Australian & New Zealand Journal of Statistics

Aust. N. Z. J. Stat. **61**(2), 2019, 89–133 doi: 10.1111/anzs.12257

Exact or approximate inference in graphical models: why the choice is dictated by the treewidth, and how variable elimination can be exploited

N. Peyrard[1,*] M.-J. Cros[1] S. de Givry[1] A. Franc[2]
S. Robin[3,4] R. Sabbadin[1] T. Schiex and M. Vignes[5]

INRA, France and School of Fundamental Sciences, New Zealand

Summary

Probabilistic graphical models offer a powerful framework to account for the dependence structure between variables, which is represented as a graph. However, the dependence between variables may render inference tasks intractable. In this paper, we review techniques exploiting the graph structure for exact inference, borrowed from optimisation and computer science. They are built on the principle of variable elimination whose complexity is dictated in an intricate way by the order in which variables are eliminated. The so-called treewidth of the graph characterises this algorithmic complexity: low-treewidth graphs can be processed efficiently. The first point that we illustrate is therefore the idea that for inference in graphical models, the number of variables is not the limiting factor, and it is worth checking the width of several tree decompositions of the graph before resorting to the approximate method. We show how algorithms providing an upper bound of the treewidth can be exploited to derive a 'good' elimination order enabling to realise exact inference. The second point is that when the treewidth is too large, algorithms for approximate inference linked to the principle of variable elimination, such as loopy belief propagation and variational approaches, can lead to accurate results while being much less time consuming than Monte-Carlo approaches. We illustrate the techniques reviewed in this article on benchmarks of inference problems in genetic linkage analysis and computer vision, as well as on hidden variables restoration in coupled Hidden Markov Models.

Key words: computational inference; marginalisation; message passing; mode evaluation; variational approximations

1. Introduction

Graphical models (Lauritzen 1996; Bishop 2006; Koller & Friedman 2009; Barber 2012; Murphy 2012) are formed by variables linked to each other by stochastic relationships. They enable the modelling of dependencies in possibly high-dimensional heterogeneous data and the capture of uncertainty. Graphical models have been applied in a wide range of areas when elementary units locally interact with each other, such as image analysis (Solomon & Breckon 2011), speech recognition (Baker *et al.* 2009), bioinformatics (Liu *et al.* 2009;

*Author to whom correspondence should be addressed.
[1]INRA UR 875 MIAT, Chemin de Borde Rouge, 31326, Castanet-Tolosan, France
e-mail: nathalie.peyrard@inra.fr
[2]INRA UMR 1202, Biodiversité, Gènes et Communautés, 69, route d'Arcachon, Pierroton, 33612, Cestas Cedex, France
[3]AgroParisTech, UMR 518 MIA, 16 rue Claude Bernard, Paris 5e, France
[4]INRA, UMR 518 MIA, 16 rue Claude Bernard, Paris 5e, France
[5]Institute of Fundamental Sciences, Massey University, Palmerston North, New Zealand

Maathuis *et al.* 2010; Höhna *et al.* 2014) and ecology (Illian *et al.* 2013; Bonneau *et al.* 2014; Carriger & Barron 2016) to name a few.

In real applications, a large number of random variables with a complex dependency structure are involved. As a consequence, inference tasks such as the calculation of a normalisation constant, of a marginal distribution or of the mode of the joint distribution can be challenging. Three main approaches exist to evaluate such quantities for a given distribution p defining a graphical model: (i) compute them in an exact manner; (ii) use a stochastic algorithm to sample from the distribution p to get (unbiased) estimates; (iii) derive an approximation of p for which the exact calculation is possible. Exact computation on p is appealing. However, if a brute-force method is used, it can lead to very time and memory consuming procedures for large problems. But for some graphical model structures, the exact inference can be performed using efficient techniques, even for a large number of variables. This is the topic of this article. Approach (*b*) is probably the most widely used by statisticians and modellers. Stochastic algorithms such as Monte-Carlo Markov Chains, MCMC (Robert & Casella 2004), Gibbs sampling (Geman & Geman 1984; Casella & George 1992) and particle filtering (Gordon, Salmond & Smith 1993) have become standard tools in many fields of application using statistical models. The last approach includes variational approximation techniques (Wainwright & Jordan 2008), which are starting to become common practice in computational statistics. In essence, approaches of type (*b*) provide an approximate answer to the original problem whereas approaches of type (*c*) provide an exact answer to an approximate problem, simpler to the original one but assumed to be a good representation.

In this paper, we focus on approaches of type (1) and (3), and we will review techniques for exact or approximate inference in graphical models borrowed from both optimisation and computer science. They are computationally efficient, yet not always standard in the statistician's toolkit. The characterisation of the structure of the graph G associated with a graphical model (precise definitions are given in Section 2) enables both determination of whether the exact calculation of the quantities of interest (marginal distribution, normalisation constant, mode) can be implemented efficiently and the derivation of a class of operational algorithms. When the exact calculation cannot be achieved efficiently, a similar analysis of the problem enables the practitioner to design algorithms to compute an approximation of the desired quantities with an associated acceptable complexity. Our aim is to provide the reader with the key elements to understand the power of these tools for statistical inference in graphical models.

The central algorithmic tool we focus on in this paper is the variable elimination concept (Bertelé & Brioshi 1972). In Section 3, we adopt a unified algebraic presentation of the different inference tasks (marginalisation, normalising constant or mode evaluation) to emphasise that each of them can be solved using a particular case of a variable elimination scheme. Consequently, the work done to demonstrate that variable elimination is efficient for one task passes on to the other ones. The key ingredient in the design of efficient algorithms based on variable elimination is the clever use of distributivity between algebraic operators. For instance, distributivity of the product ($\times$) over the sum ($+$) validates the equation $(a \times b) + (a \times c) = a \times (b + c)$ and evaluating the left-hand side of this equality requires two multiplications and one addition while evaluating the right-hand side requires one multiplication and one addition. Similarly since $\max(a + b, a + c) = a + \max(b, c)$ it is more efficient to compute the right-hand side from an algorithmic point of view. Distributivity enables minimisation of the number of operations. To perform variable elimination,

associativity and commutativity properties are also required, and underlying algebra is a semi-ring (from which some notations will be borrowed). Inference algorithms using the distributivity property have been known and published in the Artificial Intelligence and Machine Learning literature under different names, such as sum-prod, or max-sum (Pearl 1988; Bishop 2006). They are typical examples of variable elimination procedures.

Variable elimination relies on the choice of an order of elimination of the variables, via successive marginalisation or maximisation operations. The calculations are performed according to this ordering when applying distributivity. The topology of the graph G provides key information for optimal organisation of the calculations to minimise the number of elementary operations to perform. For example, when the graph is a tree, the most efficient elimination order corresponds to recursive elimination of the vertices of degree one. One starts from the leaves towards the root, and inner nodes of higher degree successively become leaves. The notion of an optimal elimination order for inference in an arbitrary graphical model is closely linked to the notion of treewidth of the associated graph G. We will see in Section 3 the reason why inference algorithms based on variable elimination with the best elimination order are of linear complexity in n, the number of variables/nodes in the graph, i.e. the size of the graph, but exponential complexity in the treewidth. Therefore treewidth is the main characterisation of G to determine if the exact inference is possible in practice or not. This notion has lead to the development of several methods for solving apparently complex inference problems, which have then been applied in biology (e.g. Tamura & Akutsu 2014). More details on these methodological and applied results are provided in the Conclusion Section.

The concept of treewidth has been proposed in computer science (Bodlaender 1993), in discrete mathematics and graph minor theory (see Robertson & Seymour 1986; Lovász 2005). Discrete mathematics existence theorems (Robertson & Seymour 1986) establish that there exists an algorithm for computing the treewidth of any graph with complexity polynomial in n (but exponential in the treewidth), and the degree of the polynomial is determined. However, this result does not tell how to derive and implement the algorithm, apart from some very specific cases such as trees, chordal graphs, and series-parallel graphs (Duffin 1965). Section 4 introduces the reader to several state-of-the-art algorithms that provide an upper bound of the treewidth, together with an associated elimination order. These algorithms are therefore useful tools to test if exact inference is achievable and, if applicable, to derive an exact inference algorithm based on variable elimination. Their behaviour is illustrated on benchmarks borrowed from combinatorial optimisation competitions.

Variable elimination also leads to message passing algorithms (Pearl 1988) which are now common tools in Computer Science or Machine Learning for marginal or mode evaluation. More recently, these algorithms have been reinterpreted as a way to re-parameterise the original graphical model into an updated one with different potential functions by still representing the same joint distribution (Koller & Friedman 2009). We explain in Section 5 how re-parametrisation can be used as a pre-processing tool to obtain a new parameterisation for which the inference becomes simpler. Message passing is not the only way to perform re-parametrisation, and we discuss alternative efficient algorithms proposed in the context of constraint satisfaction problems (CSP, see Rossi, van Beek & Walsh 2006). These latter algorithms have, to the best of our knowledge, not yet been exploited in the context of graphical models.

As emphasised above, algorithms in polynomial time can be designed in general for graphical models with limited treewidth only, even if it may happen to be feasible for some

specific cases with large tree-width as well (Tarlow, Givoni & Zemel 2010 and Serang 2014). Although this is not the case for many graphs, the principles of variable elimination and message passing for a tree can be applied to any graph leading to heuristic inference algorithms. The most famous heuristic is the Loopy Belief Propagation algorithm, LBP (see Kschischang, Frey & Loeliger 2001). We recall in Section 6 the result that establishes LBP as a variational approximation method. Variational methods rely on the choice of a distribution which renders inference easier. They approximate the original complex graphical model. The approximate distribution is chosen within a class of models for which efficient inference algorithms exist, that is models with small treewidth (0, 1 or 2 in practice). We review some standard choices of approximate distributions, each of them corresponds to a different underlying treewidth.

Finally, Section 7 illustrates the techniques reviewed in the article, on the case of Coupled Hidden Markov Model, CHMM (see Brand 1997). We first compare them to the problem of mode inference in a CHMM devoted to the study of pest propagation. Then we exemplify the use of different variational methods combined with the Expectation-Maximisation algorithm, EM (Dempster, Laird & Rubin 1977) to perform parameter estimation in CHMM.

Because we wanted this article useful for readers not familiar with the notion of variable elimination and treewidth, and also to provide some elements to further explore the topic, the article is quite long. For the reader interested in understanding why the treewidth is important for inference in graphical models and what algorithms can be used in practice that exploit the treewidth for exact inference, we suggest reading until Section 5.1. Then, for those who want to have a broader understanding of the topic, several directions are possible: Sections 5.2 and 5.3 present a more advanced use of variable elimination; Sections 5.2 and Section 6 introduce approximate inference methods related to variable elimination and they are illustrated in Section 7; in Sections 2, 3 and 5, the paragraphs on deterministic graphical models present the analogies with the domain of cost functions, to encourage cross-fertilisation.

2. Graphical models

2.1. Models definition

Consider a stochastic system defined by a set of random variables $\boldsymbol{X}^\top = (X_1, \ldots, X_n)$. Each variable X_i takes values in Λ_i. A realisation of $\boldsymbol{X}$ is denoted $\boldsymbol{x}^\top = (x_1, \ldots, x_n)$, with $x_i \in \Lambda_i$. The set of all possible realisations is called the state space, and is denoted $\Lambda = \prod_{i=1}^n \Lambda_i$. If A is a subset of $V = \{1, \ldots, n\}$, then X_A, x_A and Λ_A are respectively the subset of random variables $\{X_i, i \in A\}$, a possible realisation $\{x_i, i \in A\}$ of X_A and the state space of X_A respectively. If p is the joint probability distribution of $\boldsymbol{X}$ on Λ, we denote for all $\boldsymbol{x} \in \Lambda$

$$p(\boldsymbol{x}) = \Pr(\boldsymbol{X} = \boldsymbol{x}).$$

Note that we focus here on discrete variables (we will discuss inference in the case of continuous variables on examples in Section 8). A joint distribution p on Λ is said to be a probabilistic graphical model (Lauritzen 1996; Bishop 2006; Koller & Friedman 2009) indexed on a set $\mathcal{B}$ of parts of V if there exists a set $\Psi = \{\psi_B\}_{B \in \mathcal{B}}$ of maps from Λ_B to $\mathbb{R}^+$, called potential functions, indexed by $\mathcal{B}$ such that p can be expressed in the following factorised form:

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{B \in \mathcal{B}} \psi_B(x_B), \tag{1}$$

where $Z = \sum_{\boldsymbol{x} \in \Lambda} \prod_{B \in \mathcal{B}} \psi_B(x_B)$ is the normalising constant, also called partition function. The elements $B \in \mathcal{B}$ are the scopes of the potential functions and $|B|$ is the arity of the potential function ψ_B. The set of scopes of all the potential functions involving variable X_i is denoted $\mathcal{B}_i = \{B \in \mathcal{B} : i \in B\}$

One desirable property of graphical models is that of Markov local independence: if $p(\boldsymbol{x})$ can be expressed as in (1), then a variable X_i is (stochastically) independent of all others in $\boldsymbol{X}$ conditionally to the set of variables $X_{(\cup_{B \in \mathcal{B}_i} B) \setminus i}$. The set $X_{(\cup_{B \in \mathcal{B}_i} B) \setminus i}$ is called the Markov blanket of X_i, or its neighbourhood (Koller & Friedman 2009, Chapter 4). It is denoted N_i. These conditional independences can be represented, by a graph with one vertex per variable in $\boldsymbol{X}$. The question of encoding the independence properties associated with a given distribution into a graph structure has been widely described (e.g. Koller & Friedman 2009, Chapters 3 and 4), and we will not discuss it here. We consider the classical graph $G = (V, E)$ associated with the decomposition dictated in (1), where an edge is drawn between two vertices i and j if there exists $B \in \mathcal{B}$ such that i and j are in B. (Note that for sake of clarity, in some figures we will denote a node by the random variable it corresponds to). Such a representation of a graphical model is actually not as rich as the representation of (1). For instance, if $n = 3$, the two cases $\mathcal{B} = \{\{1, 2, 3\}\}$ and $\mathcal{B} = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$ are represented by the same graph G, namely a clique (i.e. a fully connected set of vertices) of size 3. Without loss of generality, we could impose in the definition of a graphical model that scopes B correspond to cliques of G. In the above example where $\mathcal{B} = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$, this can be done by defining $\psi'_{1,2,3} = \psi_{12}\psi_{23}\psi_{13}$. The original structure is then lost, and ψ' is more costly to store

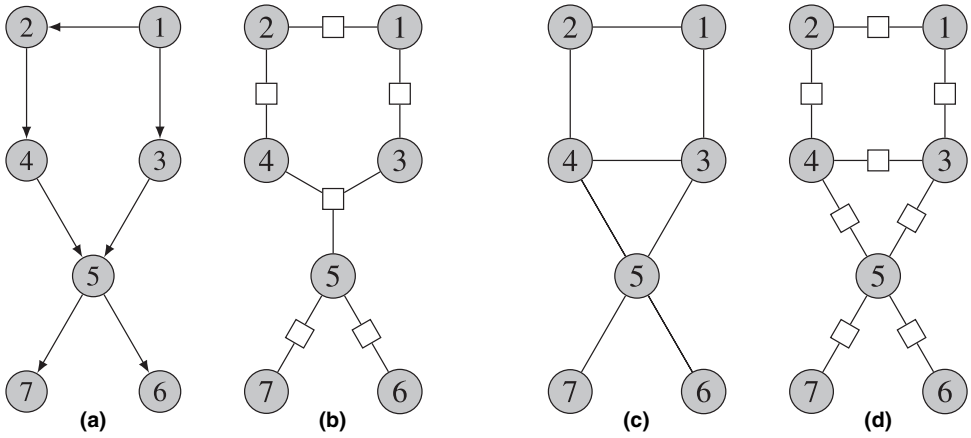

Figure 1. From left to right: (a) Graphical representation of a directed graphical model, where potential functions define the conditional probability of each variable given its parents values; (b) The corresponding factor graph, where every potential function is represented as a factor (square vertex) connected to the variables that are involved in it; (c) Graphical representation of an undirected graphical model. It is impossible from this graph to distinguish between a graphical model defined by a unique potential function on vertices 3, 4 and 5 from a model defined by 3 pairwise potential functions over each pair (3, 4), (3, 5) and (4, 5); (d) The corresponding factor graph, which unambiguously defines the potential functions, here three pairwise potential functions.

than the original potential functions. The factor graph representation goes beyond the limit of the representation G: this graphical representation is a bipartite graph with one vertex per potential function and one vertex per variable. Edges are only between functions and variables. An edge is present between a function vertex (also called factor vertex) and a variable vertex, if and only if the variable is in the scope of the potential function. Figure 1 displays examples of the two graphical representations.

Several families of probabilistic graphical models exist (Koller & Friedman 2009; Murphy 2012). They can be grouped into directed and undirected ones. The most classical directed framework is that of Bayesian network (Pearl 1988; Jensen & Nielsen 2007). In a Bayesian network, potential functions are conditional probabilities of a variable given its parents. In such models, trivially $Z = 1$. There is a representation by a directed graph where an edge is directed from a parent vertex to a child vertex (see Fig. 1a). The undirected graphical representation G is obtained by moralisation, i.e. by adding an edge between two parents of a same variables. Undirected probabilistic graphical models (see Fig. 1c) are equivalent to Markov Random Fields (MRF; Li 2001) as soon as the potential functions take values in $\mathbb{R}^+ \setminus \{0\}$. In an MRF, a potential function is not necessarily a probability distribution: ψ_B is not required to be normalised (as opposed to a Bayesian network model).

2.1.1. Deterministic graphical models

Although the term 'graphical models' is often used to refer to probabilistic graphical models, the idea of describing a joint interaction on a set of variables through local functions has also been used in artificial intelligence to concisely describe Boolean functions or cost functions, with no normalisation constraint. Throughout this article, we regularly refer to these deterministic graphical models, and we explain how the algorithms devoted to their optimisation can be directly applied to compute the mode in a probabilistic graphical model.

