propagation are local minima of the Bethe free energy. In *Conference on Neural Information Processing Systems (NIPS)*, 2002.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301), 1963.

Klaus Höffgen. Learning and robust learning of product distributions. In *Conference on Learning Theory (COLT)*, 1993.

IBM. Cplex Solver 10.0, 2010. URL `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

Alexander Ihler. Accuracy bounds for belief propagation. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

Alexander Ihler, John Fisher III, and Alan Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6, 2005.

Claus Skaanning Jensen, Augustine Kong, and Uffe Kjaerulff. Blocking Gibbs sampling in very large probabilistic expert systems. *International Journal of HumanComputer Studies*, 42, 1995.

Finn Jensen and Frank Jensen. Optimal junction trees. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1994.

Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2), 1999.

Leslie Pack Kaelbling, Michael Littman, and Andrew Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.

David Karger and Nathan Srebro. Learning Markov networks: Maximum bounded tree-width graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2001.

Richard Karp. Reducibility among combinatorial problems. In *Symposium on the Complexity of Computer Computations*, 1972.

Robert Howard Kassel. *A comparison of approaches to on-line handwritten character recognition*. PhD thesis, Massachusetts Institute of Technology, 1995.

Uffe Kjaerulff. Approximation of Bayesian networks through edge removals. Technical report, Aalborg University, 1993.

Donald Knuth. Dancing links. *Millennial Perspectives in Computer Science*, 2000.

Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5, 2004.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004.

Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

Solomon Kullback and Richard Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22, 1951.

Sanjiv Kumar and Martial Hebert. A hierarchical field framework for unified context-based classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

Lubor Ladicky, Christopher Russell, Pushmeet Kohli, and Philip Torr. Associative hierarchical CRFs for object class image segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001.

Su-In Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of Markov networks using l_1-regularization. In *Conference on Neural Information Processing Systems (NIPS)*, 2006.

Stan Li. *Markov random field modeling in computer vision*. Springer-Verlag, 1995.

Dong Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3), 1989.

Han Liu and Jian Zhang. Estimation consistency of the group lasso and its applications. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Daniel Lowd and Pedro Domingos. Learning arithmetic circuits. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

David Madigan and Jeremy York. Bayesian graphical models for discrete data. *International Statistical Review*, 63, 1995.

Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

Francesco Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5), 1991.

Robert McEliece, David MacKay, and Jung fu Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2), 1998.

Geoffrey McLachlan and David Peel. *Finite Mixture Models*. Wiley-Interscience, 2000.

Marina Meilă and Michael Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1, 2001.

Joris Mooij and Hilbert Kappen. On the properties of the Bethe approximation and loopy belief propagation on binary networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11), 2005.

Joris Mooij and Hilbert Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Transactions on Information Theory*, 12(53), 2007.

Joris Mooij and Hilbert Kappen. Bounds on marginal probability distributions. In *Conference on Neural Information Processing Systems (NIPS)*, 2008.

Greg Mori, Xiaofeng Ren, Alexei A. Efros, and Jitendra Malik. Recovering human body configurations: Combining segmentation and recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

Mukund Narasimhan and Jeff Bilmes. PAC-learning bounded tree-width graphical models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

Aniruddh Nath and Pedro Domingos. Efficient belief propagation for utility maximization and repeated inference. In *National Conference on Artificial Intelligence (AAAI)*, 2010.

Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Conference on Neural Information Processing Systems (NIPS)*, 2001.

James Park. MAP complexity results and approximation methods. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

Mark Paskin and Carlos Guestrin. Robust probabilistic inference in distributed systems. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

Daniel Peng and Frank Dabek. Large-scale incremental processing using distributed transactions and notifications. In *USENIX Conference on Operating Systems Design and Implementation*, 2010.

William Pentney, Ana-Maria Popescu, Shiaokai Wang, Henry A. Kautz, and Matthai Philipose. Sensor-based understanding of daily life via large-scale use of common sense. In *National Conference on Artificial Intelligence (AAAI)*, 2006.

William Pentney, Matthai Philipose, Jeff Bilmes, and Henry Kautz. Learning large scale common sense models of everyday life. In *National Conference on Artificial Intelligence (AAAI)*, 2007.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3, 2003.