In a deterministic graphical model with only Boolean (0/1) potential functions, each potential function describes a constraint between variables. If the potential function takes value 1, the corresponding realisation is said to satisfy the constraint. If it takes value 0, the realisation does not satisfy it. The graphical model is known as a constraint network. It describes a joint Boolean function on all variables that takes value 1 if and only if all constraints are satisfied. The problem of finding a realisation that satisfies all the constraints, called a solution of the constraint network, is the CSP (Rossi, van Beek & Walsh 2006). This framework is used to model and solve combinatorial optimisation problems. There is a wide variety of software tools to solve it.

CSPs have been extended to describe joint cost functions, decomposed as a sum of local cost functions, f_B in the weighted constraint network or cost function network (Rossi, van Beek & Walsh 2006).

$$f(\boldsymbol{x}) = \sum_{B \in \mathcal{B}} f_B(x_B).$$

In this case, cost functions take finite or infinite integer or rational values: infinity enables to express hard constraints while finite values encode costs for unsatisfied soft constraints. The problem of finding a realisation of minimum cost is the Weighted Constraint Satisfaction Problem (WCSP), which is NP-hard. It is easy to observe that any probabilistic graphical model can be translated into a weighted constraint network, and vice versa using a simple $-\ln(\cdot)$ transformation.

$$f_B(x_B) = -\ln(\psi_B), \text{with } f_B(x_B) = +\infty \Leftrightarrow \psi_B(x_B) = 0.$$

Therefore the WCSP is equivalent to finding a realisation with maximal probability in a probabilistic graphical model. With this equivalence, it becomes possible to use exact WCSP resolution algorithms that have been developed in this field for mode evaluation or for the computation of Z, the normalising constant, in a probabilistic graphical model. See for instance Viricel *et al.* (2016), for an application on a problem of protein design.

2.2. Inference tasks in probabilistic graphical models

Computations on probabilities and potentials rely on two fundamental types of operations. Firstly, multiplication (or addition in the log domain) is used to combine potentials to define a joint potential distribution. Secondly, sum or max/min can be used to eliminate variables and compute marginals or modes of the joint distribution on subsets of variables. The precise identity of these two basic operations is not important for the inference algorithms based on variable elimination. We therefore adopt a presentation using generic operators to emphasise this property of algorithms. We denote as $\odot$ and as $\oplus$ the combination operator and the elimination operator, respectively. To be able to apply the variable elimination algorithm, the only requirement is that $(\mathbb{R}^+, \oplus, \odot)$ defines a commutative semi-ring. Specifically, the semi-ring algebra offers distributivity: $(a \odot b) \oplus (a \odot c) = a \odot (b \oplus c)$. For instance, this corresponds to the distributivity of the product operation over the sum operation, i.e. $(a \times b) + (a \times c) = a \times (b + c)$, or to the distributivity of the max operation over the sum operation, i.e. $\max(a+b, a+c) = a + \max(b, c)$, or to the distributivity of the max operation over the product operation, i.e. $\max(a \times b, a \times c) = a \times (\max(b, c))$. We extend the definition of the two abstract operators $\odot$ and $\oplus$ to operators on potential functions, as follows:

Combine operator: the combination of two potential functions ψ_A and ψ_B is a new function $\psi_A \odot \psi_B : \Lambda_{A \cup B} \to \mathbb{R}^+$ defined as $\psi_A \odot \psi_B(x_{A \cup B}) = \psi_A(x_A) \odot \psi_B(x_B)$.
Elimination operator: the elimination of variable $X_i, i \in B$ from a potential function ψ_B is a new function $(\oplus_{x_i} \psi_B) : \Lambda_{B \setminus \{i\}} \to \mathbb{R}^+$ defined as $(\oplus_{x_i} \psi_B)(x_{B \setminus \{i\}}) = \oplus_{x_i} (\psi_B(x_{B \setminus \{i\}}, x_i))$. For $\oplus = +$, $(\oplus_{x_i} \psi_B)(x_{B \setminus \{i\}})$ represents the marginal sum $\sum_{x_i} \psi_B(x_{B \setminus \{i\}}, x_i)$.

Classical counting and optimisation tasks in graphical models can now be entirely written with these two operators. For simplicity, we denote by $\oplus_{x_B}$, where $B \subset V$ a sequence of eliminations $\oplus_{x_i}$ for all $i \in B$, the result being insensitive to the order in a commutative semi-ring. Similarly, $\odot_{B \in \mathcal{B}}$ represents the successive combination of all potential functions ψ_B, with $B \in \mathcal{B}$.

Counting task. Under this name, we group all tasks that involve summing over the state space of a subset of variables in $\boldsymbol{X}$. This includes the computation of the partition function Z or of any marginal distribution, as well as entropy evaluation. For $A \subset V$ and $\bar{A} = V \setminus A$, the marginal distribution p_A of X_A associated with the joint distribution p is defined as:

$$p_A(x_A) = \sum_{x_{\bar{A}} \in \Lambda_{\bar{A}}} p(x_A, x_{\bar{A}}) = \frac{1}{Z} \sum_{x_{\bar{A}} \in \Lambda_{\bar{A}}} \prod_{B \in \mathcal{B}} \psi_B(x_B).$$

If we see Z as a constant function, this can be expressed as

$$p_A \odot Z = \left(\oplus_{x_{\bar{A}}} \left(\odot_{B \in \mathcal{B}} \psi_B \right) \right),$$

where $\odot$ combines functions using $\times$ and $\oplus$ eliminates variables using $+$.

Marginal evaluation is also interesting in the case where some variables are observed. If x_O ($O \subset V$) are the values of the observed values, the marginal conditional distribution can be computed by restricting the domains of variables X_O to the observed value. This is typically the kind of computational task required in the E-step of an EM algorithm, for parameter estimation of models with hidden data.

Optimisation task. The most common optimisation task in a graphical model corresponds to the evaluation of the most probable state $\boldsymbol{x}^*$ of the random vector $\boldsymbol{X}$, defined as

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\Lambda} p(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}\in\Lambda} \prod_{B\in\mathcal{B}} \psi_B(x_B) = \arg\max_{\boldsymbol{x}\in\Lambda} \sum_{B\in\mathcal{B}} \ln \psi_B(x_B).$$

When p is an a posteriori distribution, $\boldsymbol{x}^*$ is also called the Maximum A Posteriori (MAP). The maximum itself is $\oplus_{\boldsymbol{x}}(\odot_{B\in\mathcal{B}} \ln \psi_B(x_B))$ with $\oplus$ and $\odot$ set to max and to $+$, respectively. The computation of the mode $\boldsymbol{x}^*$ does not require the computation of the normalising constant Z, however evaluating the mode probability value $p(\boldsymbol{x}^*)$ does. Another optimisation task of interest is the computation of the max-marginals of each variable X_i defined as $p^*(x_i) = \max_{x_{V\setminus i}} p(\boldsymbol{x})$.

Therefore, counting and optimisation tasks can be interpreted as two instantiations of the same computational task expressed in terms of combination and elimination operators, namely $\oplus_{x_A} \odot_{B\in\mathcal{B}} \psi_B$, where $A \subseteq V$. When the combination operator $\odot$ and the elimination operator $\oplus$ are set to $\times$ and $+$, respectively, this computational problem is known as a sum-product problem in the Artificial Intelligence literature (Pearl 1988). When $\oplus$ and $\odot$ are set to the maximum and sum operator, respectively, it is a max-sum problem (Bishop 2006, Chapter 8). In practice, it means that tasks such as solving the E-step of the EM algorithm or computing the mode in a graphical model, belongs to the same family of computational problems.

We will see in Section 3 that there exists an exact algorithm solving this general task which exploits the distributivity of the combination and elimination operators to perform operations in a smart order. From this generic algorithm, known as variable elimination (Bertelé & Brioshi 1972) or bucket elimination (Dechter 1999), one can deduce exact algorithms to solve counting and optimisation tasks in a graphical model, by instantiating the operators $\oplus$ and $\odot$.

2.2.1. Deterministic Graphical models

CSP is a $\vee$-$\wedge$ problem as it can be defined using $\vee$ (logical 'or') as the elimination operator and $\wedge$ (logical 'and') as the combination operator over Booleans. The WCSP is a min-$+$ as it uses min as the elimination operator and $+$ (or bounded variants of $+$) as the combination operator. Several other variants exist (Rossi, van Beek & Walsh 2006), including generic algebraic variants (Schiex, Fargier & Verfaillie 1995; Bistarelli, Montanari & Rossi

Table 1. Definitions of the Combine ($\odot$) and the Elimination ($\oplus$) operators for classical tasks on probabilistic and deterministic graphical models.

Task	$\oplus$	$\odot$
Marginal evaluation	$+$	$\times$
Mode evaluation	max	$+$
Existence of a solution in a CSP	$\vee$	$\wedge$
Evaluation of the minimum cost in WCSP	min	$+$

1997; Kohlas 2003; Cooper 2004; Pralet, Verfaillie & Schiex 2007). The definitions of the Combine and the Elimination operators for classical tasks on probabilistic and deterministic graphical models are given in Table 1.

2.3. Example: CHMM

We now introduce the example of CHMMs, which can be seen as extensions of Hidden Markov Chain (HMC) models to several chains in interactions. In section 7, we will use this framework to illustrate the behaviour of exact and approximate algorithms based on variable elimination.

An HMC (Figure 2) is defined by two sequences of random variables $\boldsymbol{O}$ and $\boldsymbol{H}$ of the same length, T. A realisation $\boldsymbol{O}^\top = (o_1, \dots o_T)$ of the variables $\boldsymbol{O}^\top = (O_1, \dots O_T)$ is observed, while the states of variables $\boldsymbol{H}^\top = (H_1, \dots H_T)$ are unknown (hidden). In the HMC model the assumption is made that O_i is independent of $H_{V\setminus\{i\}}$ and $O_{V\setminus\{i\}}$ given the hidden variable H_i. These independences are modelled by pairwise potential functions $\psi_{H_i,O_i}, \forall 1 \leqslant i \leqslant T$. Furthermore, the hidden variable H_i is independent of $H_1, \dots, H_{i-2}$ and $O_1, \dots, O_{i-1}$ given the hidden variable H_{i-1}. These independences are modelled by pairwise potential functions $\psi_{H_{i-1},H_i}, \forall\, 1 < i \leqslant T$. Then the model is fully defined by specifying an additional potential function $\psi_{H_1}(h_1)$ to model the initial distribution. In the classical HMC formulation (Rabiner 1989), these potential functions are normalised conditional probability distributions i.e., $\psi_{H_{i-1},H_i}(h_{i-1}, h_i) = \Pr(H_i = h_i | H_{i-1} = h_{i-1})$, $\psi_{O_i,H_i}(o_i, h_i) = \Pr(O_i = o_i | H_i = h_i)$ and $\psi_{H_1}(h_1) = \Pr(H_1 = h_1)$. As a consequence, the normalising constant Z is equal to 1, as it is in Bayesian networks.

Consider now that there is more than one hidden chain: I signals are observed at times $t \in \{1, \dots T\}$ and we denote O_t^i the variable corresponding to the observed signal i at time t. Variable O_t^i depends on some hidden state H_t^i. The CHMM framework assumes dependency between two hidden chains at two consecutive time steps (see Brand 1997): H_t^i depends not only on H_{t-1}^i, it may depend on some H_{t-1}^j for $j \neq i$. The set of the indices of chains upon which H_t^i depends (except i) is denoted L_i. This results in the graphical structure displayed on Figure 3, where $L_2 = \{1, 3\}$ and $L_1 = L_3 = \{2\}$. Such models have been considered in a series of domains such as bioinformatics (Choi *et al.* 2013), electroencephalogram analysis (Zhong & Ghosh 2002) or speech recognition (Nock & Ostendorf 2003). In a CHMM setting, the joint distribution of the hidden variables $\boldsymbol{H} = (H_t^i)_{i,t}$ and observed variables $\boldsymbol{O} = (O_t^i)_{i,t}$ factorises as

$$p(\boldsymbol{h}, \boldsymbol{o}) \propto \prod_{i=1}^{I} \psi^{\text{init}}(h_1^i) \left(\prod_{i=1}^{I} \prod_{t=2}^{T} \psi^{M}(h_{t-1}^i, h_{t-1}^{L_i}, h_t^i) \right) \times \left(\prod_{i=1}^{I} \prod_{t=1}^{T} \psi^{E}(h_t^i, o_t^i) \right), \tag{2}$$

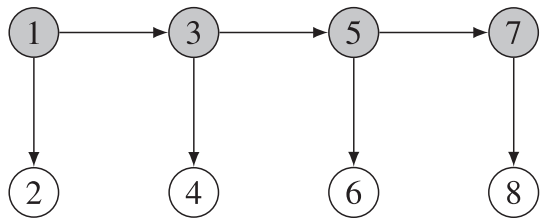

Figure 2. Graphical representation of an HMM. Hidden variables correspond to vertices 1, 3, 5, 7, and observed variables to vertices 2, 4, 6, 8.

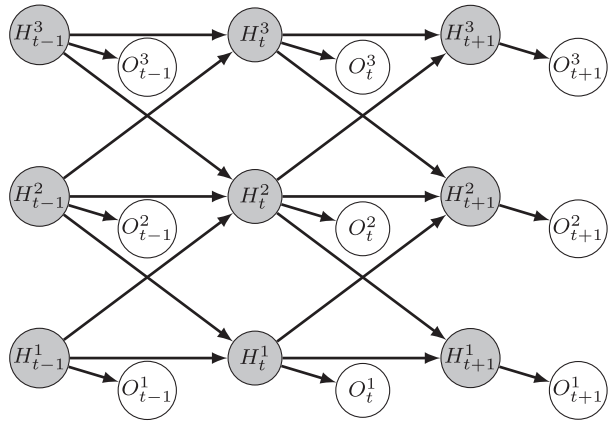

Figure 3. Graphical representation of a coupled HMM with 3 hidden chains.

where ψ^{init} is the initial distribution, ψ^M encodes the local transition function of H^i_t and ψ^E encodes the emission of the observed signal given the corresponding hidden state. A fairly comprehensive exploration of these models can be found in Murphy (2002).

Potential functions ψ^{init}, ψ^M and ψ^E can be parameterised by a set of parameters denoted θ. A classical problem for CHMMs is to have more than one iron in the fire: (i) estimate θ; and (ii) compute the mode of the conditional distribution of the hidden variables given the observations. Estimation can be performed using an EM algorithm, and as mentioned previously, the E-step of the algorithm and the mode computation task belong to the same family of computational tasks in graphical models. Both can be solved using variable elimination as discussed in the next section.

Beforehand, we present a reasonably simple example of a CHMM that will be used to illustrate the different inference algorithms introduced in this work. It models the dynamics of a pest that can spread on a landscape composed of I crop fields organised on a regular grid. The spatial neighbourhood of field i, denoted L_i, is the set of the four closest fields (three on the borders, and two in corners of the grid). $H^i_t \in \{0,1\}\,(1 \leqslant i \leqslant I,\ 1 \leqslant t \leqslant T)$ is the state of crop field i at time t. State 0 (respectively 1) represents the absence (presence) of the pest in the field. Variable H^i_t depends on H^i_{t-1} and on the H^j_{t-1}, for $j \in L_i$. The conditional probabilities of survival and apparition of the pest in field i are parameterised by 3 parameters: ϵ, the probability of contamination from outside the landscape (long-distance dispersal); ρ, the probability that the pest spreads from an infected field $j \in L_i$ to field i between two consecutive times; and v, the probability of field persistent infection between two consecutive times. We assume that contamination events from all neighbouring fields are independent. Then, if C^i_t is the number of contaminated neighbours of field i at time t (i.e. $C^i_t = \sum_{j \in L_i} H^j_t$), the contamination potential of field i at time t is:

$$\psi^M(0, h^{L_i}_{t-1}, 1) = \Pr(H^i_t = 1 | H^i_{t-1} = 0, h^j_{t-1}, j \in L_i) = \epsilon + (1-\epsilon)(1-(1-\rho)^{C^i_t}),$$

and its persistence in a contaminated state is:

$$\begin{aligned}\psi^M(1, h^{L_i}_{t-1}, 1) &= \Pr(H^i_t = 1 | h^i_{t-1} = 1, h^j_t, j \in L_i) \\ &= v + (1-v)\left(\epsilon + (1-\epsilon)(1-(1-\rho)^{C^i_t})\right).\end{aligned}$$

The (H_t^i)'s are hidden variables but monitoring observations are available. A binary variable O_t^i is observed: it takes value 1 if the pest was declared as present in the field; and 0 otherwise. Errors of detection are possible. False negative observations occur since even if the pest is there, it can be difficult to notice, and missed. On the contrary, false positive observations occur when the pest is mixed up with another one. We define the corresponding emission potential as $\psi^E(0,1) = \Pr(O_t^i = 0 | H_t^i = 1) = f_n$ and $\psi^E(1,0) = \Pr(O_t^i = 1 | H_t^i = 0) = f_p$, respectively.

3. Variable elimination for exact inference

We now describe the principle of variable elimination to solve the general inference tasks presented in Section 2.2 We first recall the Viterbi algorithm for HMC (Rabiner 1989), a classical example of variable elimination for optimisation (mode evaluation). Then, we formally describe the variable elimination procedure in the general graphical model framework. The key element is the choice of an ordering for the sequential elimination of the variables. It is closely linked to the notion of the treewidth of the graphical representation of the model. We explain how the complexity of a variable elimination algorithm is fully characterised by this notion. We also describe the extension to the elimination of blocks of variables.

3.1. Viterbi algorithm for HMC models

As a didactic introduction to exact inference on graphical models by variable elimination, we consider a well studied stochastic process: the discrete HMC.

A classical inference task for an HMC is to identify the most likely values of variables $\boldsymbol{H}$ given a realisation $\boldsymbol{o}$ of the variables $\boldsymbol{O}$. The problem is to compute $\arg\max_{\boldsymbol{h}} \Pr(\boldsymbol{H} = \boldsymbol{h} | \boldsymbol{O} = \boldsymbol{o})$, or equivalently the argument of:

$$\max_{h_1,\ldots,h_T} \left[(\psi_{H_1}(h_1)\psi_{O_1,H_1}(o_1,h_1)) \prod_{i=2}^{T} (\psi_{H_{i-1},H_i}(h_{i-1},h_i)\psi_{O_i,H_i}(o_i,h_i)) \right]. \tag{3}$$

The number of possible realisations of $\boldsymbol{H}$ is exponential in T. Nevertheless, this optimisation problem can be solved in a number of operations linear in T using the well-known Viterbi algorithm (Rabiner 1989). This algorithm, based on dynamic programming, performs successive eliminations (by maximisation) of all hidden variables, starting with H_T, and iteratively considering the H_i's for $i = T-1, T-2, \ldots, 1$. It successively computes the most likely sequence of hidden variables. By using distributivity between the max and the product operators, the elimination of variable H_T can be done by rewriting (3) as:

$$\max_{h_1,\ldots,h_{T-1}} \left[\psi_{H_1}(h_1)\psi_{O_1,H_1}(o_1,h_1) \right.$$
$$\left. \prod_{i=2}^{T-1} \left(\psi_{H_{i-1},H_i}(h_{i-1},h_i)\psi_{O_i,H_i}(o_i,h_i) \underbrace{\max_{h_T} \psi_{H_{T-1},H_T}(h_{T-1},h_T)\psi_{O_T,H_T}(o_T,h_T)}_{\text{New potential function}} \right) \right].$$

The new potential function created by maximising on H_T depends only on variable H_{T-1}. The same principle can then be applied to H_{T-1} and so forth. This is a simple application of the general variable elimination algorithm that we describe in the next section.

3.2. General principle of variable elimination

In Section 2, we have seen that counting and optimisation tasks can be formalised by the same generic algebraic formulation

$$\bigoplus_{x_A} \left(\bigodot_{B \in \mathcal{B}} \psi_B \right) \tag{4}$$

where $A \subseteq V$.

The trick behind variable elimination (Bertelé & Brioshi 1972) relies on a clever use of the distributivity property. Indeed, evaluating $(a \odot b) \oplus (a \odot c)$ as $a \odot (b \oplus c)$ requires fewer operations. Hence eliminating a in the second expression leads to fewer algebraic operations. Since distributivity applies both for counting and optimising tasks, variable elimination can be applied to both tasks. It also means that if variable elimination is efficient for one task it will also be efficient for the other one. As in the HMC example, the principle of the variable elimination algorithm for counting or optimising consists of eliminating variables one by one in an expression of the problem as in (4).

The elimination of the first variable, say $X_i, i \in A$, is performed by merging all potential functions involving X_i and applying operator $\oplus_{x_i}$ to these potential functions. Using commutativity and associativity of both operators, (4) can be rewritten as:

$$\bigoplus_{x_A} \left(\bigodot_{B \in \mathcal{B}} \psi_B \right) = \bigoplus_{x_{A \setminus \{i\}}} \bigoplus_{x_i} \left(\left(\bigodot_{B \in \mathcal{B} \setminus \mathcal{B}_i} \psi_B \right) \odot \left(\bigodot_{B \in \mathcal{B}_i} \psi_B \right) \right),$$

where $\mathcal{B}_i$ is the subset of V defined such as all its elements contain i. Then using distributivity of $\odot$ on $\oplus$, we obtain:

$$\bigoplus_{x_A} \left(\bigodot_{B \in \mathcal{B}} \psi_B \right) = \bigoplus_{x_{A \setminus \{i\}}} \left[\left(\bigodot_{B \in \mathcal{B} \setminus \mathcal{B}_i} \psi_B \right) \odot \underbrace{\left(\bigoplus_{x_i} \bigodot_{B \in \mathcal{B}_i} \psi_B \right)}_{\text{New potential function } \psi_{N_i}} \right]. \tag{5}$$

This shows that the elimination of X_i results in a new graphical model, where variable X_i and the potential functions $\psi_B, B \in \mathcal{B}_i = \{B', x_i \in B'\}$ do not appear anymore. They are replaced by a new potential ψ_{N_i} which does not involve X_i, but depends on its neighbours in G. The graph associated with the new graphical model is in a sense similar to the one of the original model. It is updated as follows: vertex X_i is removed, and neighbours X_{N_i} of X_i are now connected together in a clique because they are all in the scope of ψ_{N_i}. The new edges between the neighbours of X_i are called fill-in edges. For instance, when eliminating variable X_1 in the graph of Figure 4 (left), potential functions $\psi_{1,2}, \psi_{1,3}, \psi_{1,4}$ and $\psi_{1,5}$ are replaced by $\psi_{2,3,4,5} = \oplus_{x_1}(\psi_{1,2} \odot \psi_{1,3} \odot \psi_{1,4} \odot \psi_{1,4})$. The new graph is shown in Figure 4 (right).

3.2.1. Interpretation for marginalisation, maximisation and finding the mode of a distribution

When the first elimination step is applied with $\oplus = +$ and $\odot = \times$, the probability distribution defined by this new graphical model is the marginal distribution $p_{V \setminus \{i\}}(x_{V \setminus \{i\}})$ of the original distribution p (up to a constant). The complete elimination can be obtained by successively eliminating all variables in X_A. The result is a graphical model over $X_{V \setminus A}$, which specifies the marginal distribution $p_{V \setminus A}(x_{V \setminus A})$. When $A = V$, the result is a model with a single constant potential function with value Z.

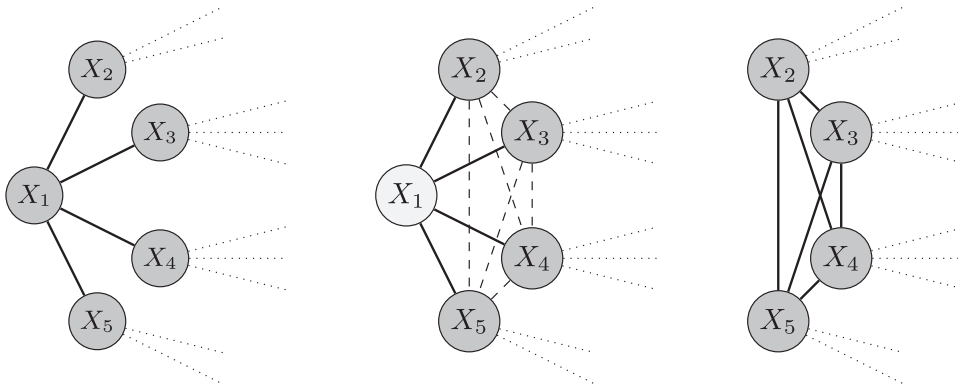

Figure 4. Elimination of variable X_1 replaces the four pairwise potential functions involving variable X_1 with a new potential ψN_1, involving the four neighbours of vertex 1 in the original graph. The new edges created between these four vertices are called fill-in edges (dashed edges in the middle figure).

If instead $\oplus$ is max, and $\odot = \times$ (or $+$ with a log transformation of the potential functions) and $A = V$, the last potential function obtained after elimination of the last variable is equal to the maximum of the non-normalised distribution. So evaluating Z or the maximal probability of a graphical model can both be performed with the same variable elimination algorithm, just changing the definition of the $\oplus$ (and $\odot$ if needed) operator(s). Lastly, if one is interested in the mode itself, an additional computation is required. The mode is actually obtained by induction: if $x^*_{V\setminus\{i\}}$ is the mode of the graphical model obtained after the elimination of the first variable, X_i, then the mode of p can be defined as $(x^*_{V\setminus\{i\}}, x^*_i)$, where x^*_i is a value in Λ_i that maximises $\odot_{B\in\mathcal{B}}\psi_B(x^*_{V\setminus\{i\}}, x_i)$. This maximisation is straightforward to derive because x_i can take only $|\Lambda_i|$ values. $x^*_{V\setminus\{i\}}$ itself is obtained by successively building the mode of the graphical model obtained after elimination of the second variable, and so on. We stress here that the procedure requires retention of the intermediary potential functions ψ_{N_i} created during the successive eliminations.

3.2.2. Complexity of the intermediary potential functions and variable elimination ordering

When eliminating a variable X_i, the task which can be computationally expensive is the computation of the intermediate ψ_{N_i}. It requires computation of the product $\odot_{B\in\mathcal{B}_i}\psi_B(x_B)$ of several potential functions for all elements of $\Lambda_{N_i\cup\{i\}}$, the state space of $X_{N_i\cup\{i\}}$. The time and space complexity of the operation are entirely determined by the cardinality $|N_i|$ of the set of indices in N_i. If $K = \max_{j\in V}|\Lambda_j|$, the time complexity (i.e. number of elementary operations performed) is in $O(K^{|N_i|+1})$ and space complexity (i.e. memory space needed) is in $O(K^{|N_i|})$. Complexity is therefore exponential in $|N_i|$, the number of neighbours of the eliminated variable in the current graphical model. The total complexity of the variable elimination is then exponential in the maximum cardinality $|N_i|$ over all successive eliminations. However note that it is linear in n, which means that a large n is not necessarily a problem for having access to exact inference. Because the graphical model changes at each elimination step, this number usually depends on the order in which variables are eliminated.

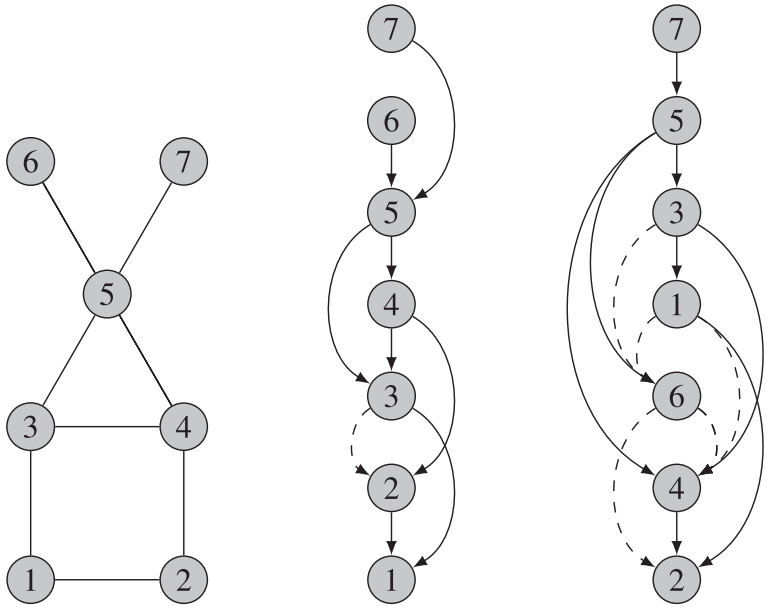

Figure 5. A graph and two elimination orders. Left, the graph; middle, induced graph associated with the elimination order (7, 6, 5, 4, 3, 2, 1). Vertices are eliminated from the first to the last element of the list. The maximum size of N_i sets created during elimination is 2 (maximum number of outgoing edges) and only one (dashed) fill-in edge is added when vertex 4 is eliminated; right, induced graph associated with the elimination order (7, 5, 3, 1, 6, 4, 2). The maximum size of N_i sets created during elimination is 3 and 5 (dashed) fill-in edges are used.

As a consequence, the prerequisite for application of variable elimination is to decide for an ordering of the elimination of the variables. As illustrated in Figure 5 two different orders can lead to two different N_i subsets. The key message is that the choice of the order is crucial. It dictates the efficiency of the variable elimination procedure. We now illustrate and formalise this intuition.

3.3. When is variable elimination efficient?

We can understand why the Viterbi algorithm is an efficient algorithm for mode evaluation in a HMC. The graph associated with an HMC is comb shaped: the hidden variables form a line and each observed variable is a leaf in the comb (see Figure 2). So it is possible to design an elimination order where the current variable to eliminate has a unique neighbour in the graphical representation of the current model: for instance, the ordering $H_T, H_{T-1}, \ldots, H_1$. By convention, the first eliminated variable is the first variable listed in this ordering (note that variables O_t do not have to be eliminated since their value is known). Following this elimination order, when eliminating a variable using $\oplus$, the resulting graphical model has one fewer vertex than the previous one and no fill-in edge. Indeed, the new potential function ψ_{N_i} is a function of a single variable since $|N_i| = 1$. The Viterbi algorithm has a space complexity of $O(TK)$ and a time complexity of $O(TK^2)$.

More generally, variable elimination is very efficient, i.e. leads to transitional N_i sets of small cardinality, on graphical models whose graph representation is a tree. More specifically,

for such graph structure, it is always possible to design an elimination order where the current variable to eliminate has only one neighbour in the graphical representation of the current model.

Another situation where variable elimination can be efficient is when the graph associated with the graphical model is chordal (any cycle of length four or more has a chord i.e., an edge connecting two non-adjacent vertices in the cycle), and when the size of the largest clique is low. The rationale for this interesting property is explained intuitively here. In Figure 4, new edges are created between neighbours of the eliminated vertex. If this neighbourhood is a clique, no new edge is added. A vertex whose neighbourhood is a clique is called a simplicial vertex. Chordal graphs have the property that there exists an elimination order of the vertices, such that every vertex during the elimination process is simplicial (Habib & Limouzy 2009). Consequently, there exists an elimination order such that no fill-in edges are created. Thus, the size of a transitional N_i's is dictated by the size of the clique formed by the neighbours of i Let us note that a tree is a chordal graph, in which all edges and only edges are cliques. Hence, for a tree, simplicial vertices are vertices of degree one. The elimination of degree one vertices on a tree is an example of simplicial elimination on a chordal graph.

For arbitrary graphs, if the maximal scope size of the intermediate ψ_{N_i} functions created during variable elimination is too large, then memory and time required for the storage and computation quickly exceed computer capacities. Depending on the chosen elimination order, this maximal scope can be reasonable from a computational point of view, or too large. So again, the choice of the elimination order is crucial. In the case of CHMMs, we can imagine two different elimination orders: either time slice per time slice, or chain by chain. (We omit the observed variables that are known and do not have to be eliminated.) For the first order, starting from the oriented graph of Figure 3, we first moralise it. Then, elimination of the variables H_T^i of the last time step does not add any fill-in edges. However, when eliminating variables H_{T-1}^i for $1 \leqslant i \leqslant I-1$, due to the temporal dependences between the chains, we create an intermediate potential function depending on $I+1$ variables (H_{T-1}^I and the H_{T-2}^i for all chains). Then when successively eliminating temporal slices, the maximal size of the intermediate potential functions created is $I+1$. For the second elimination order, still starting from the moralised version of the oriented graph, after eliminating all variables H_t^1 for $1 \leqslant t \leqslant T-1$, we create an intermediate potential function depending on $T+1$ variables (H_T^1 and H_t^2 for all t). Then when successively eliminating chains, the maximal size of the intermediate potential functions created is $T+1$. So depending on the values of I and T, we will not select the same elimination order.

3.4. The treewidth to characterise variable elimination complexity

The lowest complexity achievable when performing variable elimination is characterised by a parameter called the treewidth of the graph associated with the original graphical model. This concept has been repeatedly discovered and redefined. The treewidth of a graph is sometimes called its induced width (Dechter & Pearl 1988), its minimum front size (Liu 1992), its k-tree number (Arnborg 1985) its dimension (Bertelé & Brioshi 1972), and is also equal to the min-max clique number of G minus one (Arnborg 1985) to name a few. The treewidth is also a key notion in the theory of graph minors (Robertson & Seymour 1986; Lovász 2005).

We insist here on two definitions. The first one (Bodlaender 1993) relies on the notion of the induced graph (see Definition 1 below). It highlights the close relationship between fill-in edges and the intermediate N_i sets created during variable elimination. The second one (Robertson & Seymour 1986; Bodlaender 1993) is the most commonly used characterisation of the treewidth using the so-called tree decompositions, also known as junction trees, which are key tools to derive variable elimination algorithms. It underlies the block-by-block elimination procedure described in Section 3.5.

Definition 1. Let $G=(V,E)$ be a graph defined by a set of vertices indexed on V and a set E of edges. Given an ordering π of the vertices of G, the induced graph G_π^{ind} is defined in a constructive way as follows. First, G and G_π^{ind} have same vertices. Then for each edge in E an oriented edge is added in G_π^{ind} going from the first of the two nodes according to π toward the second. Then each vertex i of V is considered one after the other following the order defined by π. When vertex i is treated, an oriented edge is created between all pairs of neighbours of i in G that follow i in the ordering defined by π. Again the edge is going from the first of the two nodes according to π toward the second.

The induced graph G_π^{ind} is also called the fill graph of G, and the process of computing it is sometimes referred to as 'playing the elimination game' on G, as it just simulates elimination on G using the variable ordering π (see an example on Figure 5). This graph is chordal (Vandenberghe & Andersen 2014). It is known that every chordal graph G has at least one vertex ordering π such that $G_\pi^{\text{ind}}=G$ (omitting the fact that edges of G_π^{ind} are directed), called a perfect elimination ordering (Fulkerson & Gross 1965).

The second notion that enables us to define the treewidth is the notion of tree decomposition. Intuitively, a tree decomposition of a graph G organises the vertices of G in clusters of vertices which are linked by edges such that the graph obtained is a tree. Specific constraints on the way vertices of G are associated with clusters in the decomposition tree are required. These constraints ensure that the resulting tree decomposition has properties useful for building variable elimination algorithms.

Definition 2. Given a graph $G=(V,E)$, a tree decomposition of G, T, is a tree $(\mathcal{C},E_T)$, where $\mathcal{C}=\{C_1,\ldots,C_l\}$ is a family of subsets of V (called clusters), and E_T is a set of edges between the subsets C_i, satisfying the following properties:

- The union of all clusters C_k equals V (each vertex of G is associated with at least one vertex of T).
- For every edge (i,j) in E, there is at least one cluster C_k that contains both i and j.
- If clusters C_k and C_l both contain a vertex i of G, then all clusters C_s of T in the (unique) path between C_k and C_l contain i as well: clusters containing vertex i form a connected subset of T. This is known as the running intersection property.