Gregory Pottie and William Kaiser. Wireless integrated network sensors. *Communications of the ACM*,

43, 2000.

Maurice Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82(1), 1998.

Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. (Online) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2003.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

Neil Robertson and Paul Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1), 1984.

Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2), 1996.

Jarkko Salojärvi, Kai Puolamäki, and Samuel Kaski. Expectation maximization algorithms for conditional likelihoods. In *International Conference on Machine Learning (ICML)*, 2005.

Ashutosh Saxena, Sung Chung, and Andrew Ng. 3-D depth reconstruction from a single still image. *International Journal of Computer Vision*, 76, 2008.

Robert Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3), 1999.

Mark Schmidt, Alexandru Niculescu-Mizil, and Kevin Murphy. Learning graphical model structure using l_1-regularization paths. In *Conference on Artificial Intelligence (AAAI)*, 2007.

Mark Schmidt, Kevin Murphy, Glenn Fung, and Rómer Rosales. Structure learning in random fields for heart motion abnormality detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 1978.

Glenn Shafer and Prakash Shenoy. Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2, 1990.

Dafna Shahaf, Anton Chechetka, and Carlos Guestrin. Learning thin junction trees via graph cuts. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Solomon Eyal Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2), 1994.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, 2006.

Ajit Singh and Andrew Moore. Finding optimal Bayesian networks by dynamic programming. Technical Report CMU-CALD-05-106, Carnegie Mellon University, Center for Automated Learning and Discovery, 2005.

Parag Singla and Pedro Domingos. Discriminative training of Markov logic networks. In *National Conference on Artificial Intelligence (AAAI)*, 2005.

Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *National Conference on Artificial Intelligence (AAAI)*, 2008.

Le Song, Arthur Gretton, Danny Bickson, Yucheng Low, and Carlos Guestrin. Kernel belief propagation.

In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, 2001.

Erik Sudderth, Alexander Ihler, William Freeman, and Alan Willsky. Nonparametric belief propagation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and its Connections*, 2004.

Charles Sutton and Andrew McCallum. Piecewise pseudolikelihood for efficient CRF training. In *International Conference on Machine Learning (ICML)*, 2007.

Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. Max-margin parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

Benjamin Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2003.

Sekhar Tatikonda and Michael Jordan. Loopy belief propogation and Gibbs measures. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

Bo Thiesson, Christopher Meek, David Maxwell Chickering, and David Heckerman. Learning mixtures of DAG models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998.

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006.

Antonio Torralba, Kevin Murphy, and William Freeman. Contextual models for object detection using boosted random fields. In *Conference on Neural Information Processing Systems (NIPS)*, 2004.

Douglas Vail and Manuela Veloso. Feature selection for activity recognition in multi-robot domains. In *National Conference on Artificial Intelligence (AAAI)*, 2008.

Douglas Vail, Manuela Veloso, and John Lafferty. Conditional random fields for activity recognition. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2007.

Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 1984.

Robert van Engelen. Approximating Bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8), 1997.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

Martin Wainwright and Michael Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49, 2003.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. MAP estimation via agreement on trees:

message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11), 2005.

Martin Wainwright, Pradeep Ravikumar, and John Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *Conference on Neural Information Processing Systems (NIPS)*, 2006.

Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1), 2000.

Yair Weiss and William Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2), 2001a.

Yair Weiss and William Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10), 2001b.

Michael Wellman, John Breese, and Robert Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 7, 1992.

Chen Yanover and Yair Weiss. Approximate inference and protein folding. In *Conference on Neural Information Processing Systems (NIPS)*, 2002.

Chen Yanover, Ora Schueler-Furman, and Yair Weiss. Minimizing and learning energy functions for side-chain prediction. *Research in Computational Molecular Biology*, 2007.

Jonathan Yedidia, William Freeman, and Yair Weiss. Generalized belief propagation. In *Conference on Neural Information Processing Systems (NIPS)*, 2000.

Jonathan Yedidia, William Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*. Morgan Kaufmann Publishers Inc., 2003.

Jonathan Yedidia, William Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51, 2005.

Jun Zhu and Eric Xing. Maximum entropy discrimination Markov networks. *Journal of Machine Learning Research*, 10, 2009.

Jun Zhu, Ni Lao, and Eric Xing. Grafting-light: fast, incremental feature selection and structure learning of Markov random fields. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.