The concept of tree decomposition is illustrated in Figure 6.

Definition 3. The following two definitions of the treewidth Derived, respectively, from the notion of induced graph, and from that of tree decomposition are equivalent:

- The treewidth $\mathcal{TW}_\pi(G)$ of a graph G for the ordering π is the maximum number of outgoing edges of a vertex in the induced graph G_π^{ind}. The treewidth $\mathcal{TW}(G)$ of a graph G is the minimum treewidth over all possible orderings π.

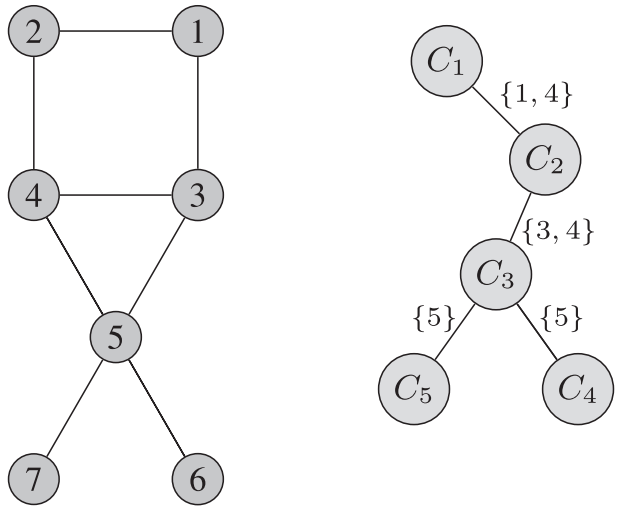

Figure 6. Left: graphical representation of a graphical model. Right: tree decomposition over clusters $C_1 = \{1,2,4\}, C_2 = \{1,3,4\}, C_3 = \{3,4,5\}, C_4 = \{5,6\}$ and $C_5 = \{5,7\}$. Each edge between two clusters is labelled by their shared variables.

- The width of a tree decomposition $(\mathcal{C}, E_T)$ is the size of the largest $C_i \in \mathcal{C}$ minus 1, and the treewidth $\mathcal{TW}(G)$ of a graph is the minimum width among all its tree decompositions.

It is not trivial to establish the equivalence (see Meseguer, Rossi & Schiex 2006, Chapter 7, and Schiex 1999). The term $\mathcal{TW}_\pi(G)$ is exactly the cardinality of the largest set N_i created during variable elimination with elimination order π. For example, in Figure 5, the middle and right graphs are the two induced graphs for two different orderings and $\mathcal{TW}_\pi(G)$ is equal to 2 with the first ordering and to 3 with the second. It is easy to see that in this example $\mathcal{TW}(G) = 2$. The treewidth of the graph of the HMC model, and of any tree is equal to 1.

It has been established that finding a minimum treewidth ordering π for a graph G, finding a minimum treewidth tree decomposition, or computing the treewidth of a graph are of equivalent complexity. For an arbitrary graph, computing the treewidth is not an easy task. Section 4 is dedicated to this question, from both a theoretical and a practical point of view.

The treewidth is therefore a key indicator to answer the driving subject of this review: will variable elimination be efficient for a given graphical model? For instance, the principle of variable elimination was applied to the exact computation of the normalising constant of an MRF on a small r by c lattice in Reeves & Pettitt (2004). For this regular graph, it is known that the treewidth is equal to $\min(r, c)$. So exact computation through variable elimination is possible for lattices with a small value for $\min(r, c)$ (even if $\max(r, c)$ is large). It is however well beyond computer capacities for real challenging problems in image analysis. In this case variable elimination can still be used, to define heuristic computational solutions, as in the following examples. In a Bayesian setting, in Friel & Rue (2007) the authors proposed an estimation of the a posteriori distribution of the parameter of a binary Hidden Markov Random Field (HMRF) without resorting to MCMC, by combining the work of Reeves

& Pettitt (2004) and an approximation of the dependencies. Still for HMRF, Friel *et al.* (2009) proposed an algorithm for computing the likelihood which relies on the merging of exact computations on small sub-lattices of the original lattice. More recently, Austad & Tjelmeland (2017) applied variable elimination on an approximation of the pseudo Boolean expression of a MRF distribution.

3.5. Tree decomposition and block by block elimination

Given a graphical model and a tree decomposition of its graph, a possible alternative to solve counting or optimisation tasks is to eliminate variables by successive blocks instead of one after the other. To do so, the block by block elimination procedure (Bertelé & Brioshi 1972) relies on the tree decomposition characterisation of the treewidth. The underlying idea is to apply the variable elimination procedure on the tree decomposition, eliminating one cluster of the tree at each step. First a root cluster $C_r \in \mathcal{C}$ is chosen and used to define an order of elimination of the clusters, by progressing from the leaves toward the root. Every eliminated cluster corresponds to a leaf of the current intermediate tree. Then each potential function ψ_B is assigned to the cluster C_i in $\mathcal{C}$ such that $B \subset C_i$, which is the closest to the root. Such a cluster always exists otherwise either the running intersection property would not be satisfied or the graph of the decomposition would not be a tree. More precisely, the procedure starts with the elimination of any leaf cluster C_i of T, with parent C_j in T. Let us denote $\mathcal{B}(C_i) = \{B \in \mathcal{B}, \psi_B \text{ assigned to } C_i\}$. Here again, commutativity and distributivity are used to rewrite expression (4) (with $A = V$) as follows:

$$\bigoplus_{\boldsymbol{x}} \bigodot_{B \in \mathcal{B}} \psi_B = \bigoplus_{x_{V \setminus (C_i \setminus C_j)}} \Bigg[\bigodot_{B \in \mathcal{B} \setminus \mathcal{B}(C_i)} \psi_B \odot \underbrace{\Big(\bigoplus_{x_{C_i \setminus C_j}} \bigodot_{B \in \mathcal{B}(C_i)} \psi_B \Big)}_{\text{New potential function}} \Bigg].$$

Note that only variables with indices in $C_i \setminus C_j \equiv C_i \cap (V \setminus C_j)$ are eliminated, even if it is common to say that the cluster has been eliminated. For instance, in the example depicted in Figure 6, if the first eliminated cluster is C_1, the new potential function is $\oplus_{x_2} \psi_{1,2}(x_1, x_2)\psi_{2,4}(x_2, x_4)$, which depends only on variables X_1 and X_4. Cluster elimination continues until no cluster is left. The point of this procedure is that the intermediate potential function created after each cluster elimination may have a scope much smaller than the treewidth, leading to better space complexity (Bertelé & Brioshi 1972, Chapter 4). However, the time complexity is increased.

In summary, the lowest achievable complexity when performing variable elimination is reached for elimination orders when the cardinalities of the intermediate sets N_i are smaller or equal to the treewidth of G. This treewidth can be determined by considering cluster sizes in tree decompositions of G. Furthermore, any tree decomposition T can be used to build an elimination order and vice versa. Indeed, an elimination order can be defined by using a cluster elimination order based on T, and by choosing an arbitrary order to eliminate variables with indices in the subsets $C_i \setminus C_j$. Conversely, it is easy to build a tree decomposition from a given vertex ordering π. Since the induced graph G_π^{ind} is chordal, its maximum cliques can be identified in polynomial time. Each such clique defines a cluster C_i of the tree decomposition. Edges of T can be identified as the edges of any minimum spanning tree in the graph with vertices C_i and edges (C_i, C_j) weigthed by $|C_i \cap C_j|$.

3.5.1. Deterministic graphical models

To our knowledge, the notion of treewidth and its properties were first identified in combinatorial optimisation in Bertelé & Brioshi (1972). It was then coined 'dimension', a graph parameter which was later shown to be equivalent to the treewidth (Bodlaender 1998). Variable elimination itself is related to the Fourier-Motzkin elimination (Fourier 1827), a variable elimination algorithm which benefits from the linearity of the handled formulas. Variable elimination has been repeatedly rediscovered, as non-serial dynamic programming (Bertelé & Brioshi 1972), in the David-Putnam procedure for Boolean satisfiability problems, SAT (Davis & Putnam 1960), as Bucket elimination for the CSP and WCSP (Dechter 1999), in the Viterbi and Forward-Backward algorithms for HMM (Rabiner 1989) and many more.

There exists other situations where the choice of an elimination order has a deep impact on the complexity of the computations as in the Gaussian elimination scheme for a system of linear equations, or Choleski factorisation of very large sparse matrices, In these cases, the equivalence between elimination and decomposition was also used (see Bodlaender *et al.* 1995), and we recover the notion of undesirable fill-in in Cholesky factorisation, corresponding to non-zero elements created during the decomposition.

4. Treewidth approximation for exact inference

As already mentioned, the complexity of the counting and the optimisation tasks on graphical models is strongly linked to the treewidth $\mathcal{TW}(G)$ of the underlying graph G. If one could guess (one of) the optimal vertex ordering(s), π^*, leading to $\mathcal{TW}_{\pi^*}(G) = \mathcal{TW}(G)$, then, one would be able to achieve the 'optimal complexity' $O(K^{\mathcal{TW}(G)}n)$ for obtaining exact solutions for these tasks; we recall that K is the maximal domain size of a variable in the graphical model. However, the first obstacle to overcome is that the treewidth of a given graph cannot be evaluated easily: the treewidth computation problem is known to be NP-hard (Arnborg, Corneil & Proskurowski 1987). If one has to spend more time on finding an optimal vertex ordering than on computing probabilities in the underlying graphical model, the utility of exact treewidth computation appears limited. Therefore, an alternative line of search is to look for algorithms computing a vertex ordering π leading to a suboptimal width, $\mathcal{TW}_{\pi}(G) \geqslant \mathcal{TW}(G)$, but more efficient in terms of computational time. In the following, we describe and empirically compare heuristics which simultaneously provide a vertex ordering and an upper bound of the treewidth. Performing inference relying on this ordering is still exact. It is not optimal in terms of time complexity, but, on some problems, the inference can still be performed in reasonable time.

A broad class of heuristic approaches is that of greedy algorithms (Bodlaender & Koster 2010). They use the same iterative approach as the variable elimination algorithm (Section 3) except that they only manipulate the graph structure. They do not perform any actual combination/elimination computation. Starting from an empty vertex ordering and an initial graph G, they repeatedly select the next vertex to add in the ordering by locally optimising one of the following criteria:

- select a vertex with minimum degree in the current graph; or
- select a vertex with minimum number of fill-in edges in the current graph.

After each vertex selection, the current graph is modified by removing the selected vertex and making a clique on its neighbours. The new edges added by this clique creation

are fill-in edges. A vertex with no fill-in edges is a simplicial vertex (see Section 3.3). Fast implementations of minimum degree algorithms have been developed, see e.g., AMD (Amestoy, Davis & Duff 1996) with time complexity in $O(nm)$ (Heggernes *et al.* 2001) for an input graph G with n vertices and m edges. The minimum fill-in heuristics tend to be slower to compute but yield slightly better treewidth approximations in practice. Moreover, if a perfect elimination ordering (i.e., adding no fill-in edges) exists, this heuristic will find it. Thus, it recognises chordal graphs, and it returns the optimal treewidth in this particular case. This can be easily established from results in Bodlaender, Koster & Eijkhof 2005.

Notice that there exists linear time $O(n+m)$ algorithms to detect chordal graphs as the maximum cardinality search (MCS) greedy algorithm (Tarjan & Yannakakis 1984). MCS builds an elimination order based on the cardinality of the already processed neighbours. However, the treewidth approximation they return is often worse in practice than the previous heuristic approaches.

A simple way to improve the treewidth bound found by these greedy algorithms is to choose between candidate vertices with the same value for the selected criterion by using a second criterion, such as minimum fill-in first and then maximum degree, or to choose at random and to iterate on the resulting randomised algorithms as done in Kask *et al.* (2011).

We compared the mean treewidth upper bound found by these four approaches (minimum degree, minimum fill-in, MCS and randomised iterative minimum fill-in) on a set of five WCSP and MRF benchmarks used as combinatorial optimisation problems in various solver competitions. ParityLearning is an optimisation variant of the minimal disagreement parity CSP problem originally contributed to the DIMACS benchmark and used in the MiniZinc challenge (Optimization Research Group 2012). Linkage is a genetic linkage analysis benchmark (Elidan & Globerson 2010). GeomSurf and SceneDecomp are, respectively, geometric surface labelling and scene decomposition problems in computer vision (Andres, Beier & Kappes 2013). For each problem, it is possible to vary the number of vertices and potential functions. The number of instances per problem as well as their mean characteristics are given in Table 2. Results are reported in Figure 7 (Top). Note that here the box plots do not have the usual interpretation since different instances of the same problem do not have the same characteristics. They show the potential range of variation of the treewidth within a type of problem. The randomised iterative minimum fill-in algorithm used a maximum of 30,000 iterations or 180 seconds (respectively, 10,000 iterations and 60 seconds for ParityLearning and Linkage), compared to a maximum of 0.37 seconds used by the non-iterative approaches. The minimum fill-in algorithm (using maximum degree for breaking ties) performed better

Table 2. Characteristics of the five optimisation problems used as benchmark. For a given problem, several instances are available, corresponding to different numbers of variables (equal to the number of vertices in the underlying graph) and different numbers of potential functions.

Problem Type/Name	Number of instances	Mean no. of vertices	Mean no. of potential functions
CSP/ParityLearning	7	659	1246
MRF/Linkage	22	917	1560
MRF/GeomSurf-3	300	505	2140
MRF/GeomSurf-7	300	505	2140
MRF/SceneDecomp	715	183	672

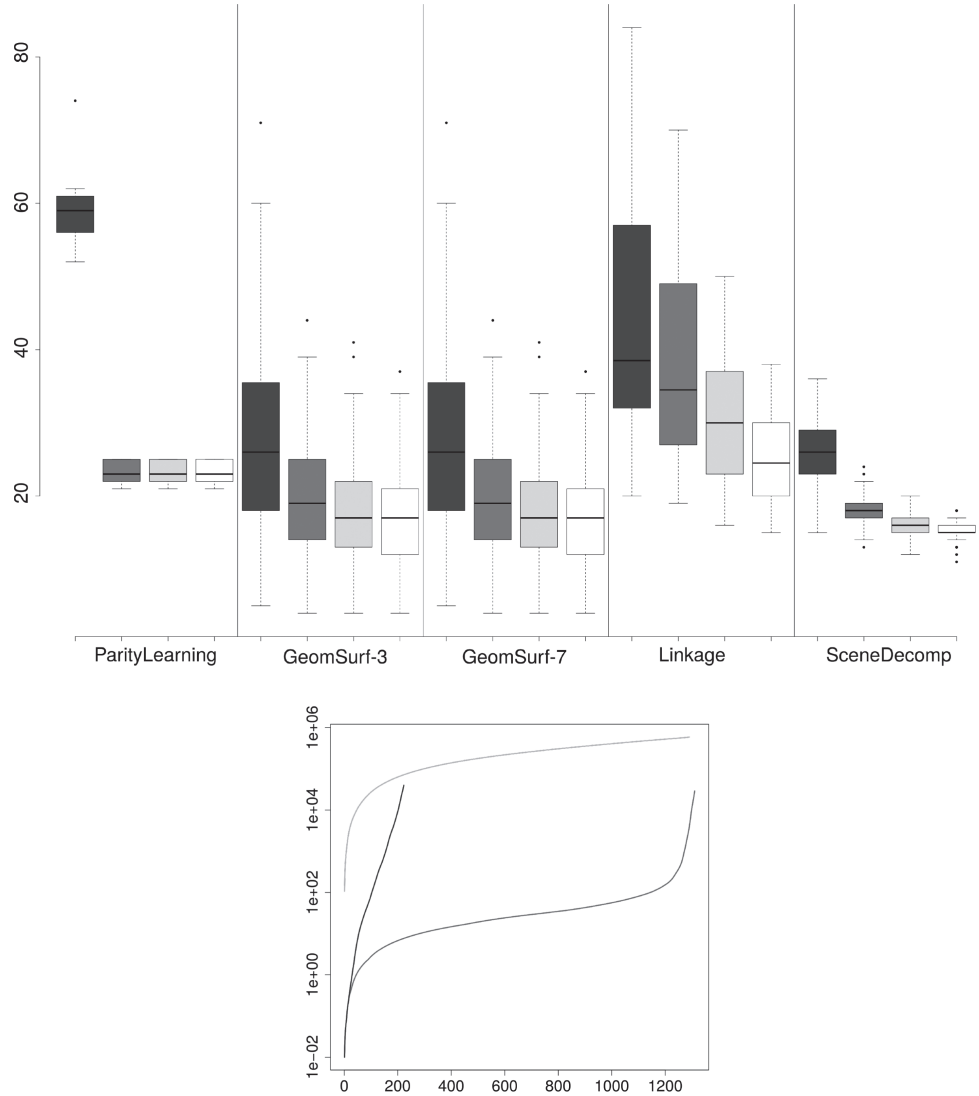

Figure 7. Top: Comparison of treewidth upper bounds provided by MCS, minimum degree, minimum fill-in and randomised iterative minimum fill-in (from dark grey to white) for the 5 categories of problems. Bottom: Mode evaluation by three exact methods exploiting minimum fill-in ordering or its randomised iterative version. Number of instances solved (x-axis) within a given CPU time in seconds (log10 scale y-axis) of ELIM (black), BTD (dark grey), and AND/OR SEARCH (gray).

than the other greedy approaches. Its randomised iterative version offers slightly improved performance, at the price of some computation time.

On the same benchmark, we then compared three exact methods for the task of mode evaluation that exploit either minimum fill-in ordering or its randomised iterative version: variable elimination (ELIM); BTD (de Givry, Schiex & Verfaillie 2006); and AND/OR Search (Marinescu & Dechter 2006). Elim and BTD exploit the minimum fill-in ordering while AND/OR Search uses its randomised iterative version. In addition, BTD and AND/OR Search

exploit a tree decomposition during a Depth First Branch and Bound method in order to get a good trade-off between memory space and search effort. Just like variable elimination, they have a worst-case time complexity exponential in the treewidth. All methods were allocated a maximum of 1 hour and 4 GB of RAM on an AMD Operon 6176 at 2.3 GHz. The results are reported in Figure 7 (Bottom), and show that BTD was able to solve more problems than the two other methods for fixed CPU time. However, the best performing method heavily depends on the problem category. For ParityLearning, ELIM was the fastest method, but it ran out of memory on 83% of the total set of instances, while BTD (resp. AND/OR Search) used less than 1.7 GB (resp. 4GB). The randomised iterative minimum fill-in heuristic used by AND/OR Search in preprocessing consumed a fixed amount of time (≈ 180 seconds, included in the CPU time measurements) larger than the cost of a simple minimum fill-in heuristics run. BTD was faster than AND/OR Search to solve most of the instances except on two problem categories (ParityLearning and Linkage).

To perform this comparison, we ran the following implementation of each method. The version of ELIM was the one implemented in the combinatorial optimisation solver toolbar 2.3 (options `-i -T3`, available at `mulcyber.toulouse.inra.fr/projects/toolbar`). The version of BTD was the one implemented in the combinatorial optimisation solver toulbar2 0.9.7 (options `-B=1 -O=-3 -nopre`). toulbar2 is available at `www7.inra.fr/mia/T/toulbar2`. This software won the UAI 2010 (Elidan & Globerson 2010) and 2014 (Gogate 2014) Inference Competitions on the MAP task. AND/OR Search was the version implemented in the open-source version 1.1.2 of daoopt (Otten *et al.* 2012) (options `-y -i 35 --slsX=20 --slsT=10 --lds 1 -m 4000 -t 30000 --orderTime=180` for benchmarks from computer vision, and `-y -i 25 --slsX=10 --slsT=6 --lds 1 -m 4000 -t 10000 --orderTime=60` for the other benchmarks) which won the Probabilistic Inference Challenge 2011 (Elidan & Globerson 2011), albeit with a different closed-source version (Otten *et al.* 2012).

5. From variable elimination to message passing

On tree-structured graphical models, message passing algorithms extend the variable elimination algorithm by efficiently computing every marginal (or max-marginal) simultaneously, when variable elimination only computes one. On general graphical models, message passing algorithms can still be applied. They either provide approximate results efficiently, or have an exponential running cost.

We also present a less classical interpretation of message passing algorithms: it may be conceptually interesting to view these algorithms as performing a re-parametrisation of the original graphical model, i.e. a rewriting of the potentials without modifying the joint distribution. Instead of producing external messages, the re-parametrisation produces an equivalent MRF, where marginals can be easily accessed, and which can be better adapted than the original one for initialising further processing.

5.1. Message passing and belief propagation

5.1.1. Message passing when the graph is a tree

Message passing algorithms over trees (Pearl 1988) can be described as an extension of variable elimination, where the marginals or max-marginals of all variables are computed

in a double pass of the algorithm. We depict the principle here when G is a tree first and for marginal computation. At the beginning of the first pass (the forward pass), each leaf i is marked as 'processed' and all other variables are 'unprocessed'. Then each leaf is successively visited. The new potential ψ_{N_i} is considered as a 'message' sent from i to its unique neighbour in the tree, say j. The message, denoted as $\mu_{i\to j}$ is a potential function over X_j only (scope of size 1). When vertex j has received the messages from all leaf vertices neighbour with it, it is marked as processed. It can then send its own messages upward (combining messages received and remaining potential functions involving X_j), to its unique unmarked neighbour.

When only one vertex remains unmarked (r, the root of the tree), the product of all the messages received by r and possibly an original potential function involving only X_r will be equal to the unnormalised marginal distribution of X_r. This results directly from the fact that the operations performed in this forward pass of message passing are equivalent to variable elimination.

To compute the marginal of another variable X_k, one can redirect the tree using k as a new root and apply the same procedure from the start. However, k splits the tree into two subtrees, one containing r, T_r, and one not, $T_{\bar{r}}$. The messages sent from leaves to k in $T_{\bar{r}}$ are the same as those sent when computing the marginal of X_r. The message sent from leaves to r in T_r likewise. The second pass (backward or downward pass) of the message passing algorithm exploits the fact that messages are shared between several marginal computations, to organise all these computations in a clever way, so that in order to compute marginals of all variables, it is enough in the second pass to send messages from the root towards the leaf. Then the unnormalised marginal at a particular vertex is computed by multiplying downward messages with upward messages arriving at that vertex. One application is the well-known forward–backward algorithm (Rabiner 1989).

Formally, in the message passing algorithm for marginals evaluation over a tree (V, E), messages $\mu_{i\to j}$ are defined for each edge $(i,j)\in E$ in a leaves-to-root-to-leaves order; there are $2|E|$ such messages, one for each edge direction. Messages $\mu_{i\to j}$ are functions of x_j, which are computed iteratively, by the following algorithm:

1. Messages leaving the leaves of the tree are computed: for each $i\in V$, where i is a leaf of the tree, and for j the unique parent of i, for all $x_i, \in \Lambda_i$:

$$\mu_{i\to j}(x_j) \leftarrow \sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i).$$

   This corresponds to computing ψ_{N_i} in (5) with $\odot$ being the sum, $\oplus$ being the product, and $\mathcal{B}_i = \{i\}\cup\{i,j\}$.
   All leaves are marked as as processed.
2. Messages are sent upward through all edges. Message updates are performed iteratively, from marked nodes i to their only unmarked neighbour j through edge $(i,j)\in E$. Message updates take the following form for all $x_j \in \Lambda_j$:

$$\mu_{i\to j}(x_j) \leftarrow \frac{1}{K}\sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i) \prod_{k\neq j,(k,i)\in E} \mu_{k\to i}(x_i), \tag{6}$$

   where $K = \sum_{x_j}\sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i)\prod_{k\neq j,(k,i)\in E}\mu_{k\to i}(x_i)$. In theory it is not necessary to normalise the messages, but this can be useful to avoid numerical problems. In

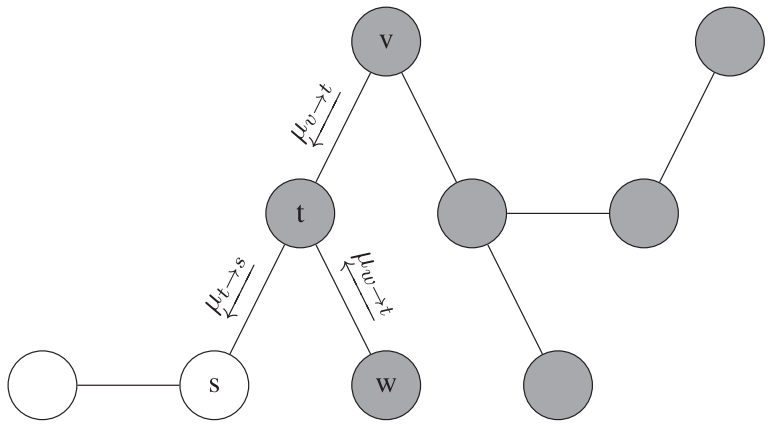

Figure 8. Example of message update on a tree. In this example, nodes t, v and w are marked, while node s is still unmarked. $\mu_{t\to s}$ is a function of all the incoming messages to node t, except $\mu_{s\to t}$.

message $\mu_{i\to j}$, the product over messages received by i corresponds to the result of the elimination of all variables associated with a vertex below i in the tree. Then with the sum over x_i, variable X_i is eliminated.
Then vertex j is marked as processed. See Figure 8 for an illustration.

3. Messages are sent downward (from root to leaves). This second phase of message updates takes the following form:

   • Unmark root node.
   • While there remains a marked node, send a message of same expression as (6) from an unmarked vertex i to j, one of its marked neighbours, and unmark the corresponding neighbour.

4. After the three steps above, messages have been transmitted through all edges in both directions. Vertex i has received upward messages eliminating variables below it in the tree and downward messages eliminating all the other variables. So the product of the messages received by i is a function only of x_i. Finally, the marginal distribution of X_i is computed as follows:

$$p_i(x_i) \leftarrow \frac{1}{K_i}\psi_i(x_i) \prod_{j,(j,i)\in E} \mu_{j\to i}(x_i).$$

   An advantage of this algorithm is that it also enables the computation of pair marginals for two vertices i and j linked by an edge, as follows:

$$p_{ij}(x_i, x_j) \leftarrow \frac{1}{K_{ij}}\psi_{ij}(x_i, x_j) \prod_{k\neq j,(k,i)\in E} \mu_{k\to i}(x_i) \prod_{l\neq i,(l,j)\in E} \mu_{l\to j}(x_j),$$

   where K_i and K_{ij} are suitable normalising constants.

Max-product and max–sum algorithms can be equivalently defined on a tree, for exact computation of the max-marginal of a joint distribution or its logarithm (see

Chapter 8 of Bishop 2006). In algebraic language, updates as defined by (6) take the general form:

$$\forall x_j \in \Lambda_j, \mu_{i\to j}(x_j) = \bigoplus_{x_i'} \psi_{ij}(x_i', x_j)\psi_i(x_i') \bigodot_{k\neq j,(k,i)\in E} \mu_{k\to i}(x_i').$$

5.1.2 Message passing when the factor graph is a tree

In some cases, the graph underlying the model may not be a tree, but the corresponding factor graph can be a tree, with factors potentially involving more than two variables (see Figure 9 for an example). In these cases, message passing algorithms can still be defined, and they lead to exact marginal value computations (or of max-marginals). However, their complexity becomes exponential in the size of the largest factor minus 1.

The message passing algorithm on a tree structured factor graph exploits the same idea of shared messages as in the case of tree structured graphical models, except that two different kinds of messages are computed:

- Factor-to-variable messages: messages from a factor B (we identify the factor with the subset B of the potential function ψ_B it represents) towards a variable i, $\mu_{B\to i}(x_i)$.
- Variable-to-factor messages: message from a variable i towards a factor B, $v_{i\to B}(x_i)$.

These are updated in a leaf-to-root direction and then backward, as above, but two different updating rules are used instead of (6): for all $x_i \in \Lambda_i$

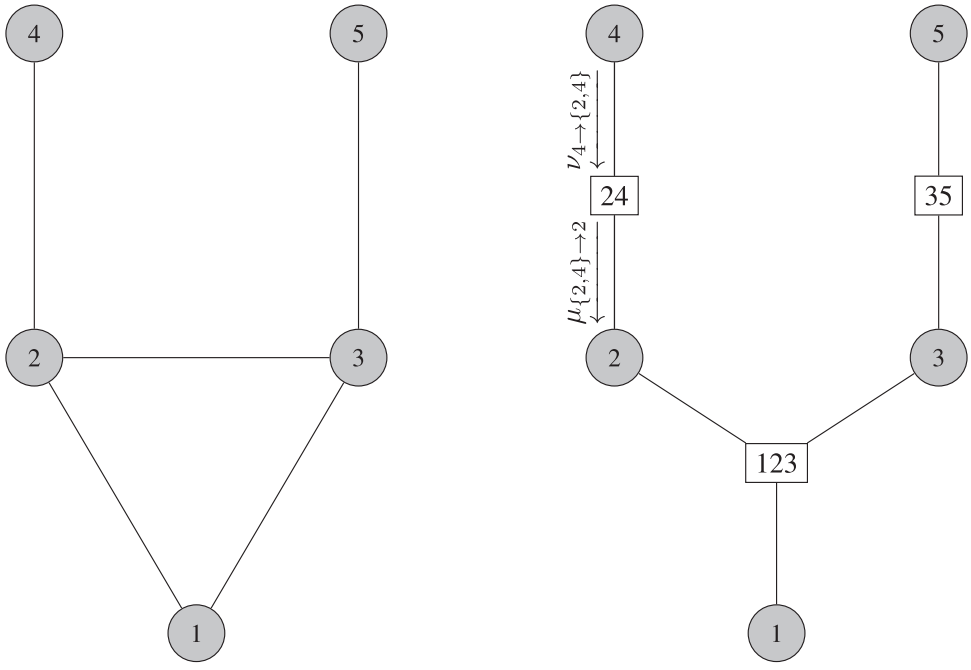

Figure 9. Left: Graphical representation of a graphical model with a potentiel function involving variables X_1, X_2 and X_3. This is not a tree. Right: Corresponding factor graph, which is a tree. For applying message passing, the root is variable 1, while variables 4 and 5 are leaves. For the left branch, the first messages sent is $v_{4\to\{2,4\}}(x_4)$ followed by $\mu_{\{2,4\}\to 4}(x_2)$.

$$\mu_{B\to i}(x_i) \leftarrow \sum_{x_{B\setminus i}} \left(\psi_B(x_B) \prod_{j\in B\setminus i} v_{j\to B}(x_j) \right),$$
$$v_{i\to B}(x_i) \leftarrow \prod_{B'\neq B, i\in B'} \mu_{B'\to i}(x_i).$$

Then, the marginal probabilities are obtained by local marginalisation, as in Step 4 of the algorithm of Subsection 5.1.1 above.

$$p_i(x_i) \leftarrow \frac{1}{K_i}\psi_i(x_i) \prod_{B, i\in B} \mu_{B\to i}(x_i), \forall x_i \in \Lambda_i, \tag{7}$$

where K_i is again a normalising constant.

5.2. When the factor graph is not a tree

When the factor graph of the graphical model is not a tree, the two-pass message passing algorithm can no longer be applied directly as is because of the loops. Yet, for general graphical models, this message passing approach can be generalised in two different ways.

- A tree decomposition can be computed, as previously discussed in Section 3.5. Message passing can then be applied on the resulting cluster tree, handling each cluster as a cross-product of variables following a block-by-block approach. This yields an exact algorithm, for which computations can be expensive (exponential in the treewidth) and space intensive (exponential in the separator size). A typical example of such algorithms is the algebraic exact message passing algorithm (Shafer & Shenoy 1988; Shenoy & Shafer 1990).
- Alternatively, the Loopy Belief Propagation algorithm, LBP (Frey & MacKay 1998) is another extension of message passing in which messages updates are repeated, in arbitrary order through all edges (possibly many times through each edge), until a termination condition is met. The algorithm returns approximation of the marginal probabilities (over variables and pairs of variables). The quality of the approximation and the convergence to steady-state messages is not guaranteed, hence the importance of the termination condition. However, it has been observed that LBP often provides good estimates of the marginals in practice. Still, implementing LBP must be made with care, since the scheduling of messages plays an important role with regard to speed (see Koller & Friedman 2009, p. 408). A deeper analysis of the LBP algorithm is postponed to Section 6.

5.3. Message passing and re-parametrisation

It is possible to use message passing as a re-parametrisation technique. In this case, the computed messages are directly used to reformulate the original graphical model in a new equivalent graphical model with the same graphical structure. By ‘equivalent’ we mean that the potential functions are not the same but they define the same joint distribution as the original graphical model.

Several methods for re-parametrisation have been proposed both in the field of probabilistic graphical models (Koller & Friedman 2009, Chapters 10 and 13) or in the field

of deterministic graphical models (Cooper *et al.* 2010). They all share the same advantage: the re-parameterised formulation can be computed to satisfy precise requirements. It can be designed so that the re-parameterised potential functions contains some information of interest (marginal distributions on singletons, on pairs $p_i(x_i)$, max-marginals $p^*(x_i)$, or their approximation). It can also be optimised in order to tighten a bound on the probability of a mode (or MAP) assignment (Schiex 2000; Kolmogorov 2006; Cooper *et al.* 2010; Huang & Koller 2013) or on the partition function (Wainwright, Jaakkola & Willsky 2005; Liu & Ihler 2011; Viricel *et al.* 2016). Originally naive bounds can be tightened into non-naive ones by re-parametrisation. An additional advantage of the re-parametrised distribution is in the context of incremental updates, where we have to perform inference based on the observation of some of the variables, and new observations (new evidence) are introduced incrementally. Since the re-parameterised model already includes the result of previous inferences, it is more interesting (in terms of number of messages to send) to perform the updated inference when starting with this expression of the joint distribution that with the original one (Koller & Friedman 2009, Chapter 10).

The idea behind re-parametrisation is conceptually very simple: when a message $\mu_{i\to j}$ is computed, instead of keeping it as a message, it is possible to combine any potential function involving X_j with $\mu_{i\to j}$, using $\odot$. To preserve the joint distribution defined by the original graphical model, we need to divide another potential function involving X_j by the same message $\mu_{i\to j}$ using the inverse of $\odot$.

5.3.1. Example of computation of the max-marginals.

We illustrate here how re-parametrisation can be exploited to directly extract all (unnormalised) max-marginals $p^*(x_i)$ from the order 1 potentials of the new model. In this case ψ_{ij} is divided by $\mu_{i\to j}$, while ψ_j is multiplied by $\mu_{i\to j}$. The same procedure can be run by replacing max by $+$ in the message definition to obtain all singleton marginals $p(x_i)$ instead.

Let us consider a graphical model with 3 binary variables. The potential functions defining the graphical model are:

$$\psi_1(x_1)=(3,1), \quad \psi_2(x_2)=(2,6), \quad \psi_3(x_3)=(3,4)$$

$$\psi_{12}(x_1,x_2)=\begin{pmatrix}3 & 2\\ 5 & 4\end{pmatrix}, \ \psi_{23}(x_2,x_3)=\begin{pmatrix}4 & 8\\ 4 & 1\end{pmatrix}.$$

Since the graph of the model is a single path and is thus tree-structured, we just need two passes of messages. We use vertex 2 as the root. The first messages, from the leaves to the root, are:

$$\mu_{1\to 2}(x_2)=\max_{x_1}\psi_1(x_1)\psi_{12}(x_1,x_2),$$

$$\mu_{3\to 2}(x_2)=\max_{x_3}\psi_3(x_3)\psi_{23}(x_2,x_3).$$

We obtain

$$\mu_{1\to 2}(0)=\max(3\times 3,1\times 2)=9, \quad \mu_{1\to 2}(1)=\max(3\times 2,1\times 4)=6,$$

$$\mu_{3\to 2}(0)=\max(3\times 4,4\times 8)=32, \quad \mu_{3\to 2}(1)=\max(3\times 4,4\times 1)=12.$$

Potentials ψ_{12} and ψ_{23} are divided respectively by $\mu_{1\to 2}$ and $\mu_{3\to 2}$, while ψ_2 is multiplied by these two same messages. For instance

$$\psi'_2(0) = \psi_2(0)\mu_{1\to 2}(0)\mu_{3\to 2}(0) = 2\times 9\times 32 = 576,$$

$$\psi'_2(1) = \psi_2(1)\mu_{1\to 2}(1)\mu_{3\to 2}(1) = 6\times 6\times 12 = 532,$$

$$\psi'_{12}(x_1,0) = \frac{\psi_{12}(x_1,0)}{\mu_{1\to 2}(0)},\ \psi'_{12}(x_1,1) = \frac{\psi_{12}(x_1,1)}{\mu_{1\to 2}(1)}.$$

All the updated potentials are:

$$\psi'_1(x_1) = \psi_1(x_1) = (3,1),\ \psi'_2(x_2) = (576,432),\ \psi'_3(x_3) = \psi_3(x_3) = (3,4),$$

$$\psi'_{12}(x_1,x_2) = \begin{pmatrix} 3/9 & 2/6 \\ 5/9 & 4/6 \end{pmatrix} = \begin{pmatrix} 1/3 & 1/3 \\ 5/9 & 2/3 \end{pmatrix},$$

$$\psi'_{23}(x_2,x_3) = \begin{pmatrix} 4/32 & 8/32 \\ 4/12 & 1/12 \end{pmatrix} = \begin{pmatrix} 1/8 & 1/4 \\ 1/4 & 1/12 \end{pmatrix}.$$

Note that here and below we identify each pairwise potential function with its representation by a 2×2 matrix since all variables are binary. Then messages from the root towards the leaves are computed using these updated potentials:

$$\mu_{2\to 1}(x_1) = \max_{x_2} \psi'_2(x_2)\psi'_{12}(x_1,x_2) = (192,320),$$

$$\mu_{2\to 3}(x_3) = \max_{x_2} \psi'_2(x_2)\psi'_{23}(x_2,x_3) = (144,144).$$

Finally, potentials ψ'_{12} and ψ'_{23} are divided respectively by $\mu_{2\to 1}$ and $\mu_{2\to 3}$, while ψ'_1 and ψ'_3 are multiplied by $\mu_{2\to 1}$ and $\mu_{2\to 3}$ respectively, leading to the re-parameterised potentials

$$\psi''_1(x_1) = (3\times 192, 1\times 320) = (576,320), \psi''_2(x_2) = (576,432),$$

$$\psi''_3(x_3) = (3\times 144, 4\times 144) = (432,576),$$

$$\psi''_{12}(x_1,x_2) = \begin{pmatrix} \frac{1}{3\times 192} & \frac{1}{3\times 192} \\ \frac{5}{9\times 320} & \frac{2}{3\times 320} \end{pmatrix} = \begin{pmatrix} \frac{1}{576} & \frac{1}{576} \\ \frac{1}{576} & \frac{1}{480} \end{pmatrix},$$

$$\psi''_{23}(x_2,x_3) = \begin{pmatrix} \frac{1}{8\times 144} & \frac{1}{4\times 144} \\ \frac{1}{3\times 144} & \frac{1}{12\times 144} \end{pmatrix} = \begin{pmatrix} \frac{1}{1152} & \frac{1}{576} \\ \frac{1}{432} & \frac{1}{1728} \end{pmatrix}.$$

Then we can directly read the (unnormalised) max-marginal from the singleton potentials. For instance $\max_{x_2,x_3} \psi_1(0)\psi_2(x_2)\psi_3(x_3)\psi_{12}(0,x_2)\psi_{23}(x_2,x_3) = 576 = \psi''(0)$.

We can check that the original graphical model and the re-parameterised one define the same joint distribution by comparing to the (unnormalised) probabilities of each possible state (see Table 3).

Table 3. The unnormalised probabilities of the eight possible states in the original and re-parameterised models. One can check that the re-parameterised version describes the same joint distribution than the original one.

x_1	x_2	x_3	Original		Reparameterised
			$\psi_1\ \psi_2\ \psi_3\ \psi_{12}\ \psi_{23}$		$\psi''_1\ \psi''_2\ \psi''_3\ \psi''_{12}\ \psi''_{23}$
0	0	0	$3\times2\times3\times3\times4$	$=216=$	$576\times576\times432\times1/576\times1/1152$
0	0	1	$3\times2\times4\times3\times8$	$=576=$	$576\times576\times576\times1/576\times1/576$
0	1	0	$3\times6\times3\times2\times4$	$=432=$	$576\times432\times432\times1/576\times1/432$
0	1	1	$3\times6\times4\times2\times1$	$=144=$	$576\times432\times576\times1/576\times1/1728$
1	0	0	$1\times2\times3\times5\times4$	$=120=$	$320\times576\times432\times1/576\times1/1152$
1	0	1	$1\times2\times4\times5\times8$	$=320=$	$320\times576\times576\times1/576\times1/576$
1	1	0	$1\times6\times3\times4\times4$	$=288=$	$320\times432\times432\times1/480\times1/432$
1	1	1	$1\times6\times4\times4\times1$	$=96\ =$	$320\times432\times576\times1/480\times1/1728$

5.3.2. Re-parametrisation to compute pairwise or cluster joint distributions

One possibility is to incorporate the messages in the binary potentials, in order to directly extract the pairwise joint distributions as described in Koller & Friedman (2009, Chapter 10): ψ_{ij} is replaced by $\psi_{ij}\odot\mu_{i\to j}\odot\mu_{j\to i}$ while ψ_i is divided by $\mu_{j\to i}$ and ψ_j by $\mu_{i\to j}$. If, for example, sum-product messages are computed, each re-parameterised pairwise potential ψ_{ij} can be shown to be equal to the (unnormalised) marginal distribution of (X_i, X_j) (or an approximation of it if the graph is loopy).

In tree-structured problems, the resulting graphical model is said to be calibrated to emphasise the fact that all pairs of binary potentials sharing a common variable agree on the marginal distribution of this common variable (here x_i):

$$\bigoplus_{x_j}\psi_{ij}=\bigoplus_{x_k}\psi_{ik}.$$

In the loopy case, if an exact approach using tree decomposition is followed, the domains of the messages have a size exponential in the size of the intersection of pairs of clusters, and the re-parametrisation will create new potentials of this size. These messages are included inside the clusters. Each resulting cluster potential will be the (unnormalised) marginal of the joint distribution on the cluster variables. Again, a re-parameterised graphical model on a tree-decomposition is calibrated, and any two intersecting clusters agree on their marginals. This is exploited in the Lauritzen-Spiegelhalter and Jensen sum-product-divide algorithms (Lauritzen & Spiegelhalter 1988; Jensen, Olesen & Andersen 1990). Besides its interest for incremental updates in this context, the re-parameterised graphical model using tree decomposition allows us to locally compute exact marginals for any set of variables in the same cluster.

If a local 'loopy' approach is used instead, re-parameterisations do not change scopes, but provide a re-parameterised model. Estimates of the marginals of the original model can be read directly. For mode (or MAP), such re-parameterisations can follow clever update rules to provide convergent re-parameterisations maximising a well defined criterion. Typical examples of this process are the sequential version of the tree re-weighted algorithm, TRWS (Kolmogorov 2006), or the Max-Product Linear Programming algorithm, MPLP (Globerson & Jaakkola 2008) which aims to optimise a bound on the non-normalised probability of the mode. These algorithms can be exact on graphical models with loops, provided the potential

functions are all submodular (often described as the discrete version of convexity, see for instance Topkis 1978; Cohen *et al.* 2004).

5.3.3. Re-parametrisation in deterministic graphical models

Re-parameterising message passing algorithms have also been used in deterministic graphical models. They are then known as 'local consistency' enforcing or constraint propagation algorithms. On one side, a local consistency property defines the targeted calibration property. On the other side, the enforcing algorithm uses so-called Equivalence Preserving Transformations to transform the original network into an equivalent network, i.e. defining the same joint function, which satisfies the desired calibration/local consistency property. Similar to LBP, Arc Consistency (Waltz 1972; Rossi, van Beek & Walsh 2006) is the most usual form of local consistency, and is related to Unit Propagation in SAT (Biere *et al.* 2009). Arc consistency is exact on trees, while it is usually incrementally maintained during an exact tree search, using re-parametrisation. Because of the idempotency of logical operators (they can be applied several time without changing the result obtained after the first application), local consistencies always converge to a unique fixed point.

Local consistency properties and algorithms for WCSPs are closely related to message passing for MAP. They are however always convergent, thanks to suitable calibration properties (Schiex 2000; Cooper & Schiex 2004; Cooper *et al.* 2010), and also solve tree structured problems or problems where all potential functions are submodular.

These algorithms can be directly used to tackle the max-product and sum-product problems in a MRF. The re-parametrised MRF is then often more informative that the original one. For instance, under the simple conditions that all potential functions with scope larger than 1 are bounded by 1, a trivial upper bound of the normalising constant Z is $\prod_i \sum_{x_i} \psi_i(x_i)$. This naive upper bound can be considerably tightened by re-parameterising the MRF using a soft-arc consistency algorithm (Viricel *et al.* 2016).

6. Heuristics and approximations for inference

We have mainly discussed methods for exact inference in graphical models. They are useful if an order for variable elimination with small treewidth is available. In many real life applications, interaction network are seldom tree-shaped, and their treewidth can be large (e.g. a grid of pixel in image analysis). Consequently, exact methods cannot be applied anymore. However, they provide the inspiration to derive heuristic methods for inference that can be applied to any graphical model. What is meant by a heuristic method is an algorithm that is (a priori) not derived from the optimisation of a particular criterion, as opposed to what we would call an approximation method. Nevertheless, we shall soften this distinction, and show that well performing message passing-based heuristics can sometimes be interpreted as approximate methods. For the marginalisation task, the most widespread heuristic derived from variable elimination and message passing principles is LBP. In the last decade, a better understanding of this heuristic was reached, and it can now be re-interpreted as a particular instance of variational approximation methods (Wainwright & Jordan 2008). A variational approximation of a distribution p is defined as the best approximation of p in a class $\mathcal{Q}$ of tractable distributions (for inference), according to the Kullback–Leibler divergence. Depending on the application (e.g. discrete or continuous variables), several choices for $\mathcal{Q}$ can be considered. The connection with variable elimination principles and

treewidth is not obvious at first sight. However, as we just emphasised, LBP can be cast in the variational framework. The treewidth of the chosen variational distribution depends on the nature of the variables: (i) in the case of discrete variables the treewidth need be low: in most cases, the class $\mathcal{Q}$ is formed by independent variables (mean field approximation), with associated treewidth equal to 0, and some works consider a class $\mathcal{Q}$ with associated treewidth equal to 1 (see Section 6.1); (ii) in the case of continuous variables, the treewidth of the variational distribution is the same as in the original model: $\mathcal{Q}$ is in general chosen to be the class of multivariate Gaussian distributions, for which numerous inference tools are available.

We recall here the two key components for a variational approximation method: the Kullback–Leibler divergence and the choice of a class of tractable distributions. We then explain how LBP can be interpreted as a variational approximation method. Finally we recall the rare examples where some statistical properties of an estimator obtained using a variational approximation have been established. In Section 7 we will illustrate how variational methods can be used to derive approximate EM algorithms for estimation in CHMM.

6.1. Variational approximations

The Kullback–Leibler divergence, $KL(q\|p)$, is equal to $\sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log q(\boldsymbol{x})/p(\boldsymbol{x})$ and measures the dissimilarity between two probability distributions p and q. KL is not symmetric, hence not a distance. It is positive, and it is null if and only if p and q are equal. Let us consider now that q is constrained to belong to a family $\mathcal{Q}$, which does not include p. The solution q^* of $\arg\min_{q\in\mathcal{Q}} KL(q\|p)$ is then the best approximation of p in $\mathcal{Q}$ according to the Kullback–Leibler divergence. It is called the variational distribution. If $\mathcal{Q}$ is a set of tractable distributions for inference, then marginals, mode or normalising constant of q^* can be used as approximations for the same quantities on p.

Variational approximations were originally defined in the field of statistical mechanics, as approximations of the minimum of the free energy $F(q)$,

$$F(q) = -\sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log \prod_{B\in\mathcal{B}} \psi_B(x_B) + \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log q(\boldsymbol{x}).$$

They are also known as Kikuchi approximations or Cluster Variational Methods, CVM (Kikuchi 1951). Minimising $F(q)$ is equivalent to minimising $KL(q\|p)$, since

$$F(q) = -\sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log p(\boldsymbol{x}) - \log(Z) + \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log q(\boldsymbol{x}) = KL(q\|p) - \log(Z).$$

The mean field approximation is the most naive approximation among the family of Kikuchi approximations. Let us consider a binary Markov random field on n vertices whose joint distribution is

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_i \exp\left(a_i x_i + \sum_{(i,j)\in E} b_{ij} x_i x_j\right), \ \forall x_i \in \{0,1\}$$

We can derive its mean field approximation, corresponding to the class $\mathcal{Q}^{MF}$ of fully factorised distributions (i.e. an associated graph of treewidth equal to 0): $\mathcal{Q}^{MF} = \{q, \text{ such that } q(\boldsymbol{x}) = \prod_{i\in V} q_i(x_i)\}$.

Since variables are binary $\mathcal{Q}^{MF}$ corresponds to joint distributions of independent Bernoulli variables with respective parameters $q_i = q_i(1)$. Namely for all q in $\mathcal{Q}^{MF}$, we can write

$q(\boldsymbol{x}) = \prod_i q_i^{x_i}(1-q_i)^{1-x_i}$. The optimal approximation (in terms of Kullback–Leibler divergence) within this class of distributions is characterised by the set of q_i's which minimise $KL(q\|p)$. Denoting E_q the expectation with respect to q, $KL(q\|p) - \log Z$ is

$$\begin{aligned}&\mathrm{E}_q\left(\sum_i \left[X_i \log q_i + (1-X_i)\log(1-q_i)\right] - \sum_i a_i X_i - \sum_{(i,j)\in E} b_{ij} X_i X_j\right)\\ &= \sum_i \left[q_i \log q_i + (1-q_i)\log(1-q_i)\right] - \sum_i a_i q_i - \sum_{(i,j)\in E} b_{ij} q_i q_j.\end{aligned}$$

This expectation has a simple form because of the specific structure of q. Minimising it with respect to q_i gives the fixed-point relation that each optimal q_i^{MF}'s must satisfy:

$$\log\left[q_i^{MF}/(1-q_i^{MF})\right] = a_i + \sum_{j:(i,j)\in E} b_{ij} q_j^{MF}.$$

leading to

$$q_i^{MF} = \frac{\exp(a_i + \sum_{j:(i,j)\in E} b_{ij} q_j^{MF})}{1+\exp(a_i + \sum_{j:(i,j)\in E} b_{ij} q_j^{MF})}.$$

It is interesting to note that this expression is very close to the expression of the conditional probability that $X_i = 1$ given that all other variables in the neighbourhood of i:

$$\Pr(X_i = 1 | x_{N_i}) = \frac{\exp(a_i + \sum_{j:(i,j)\in E} b_{ij} x_j)}{1+\exp(a_i + \sum_{j:(i,j)\in E} b_{ij} x_j)}.$$

The variational distribution q_i^{MF} can be interpreted as equal to this conditional distribution, with neighbouring variables fixed to their expected values under the distribution q^{MF}. This explains the mean field approximation name. Note that in general q_i is not equal to the marginal $p_i(1)$.

The choice of the class $\mathcal{Q}$ is indeed a critical trade-off between opposing desirable properties: it must be large enough to guarantee a good approximation, and small enough to contain only distributions for which inference in manageable. In the next section, a particular choice for $\mathcal{Q}$, the Bethe class, is emphasised. In particular, this enables us to link the LBP heuristic to variational methods. Other choices are possible, and have been used. For instance, in the structured mean field setting (Ghahramani & Jordan 1997; Wainwright & Jordan 2008), the distribution of a factorial HMM is approximated using a variational approach; the multivariate hidden state is decoupled, and the variational distribution q of the conditional distribution of hidden states is that of independent Markov chains (here again, the treewidth is equal to 1). The Chow-Liu algorithm (Chow & Liu 1968) computes the minimum of $KL(p\|q)$ for a distribution q whose associated graph is a spanning tree of the graph of p. This amounts to computing the best approximation of p among graphical models with treewidth equal to 1. Finally, an alternative to treewidth reduction is to choose the variational approximation in the class of exponential distributions. This has been applied to Gaussian process classification (Kim & Ghahramani 2006) using a multivariate Gaussian approximation of the posterior distribution of the hidden field. This method relies on the use of the EP algorithm (Minka 2001). In this algorithm, $KL(p\|q)$ is minimised instead of $KL(q\|p)$. The choice of minimising one or the other depends on computational tractability.

6.2. LBP heuristics as a variational method

If p and q are pairwise MRF whose associated graph $G=(V,E)$ is the same and is a tree, then $q(\boldsymbol{x})=\prod_{(i,j)\in E} q(x_i,x_j)/\prod_{i\in V} q(x_i)^{d_i-1}$, where $\{q(x_i,x_j)\}$ and $\{q(x_i)\}$ are coherent sets of order 2 and order 1 marginals of q, respectively, and d_i is the degree of vertex i in the tree. In this particular case, the free energy is expressed as (see Heskes, Zoeter & Wiegerinck 2004; Yedidia, Freeman & Weiss 2005)

$$F(q)=-\sum_{(i,j)\in E}\sum_{x_i,x_j} q(x_i,x_j)\log\psi(x_i,x_j)-\sum_{i\in V}\sum_{x_i} q(x_i)\log\psi(x_i)$$
$$+\sum_{(i,j)\in E}\sum_{x_i,x_j} q(x_i,x_j)\log q(x_i,x_j)+\sum_{i\in V}(d_i-1)\sum_{x_i} q(x_i)\log q(x_i)$$

The Bethe approximation consists in applying to an arbitrary graphical model the same formula of the free energy as the one used for a tree, then minimising it over the variables $\{q(x_i,x_j)\}$ and $\{q(x_i)\}$ under the constraint that they are probability distributions and that $q(x_i)$ is the marginal of $q(x_i,x_j)$. By extension, the Bethe approximation can be interpreted as a variational method associated with the family $\mathcal{Q}^{\text{Bethe}}$ of unnormalised distributions that can be expressed as $q(\boldsymbol{x})=\prod_{(i,j)\in E} q(x_i,x_j)/\prod_{i\in V} q(x_i)^{d_i-1}$ with $\{q(x_i,x_j)\}$ and $\{q(x_i)\}$ coherent sets of order 2 and order 1 marginals.

Yedidia, Freeman & Weiss (2005) established that the fixed points of LBP (when they exist, convergence is still not well understood, see Weiss 2000 and Mooij & Kappen 2007) are stationary points of the problem of minimising the Bethe free energy, or equivalently $KL(q\|p)$ with q in the class $\mathcal{Q}^{\text{Bethe}}$ of distributions. Furthermore, Yedidia, Freeman & Weiss (2005) showed that for any class of distributions $\mathcal{Q}$ corresponding to a particular CVM method, it is possible to define a generalised LBP algorithm whose fixed points are stationary points of the problem of minimising $KL(q\|p)$ in $\mathcal{Q}$.

The drawback of the LBP algorithm and its extensions (Yedidia, Freeman & Weiss 2005) is that they are not associated with any theoretical bound on the error made on the marginals approximations. Nevertheless, LBP is increasingly used for inference in graphical models for its good behaviour in practice (Murphy, Weiss & Jordan 1999). It is implemented in software packages for inference in graphical models such as libDAI (Mooij 2010) or OpenGM2 (Andres, Beier & Kappes 2012).

6.3. Statistical properties of variational estimates

Maximum-likelihood parameter estimation in graphical models is often intractable because it could require to compute marginals or normalising constants. A computationally efficient alternative to Monte-Carlo estimates are variational estimates, obtained using a variational approximation of the model. From a statistical point-of-view, because variational estimation is only an approximation of maximum-likelihood estimation, the resulting parameter estimates do not benefit of the typical properties of Maximum Likelihood Estimates (MLE), such as consistency or asymptotic normality. Unfortunately, no general theory exists for variational estimates, and results are available only for some specific models (see e.g. Hall, Ormerod & Wand 2011 for the consistency in the Poisson log-normal model and Blei, Kucukelbir & McAuliffe 2017 for some other examples). From a more general point of view, in a Bayesian context, Wang & Titterington (2005) and Wang & Titterington (2006)

studied the properties of variational estimates. They proved that the approximate conditional distribution are centred on the true posterior mean, but with a too small variance. Celisse, Daudin & Pierre (2012) proved the consistency of the (frequentist) variational estimates of the Stochastic Block Model (SBM), while Gazal, Daudin & Robin (2012) empirically established the accuracy of their Bayesian counterpart. Variational Bayes estimates are also proposed by Jaakkola & Jordan (2000) for logistic regression, and the approximate posterior also turns out to be very accurate. A heuristic explanation for these two positive examples (SBM and logistic regression) is that, in both cases, the class $\mathcal{Q}$ used for the approximate conditional (or posterior) distribution q is sought so as to asymptotically contain the true conditional distribution.

7. Illustration on CHMM

In this last section, we illustrate how the different algorithms discussed, in the CHMM framework, perform in practice for marginal inference when the model parameters are known, and how concretely they can be exploited in the EM algorithm to perform parameter estimation.

7.1. Comparison of exact variable elimination, variational inference and Gibbs sampling in practice

We compared the following inference algorithms on the problem of computing the marginals of all the hidden variables of the CHMM model of pest propagation described in Section 2.3, conditional on the observed variables. We simulated 10 datasets with the following parameters values: $\rho = 0.2$, $\nu = 0.5$, $\epsilon = 0.15$, $f_n = 0.3$ and $f_p = 0.1$. For each data set, we ran the following algorithms, using libDAI software (Mooij 2010): Junction Tree (JT, exact method using the principles of tree decomposition and block by block elimination); LBP; Mean Field approximation (MF); and Gibbs Sampling (GS, Geman & Geman 1984),with 10,000 runs, each with a burn-in of 100 iterations and then 10,000 iterations. We compared the algorithms on three criteria: running time (*time* variable), mean absolute difference between the true marginal probability of state 0 and the estimated one, over all hidden variables (*diff-marg* variable), and percentage of hidden variables which are not restored to their true value with the mode of the estimated marginal (*error-resto* variable). The results (see Table 4) are presented for increasing values of n, the number of rows (and also of columns) of the square grid of fields (i.e. $I = n^2$). Beyond $n = 3$, JT fails due to time and space complexity, so for computing *diff-marg* we used the GS marginals instead of the true marginals. These results illustrate well the fact that approximate inference methods based on the principle of variable elimination are very time efficient compared to Monte-Carlo methods (less than 4 minutes for a problem with $I = 10,000$ hidden variables), while being still very accurate. Furthermore, even a naive variational method like the mean field one can be interesting if accurate marginal estimates are not required but we are only interested in preserving their mode.

7.2. Variational approximation for estimation in CHMM

We now illustrate how variational approximations have been used for parameter estimation using an EM algorithm in the case of CHMM.

Table 4. Comparison of Junction Tree (JT), Loopy Belief Propagation (LBP), Mean Field (MF) and Gibbs Sampling (GS) inference algorithms on the CHMM model of pest propagation: (a) running time, in seconds; (b) mean difference between the true and the estimated marginal of state 0 (when JT fails due to time and space complexity, we use GS marginals as true ones); (c) percentage of hidden variables not restored to their true value when using the mode of the marginals.

(a)

time	JT	LBP	MF	GS
$n=3$	0.04	0.04	0.03	1.05
$n=5$	–	0.19	0.14	3.30
$n=10$	–	1.07	0.65	13.99
$n=100$	–	219.31	134.31	3,499.6
$n=200$	–	1,026.2	746.68	29,341.0

(b)

diff-marg	LBP	MF	GS
$n=3$	0.001	0.032	0.032
$n=5$	0.003	0.037	–
$n=10$	0.003	0.032	–
$n=100$	0.003	0.032	–
$n=200$	0.003	0.032	–

(c)

error-resto	JT	LBP	MF	GS
$n=3$	20.00	19.80	19.26	20.19
$n=5$	–	18.60	19.27	18.93
$n=10$	–	17.87	17.70	17.83
$n=100$	–	18.19	18.39	18.20
$n=200$	–	18.18	18.40	18.18

Exact EM algorithm CHMMs are examples of incomplete data models, as they involve variables $(\boldsymbol{O}, \boldsymbol{H})$, and only variables $\boldsymbol{O}$ are observed. Maximum likelihood inference for such models aims at finding the values of the parameters θ which maximise the (log-)likelihood of the observed data $\boldsymbol{o}$, i.e. solve $\max_\theta \log p^\theta(\boldsymbol{o})$. The most popular algorithm to achieve this task is the EM algorithm. One of its formulations reads as an iterative maximisation procedure of the following functional:

$$F(\theta, q) = \mathrm{E}_q(\log p^\theta(\boldsymbol{o}, \boldsymbol{H})) - \mathrm{E}_q(\log q(\boldsymbol{H})) = \log p^\theta(\boldsymbol{o}) - KL(q(\boldsymbol{H}) \| p^\theta(\boldsymbol{H}|\boldsymbol{o})),$$

where q stands for any distribution on the hidden variables $\boldsymbol{HY}$, and E_q stands for the expectation under the arbitrary distribution q. The EM algorithm consists in alternatively maximising $F(\theta, q)$ with respect to q (E-step) and to θ (M-step). The solution of the E-step is $q(\boldsymbol{h}) = p^\theta(\boldsymbol{h}|\boldsymbol{o})$, since the Kullback–Leibler divergence is then minimal, and even null in this case. When replacing $q(\boldsymbol{h})$ by $q(\boldsymbol{h}) = p^\theta(\boldsymbol{h}|\boldsymbol{o})$ in F, we find that the M-step amounts to maximising $\mathrm{E}[\log p^\theta(\boldsymbol{o}, \boldsymbol{H})|\boldsymbol{o}]$.

Exact computation of $p^\theta(\boldsymbol{h}|\boldsymbol{o})$ can be performed by observing that (2) can be rewritten as

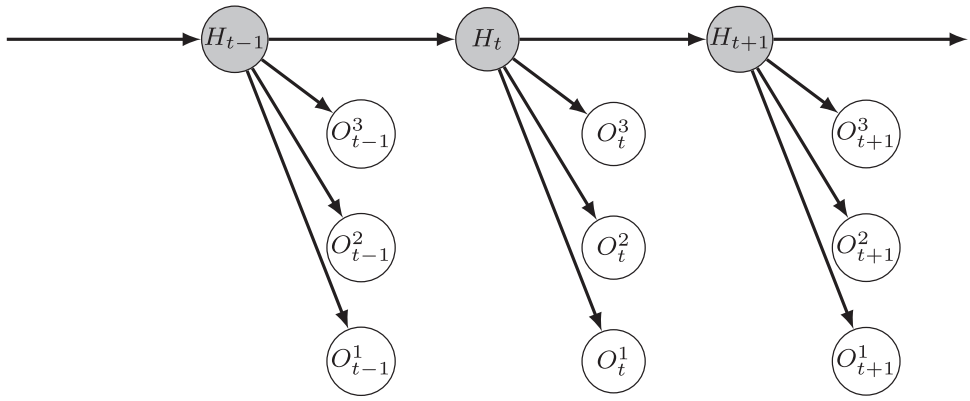

Figure 10. Graphical representation of $p(h, o)$ for a coupled HMM when merging hidden variables at each time step.

$$p^\theta(\boldsymbol{h}, \boldsymbol{o}) \propto \psi^{\text{init}'}(h_1) \left(\prod_{t=2}^{T} \psi^{M'}(h_{t-1}, h_t) \right) \times \left(\prod_{i=1}^{I} \prod_{t=1}^{T} \psi^{E}(h_t^i, o_t^i) \right),$$

where $\psi^{\text{init}'}$ is the global initial distribution, equal to $\prod_{i=1}^{I} \psi^{\text{init}}(h_1^i)$, and $\Psi^{M'}$ is the global transition probability, equal to $\prod_{i=1}^{I} \psi^{M}(h_{t-1}^i, h_{t-1}^{L-i}, h_t^i)$. This expression is equivalent to merging all hidden variables of a given time step. It corresponds to the graphical model given in Figure 10. Denoting by K the number of possible values for each hidden variable, we end up with a regular hidden Markov model with K^I possible hidden states. Both $p^\theta(\boldsymbol{h}|\boldsymbol{o})$ and its mode can then be computed in an exact manner with either the forward–backward recursion or the Viterbi algorithm for mode evaluation. Both procedures have the same complexity: $O(TK^{2I})$. The exact calculation can therefore be achieved provided that K^I remains small enough, but becomes intractable when the number of signals I exceeds a few tens.

7.2.1. Several variational approximations for the EM algorithm

For more complex graphical structures, explicitly determining $p^\theta(\boldsymbol{h}|\boldsymbol{o})$ can be too expensive to perform exactly. A first approach to derive an approximate E-step is to seek a variational approximation of $p^\theta(\boldsymbol{h}|\boldsymbol{o})$ assuming that $q(\boldsymbol{h})$ is restricted to a family $\mathcal{Q}$ of tractable distributions, as described in Section 6.1. The choice of $\mathcal{Q}$ is critical, and requires achievement of an acceptable balance between approximation accuracy and computation efficiency. Choosing $\mathcal{Q}$ typically amounts to breaking down some dependencies in the original distribution to end up with some tractable distribution. In the case of CHMM, the simplest distribution is the class of fully factorised distributions (i.e. mean field approximation):

$$\mathcal{Q}_0 = \{q : q(\boldsymbol{h}) = \prod_{i=1}^{I} \prod_{t=1}^{T} q_{it}(h_t^i)\}.$$

Such an approximation of $p^\theta(\boldsymbol{h}|\boldsymbol{o})$ corresponds to the graphical model of Figure 11. Intuitively, this approximation replaces the stochastic influence between the hidden variables by its mean value.

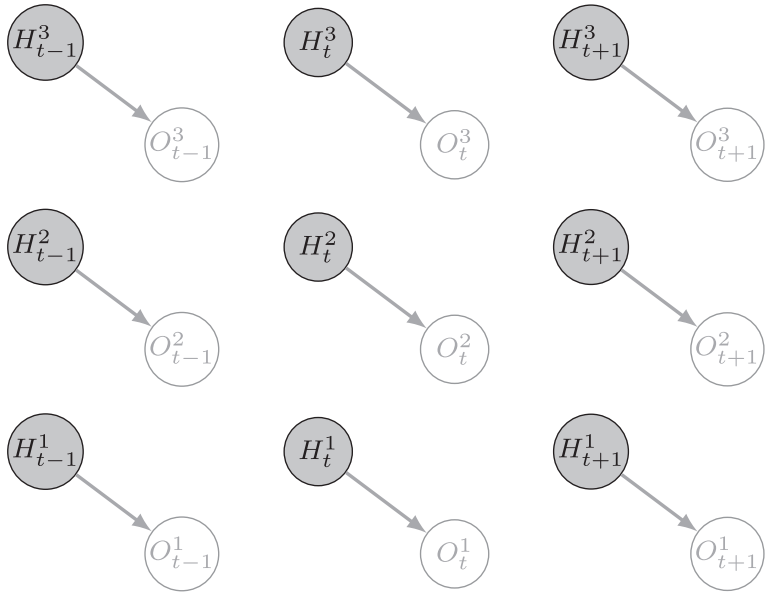

Figure 11. Graphical representation for the mean-field approximation of $p(h, o)$ in a coupled HMM. Observed variables are indicated in light grey since they are not part of the variational distribution which is a distribution only on the hidden variables.

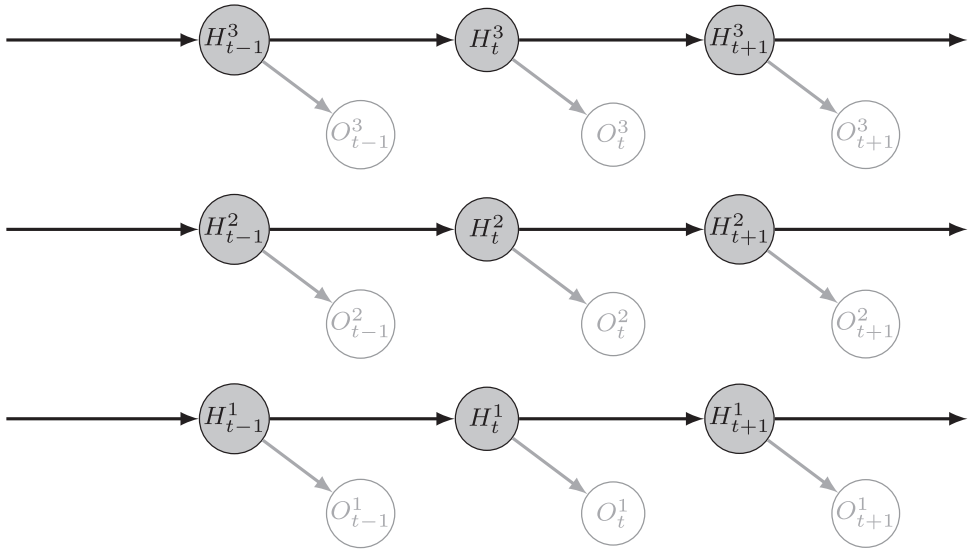

Figure 12. Graphical representation for the approximation of $p(h, o)$ in a coupled HMM by independent heterogeneous Markov chain. Observed variables are indicated in light grey since they are not part of the variational distribution which is a distribution only on the hidden variables.

As suggested in Wainwright & Jordan (2008), a less drastic approximation of $p^{\theta}(\boldsymbol{h}|\boldsymbol{o})$ can be obtained using the distribution family of independent heterogeneous Markov chains:

$$\mathcal{Q}_M = \{q : q(h) = \prod_i \prod_t q_{it}(h_t^i | h_{t-1}^i)\}.$$

which is consistent with the graphical representation of an independent HMM, as depicted in Figure 12.

An alternative is to use the Bethe approximation of $F(\theta, q)$. Then the LBP algorithm can be used to provide an approximation of the conditional marginal distributions on singletons and pairs of variables (no other marginals are involved in the E step of EM). This approach has been proposed in Heskes, Zoeter & Wiegerinck (2004). The advantage of this approach compared to the variational approximations based on families $\mathcal{Q}_0$ or $\mathcal{Q}_M$, is that it provides an approximation of the joint conditional distribution of pairs of hidden variables within the same time step, instead of assuming that they are independent.

8. Conclusion and discussion

This tutorial on variable elimination for exact and approximate inference is an introduction to the basic concepts of variable elimination, message passing and their links with variational methods. It introduces these fields to statisticians confronted with inference in graphical models. The main message is that the exact inference should not be systematically ruled out. Before looking for an efficient approximate method, wise advice is to try to evaluate the treewidth of the graphical model. In practice, this question is not easy to answer. Nevertheless several algorithms exist that provide an upper bound of the treewidth together with the associated variable elimination order (minimum degree, minimum fill-in, maximum cardinality search, etc). Even if it is not optimal, this ordering can be used to perform exact inference if the bound is small enough.

Examples where the low treewidth of the graphical model has been successfully exploited to perform exact inference in problems apparently too complex are numerous. Korhonen & Parviainen (2013) simplified the NP-hard problem of learning the structure of a Bayesian network from data when the underlying network has 'low' treewidth. They proposed an exact score-based algorithm to learn graph structure using dynamic programming. Berg, Järvisalo & Malone (2014) compared their approach with an encoding of the algorithm in the framework of Maximum Satisfiability and obtained improved performance on classical Machine Learning datasets with networks up to 29 nodes. Akutsu, Tamura & Horimoto (2009) tackled the problem of Boolean acyclic network completion. More specifically, their aim was to achieve the smallest number of modifications in the network, so that the distribution is consistent with the binary observations at the nodes. The authors established the general NP-completeness of the problem even for tree-structured networks. They however reported that these problems can be solved in polynomial time for networks with bounded treewidth and in-degree, and with enough samples (in the order of at least log of the number of nodes). Their findings were applied (Tamura & Akutsu 2014) to obtain the sparsest possible set of modifications in the activation and inhibition functions of a signalling network (comprising 57 nodes and 154 edges) after a hypothesised cell-state alteration in colorectal cancer patients. Xing (2004) introduced two Bayesian probabilistic graphical models of genomic data analysis devoted to (i) the identification of motifs and cis-regulatory modules from transcriptional

regulatory sequences, and (ii) the haplotype inference from genotypes of SNPs (single-nucleotide polymorphisms). The inference for these two high-dimensional models on hybrid distributions is computationally very complex. The author noted that the exact computation (e.g. of MAP or marginal distributions) might be feasible for models of bounded tree-width, if a good variable ordering is available. However, the question on how to find this latter one is not addressed, and an approximate generalised mean field inference algorithm is developed. Finally, the reader can find in Berger, Singht & Xu 2008 more illustration of how the notion of treewidth can help simplifying the parametrisation of many algorithms in bioinformatics.

For the reader interested in testing the inference algorithms presented in this article, the list provided by Kevin Murphy (`https://www.cs.ubc.ca/~murphyk/Software/bnsoft.html`), even though slightly out-dated, gives a good idea of the variety of existing software packages, most of them being dedicated to a particular family of graphical models (directed, or undirected). One of the reasons why variable elimination-based techniques for inference in graphical models is not well widespread outside the communities of researchers in Computer Science and Machine Learning is probably that no software exists which it is both generic and with an easy interface from R, Python or Matlab.

Obviously this tutorial is not exhaustive, since we chose to focus on fundamental concepts. While many important results on treewidth and graphical models are several decades old, the area is still lively, and we now broaden our discussion to a few recent works which tackle some challenges related to the computation of the treewidth.

Because they offer efficient algorithms, graphical models with a bounded treewidth offer an attractive target when the aim is to learn a model that best represents some given sample. In Kumar & Bach (2013), the problem of learning the structure of an undirected graphical model with bounded treewidth is approximated by a convex optimisation problem. The resulting algorithm has polynomial time complexity. As discussed in Kumar & Bach (2013), this algorithm is useful in deriving tractable candidate distributions in a variational approach, which go beyond the usual variational distributions with treewidth zero or one.

For optimisation (mode evaluation), other exact techniques are offered by tree search algorithms such as Branch and Bound (Lawler & Wood 1966), that recursively consider possible conditioning of variables. These techniques often exploit limited variable elimination processing to prevent exhaustive search, either using message-passing like algorithms (Cooper *et al.* 2010) to compute bounds that can be used for pruning, or by performing 'on-the-fly' elimination of variables with small degree (Larrosa 2000).

Beyond pairwise potential functions, the time needed for simple update rules of message passing becomes exponential in the size of the scope of the potential functions. However, for specific potential functions involving many (or all) variables, exact messages can be computed in reasonable time, even in the context of convergent message passing for optimisation. This can be done using polytime graph optimisation algorithms such as shortest path or mincost flow algorithms. Such functions are known as global potential functions (Vicente, Kolmogorov & Rother 2008; Werner 2008) in probabilistic graphical models, and as global cost functions (Lee & Leung 2009; Allouche *et al.* 2012; Lee & Leung 2012) in deterministic Cost Function Networks.

Different problems appear with continuous variables, where counting requires integration of functions. Here again, for specific families of distributions, exact (analytic) computations are possible for distributions with conjugate distributions. For message passing, several

solutions have been proposed. For instance, a recent message passing scheme proposed by Noorshams & Wainwright (2013) relies on the combination of orthogonal series approximation of the messages, and the use of stochastic updates. We refer the reader to references in Noorshams & Wainwright (2013) for alternative methods dealing with continuous variables message passing. Variational methods are also largely exploited for continuous variables, in particular in Signal Processing (Smidi & Quinn 2006).

Finally, we have excluded Monte-Carlo methods from the scope of our review. However, the combination of the inference methods presented in this article and stochastic methods for inference is a new area that researchers have begun exploring. Recent sampling algorithms have been proposed that use exact optimisation algorithms to sample points with high probability in the context of estimating the partition function. Additional control in the sampling method is needed to avoid biased estimations: this may be hashing functions enforcing a fair sampling (Ermon *et al.* 2014) or randomly perturbed potential functions using a suitable noise distribution (Hazan, Maji & Jaakkola 2013). More recently, Monte-Carlo and variational approaches have been combined to propose Discrete Particle Variational Inference (Saeedi *et al.* 2017), an algorithm that benefits from the accuracy of the former and the rapidity of the latter.

We hope this review will enable more cross-fertilisations of this sort, combining statistics and computer science, stochastic and deterministic algorithms for inference in graphical models.

References

Akutsu, T., Tamura, T. & Horimoto, K. (2009). Completing networks using observed data. In *Proceedings of the Conference on Algorithmic Learning Theory*, eds. R. Gavaldà, G. Lugosi, T. Zeugmann and S. Zilles, pp. 126–140. Berlin: Springer.

Allouche, D., Bessiere, C., Boizumault, P., *et al.* (2012). Filtering decomposable global cost functions. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, ed. D. Fox, pp. 407–413. Palo Alto: AAAI Press.

Amestoy, P., Davis, T.A. & Duff, I.S. (1996). An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications* **17**, 886–905.

Andres, B., Beier, T. & Kappes, J.H. (2012). OpenGM: A C++ library for discrete graphical models. arXiv 1206.0111.

Andres, B., Beier, T. & Kappes, J.H. (2013). OpenGM benchmark - CVPR'2013 section. Available from URL: http://hci.iwr.uni-heidelberg.de/opengm2/?l0=benchmark.

Arnborg, S. (1985). Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT* **25**, 2–23.

Arnborg, S., Corneil, D.G. & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic Discrete Methods* **8**, 277–284.

Austad, H. & Tjelmeland, H. (2017). Approximate computations for binary Markov random fields and their use in Bayesian models. *Statistics and Computing* **27**, 1271–1292.

Baker, J., Deng, L., Glass, J., Khudanpur, S., Lee, C.H., Morgan, N. & O'Shaughnessy, D. (2009). Developments and directions in speech recognition and understanding, Part 1 [DSP education]. *IEEE Signal Processing Magazine* **26**, 75–80.

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge: Cambridge University Press.

Berg, J., Järvisalo, M. & Malone, B. (2014). Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proceedings of the Seventeenth International Workshop on Artificial Intelligence and Statistics*, vol. 33, eds. S. Kasi and J. Corander, pp. 86–95. Brookline: PMLR.

Berger, B., Singht, R. & Xu, J. (2008). Graph algorithms for biological systems analysis. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ed. S.-H. Teng, pp. 142–151. Philadelphia: SIAM.

BERTELÉ, U. & BRIOSHI, F. (1972). *Nonserial Dynamic Programming*. Orlando: Academic Press.

BIERE, A., HEULE, M., VAN MAAREN, H. & WALSH, T. (2009). *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, vol. 185. Amsterdam, The Netherlands: IOS Press.

BISHOP, C.M. (2006). *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag.

BISTARELLI, S., MONTANARI, U. & ROSSI, F. (1997). Semiring based constraint solving and optimization. *Journal of the ACM* **44**, 201–236.

BLEI, D.M., KUCUKELBIR, A. & MCAULIFFE, J.D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association* **112**, 859–877.

BODLAENDER, H.L. (1993). A tourist guide through treewidth. *Acta Cybernetica* **11**, 1–21.

BODLAENDER, H.L. (1998). A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* **209**, 1–45.

BODLAENDER, H.L. & KOSTER, A.M.C.A. (2010). Treewidth computations I. Upper bounds. *Information and Computation* **208**, 259–275.

BODLAENDER, H.L., GILBERT, J.R., HAFSTEINSSON, H. & KLOKS, T. (1995). Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms* **18**, 238–255.

BODLAENDER, H.L., KOSTER, A. & EIJKHOF, F.V.D. (2005). Preprocessing rules for triangulation of probabilistic networks. *Computational Intelligence* **21**, 286–305.

BONNEAU, M., GABA, S., PEYRARD, N. & SABBADIN, R. (2014). Reinforcement learning-based design of sampling policies under cost constraints in Markov random fields: Application to weed map reconstruction. *Computational Statistics & Data Analysis* **72**, 30–44.

BRAND, M. (1997). Coupled hidden Markov models for modeling interacting processes. Technical report *405*, MIT.

CARRIGER, F. & BARRON, M. (2016). A practical probabilistic graphical modeling tool for weighing ecological risk-based evidence. *Soil and Sediment Contamination: An International Journal* **25**, 476–487.

CASELLA, G. & GEORGE, E.I. (1992). Explaining the Gibbs sampler. *The American Statistician* **46**, 167–174.

CELISSE, A., DAUDIN, J.J. & PIERRE, L. (2012). Consistency of maximum-likelihood and variational estimators in the stochastic block model. *Electronic Journal of Statistics* **6**, 1847–1899.

CHOI, H., FERMIN, D., NESVIZHSKII, A., GHOSH, D. & QIN, Z. (2013). Sparsely correlated hidden Markov models with application to genome-wide location studies. *Bioinformatics* **29**, 533–541.

CHOW, C.K. & LIU, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* **14**, 462–467.

COHEN, D., COOPER, M., JEAVONS, P. & KROKHIN, A. (2004). A maximal tractable class of soft constraints. *Journal of Artificial Intelligence Research* **22**, 1–22.

COOPER, M.C. (2004). Cyclic consistency: a local reduction operation for binary valued constraints. *Artificial Intelligence* **155**, 69–92.

COOPER, M.C. & SCHIEX, T. (2004). Arc consistency for soft constraints. *Artificial Intelligence* **154**, 199–227.

COOPER, M.C., DE GIVRY, S., SANCHEZ, M., SCHIEX, T., ZYTNICKI, M. & WERNER, T. (2010). Soft arc consistency revisited. *Artificial Intelligence* **174**, 449–478.

DAVIS, M. & PUTNAM, H. (1960). A computing procedure for quantification theory. *Journal of the ACM* **7**, 210–215.

DE GIVRY, S., SCHIEX, T. & VERFAILLIE, G. (2006). Exploiting tree decomposition and soft local consistency in weighted CSP. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, ed. J.A. Hendler, pp. 22–27. Palo Alto: AAAI Press.

DECHTER, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* **113**, 41–85.

DECHTER, R. & PEARL, J. (1988). Network-based heuristics for constraint satisfaction problems. In *Search in Artificial Intelligence*, eds. L. Kanal and V. Kumar, pp. 370–425. Berlin: Springer-Verlag.

DEMPSTER, A.P., LAIRD, N.M. & RUBIN, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* **39**, 1–38.

DUFFIN, R.J. (1965). Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications* **10**, 303–313.

ELIDAN, G. & GLOBERSON, A. (2010). UAI inference challenge 2010. Available from URL: www.cs.huji.ac.il/project/UAI10.

ELIDAN, G. & GLOBERSON, A. (2011). The probabilistic inference challenge. Available from URL: http://www.cs.huji.ac.il/project/PASCAL/index.php.

ERMON, S., GOMES, C., SABHARWAL, A. & SELMAN, B. (2014). Low-density parity constraints for hashing-based discrete integration. In *Proceedings of the Thirty-First International Conference on Machine Learning*, eds. E.P. Xing and T. Jebara, pp. 271–279. Brookline: PMLR.

FOURIER, J. (1827). Analyse des travaux de l'Académie royale des sciences, pendant l'année 1824. *Mémoires de l'Académie des sciences de l'Institut de France* **7**.

FREY, B. & MACKAY, D. (1998). A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*, eds. M.J. Kearns, S.A. Solla and D.A. Cohn, pp. 479–485. Cambridge: MIT Press.

FRIEL, N. & RUE, H. (2007). Recursive computing and simulation-free inference for general factorizable models. *Biometrika* **94**, 661–672.

FRIEL, N., PETTITT, A.N., REEVES, R. & WIT, E. (2009). Bayesian inference in hidden Markov random fields for binary data defined on large lattices. *Journal of Computational and Graphical Statistics* **18**, 243–261.

FULKERSON, D. & GROSS, O. (1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics* **15**, 835–855.

GAZAL, S., DAUDIN, J.J. & ROBIN, S. (2012). Accuracy of variational estimates for random graph mixture models. *Journal of Statistical Computation and Simulation* **82**, 849–862.

GEMAN, S. & GEMAN, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 721–741.

GHAHRAMANI, Z. & JORDAN, M. (1997). Factorial hidden Markov models. *Machine Learning* **29**, 245–273.

GLOBERSON, A. & JAAKKOLA, T. (2008). Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems*, eds. P. Koller, D. Schuurmans, Y. Bengio and L. Bottou, pp. 553–560. Cambridge: MIT Press.

GOGATE, V. (2014). UAI 2014 inference competition. Available from URL: www.hlt.utdallas.edu/-vgogate/uai14-competition.

GORDON, N.J., SALMOND, D.J. & SMITH, A.F.M. (1993). A novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F on Radar and Signal Processing* **140**, 107–113.

HABIB, M. & LIMOUZY, V. (2009). On some simplicial elimination schemes for chordal graphs. *Electronic Notes in Discrete Mathematics* **32**, 125–132.

HALL, P., ORMEROD, J.T. & WAND, M. (2011). Theory of Gaussian variational approximation for a Poisson mixed model. *Statistica Sinica* **21**, 369–389.

HAZAN, T., MAJI, S. & JAAKKOLA, T. (2013). On sampling from the Gibbs distribution with random maximum a-posteriori perturbations. In *Advances in Neural Information Processing Systems*, eds. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Weinberger, pp. 1268–1276. Cambridge: MIT Press.

HEGGERNES, P., EISENSTAT, S., KUMFERT, G. & POTHEN, A. (2001). The computational complexity of the minimum degree algorithm. In *14th Norwegian Computer Science Conference*. Troms, Norway.

HESKES, T., ZOETER, O. & WIEGERINCK, W. (2004). Approximate expectation maximization. *Advances in Neural Information Processing Systems* **16**, 353–360.

HÖHNA, S., HEATH, T., BOUSSAU, B., LANDIS, M., RONQUIST, F. & HUELSENBECK, J. (2014). Probabilistic graphical model representation in phylogenetics. *Systematic Biology* **63**, 753–771.

HUANG, H. & KOLLER, D. (2013). Subproblem-tree calibration: A unified approach to max-product message passing. In *Proceedings of the Thirtieth International Conference on Machine Learning*, eds. S. Dasgupta and D. McAllester, pp. 190–198. Brookline: PMLR.

ILLIAN, J., MARTINO, S., SØRBYE, S., GALLEGO-FERNÁNDEZ, J., ZUNZUNEGUI, M., ESQUIVIAS, M. & TRAVIS, J.J. (2013). Fitting complex ecological point process models with integrated nested laplace approximation. *Methods in Ecology and Evolution* **4**, 305–315.

JAAKKOLA, T. & JORDAN, M. (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing* **10**, 25–37.

JENSEN, F.V. & NIELSEN, T.D. (2007). *Bayesian Networks and Decision Graphs*. Berlin: Springer Publishing Company, Incorporated, 2nd edn.

JENSEN, F., OLESEN, K. & ANDERSEN, S. (1990). An algebra of Bayesian belief universes for knowledge-based systems. *Networks* **20**, 637–659.

KASK, K., GELFAND, A., OTTEN, L. & DECHTER, R. (2011). Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, eds. W. Burgard and D. Roth, pp. 54–60. Palo Alto: AAAI Press.

KIKUCHI, R. (1951). A theory of cooperative phenomena. *Physical Review* **81**, 988–1003.

KIM, H. & GHAHRAMANI, Z. (2006). Bayesian Gaussian process classification with the EM-EP algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**, 1948–1959.

KOHLAS, J. (2003). *Information Algebras: Generic Structures for Inference*. London: Springer.

KOLLER, D. & FRIEDMAN, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Cambridge: MIT Press.

KOLMOGOROV, V. (2006). Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**, 1568–1583.

KORHONEN, J. & PARVIAINEN, P. (2013). Exact learning of bounded tree-width Bayesian networks. In *Proceedings of the Sixteenth International Workshop on Artificial Intelligence and Statistics*, vol. 31, eds. C.M. Carvalho and P. Ravikumar, pp. 370–378. Brookline: PMLR.

KSCHISCHANG, F.R., FREY, B.J. & LOELIGER, H.A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* **47**, 498–519.

KUMAR, K.S.S. & BACH, F. (2013). Convex relaxations for learning bounded treewidth decomposable graphs. In *Proceedings of the Thirtieth International Conference on Machine Learning*, eds. S. Dasgupta and D. McAllester, pp. 525–533. Brookline: PMLR.

LARROSA, J. (2000). Boosting search with variable elimination. In *Proceedings of the Sixth Conference on Principles and Practice of Constraint Programming*, ed. R. Dechter, pp. 291–305. Berlin: LNCS, Springer.

LAURITZEN, S.L. (1996). *Graphical Models*. Oxford: Clarendon Press.

LAURITZEN, S. & SPIEGELHALTER, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society – Series B* **50**, 157–224.

LAWLER, E. & WOOD, D. (1966). Branch-and-bound methods: A survey. *Operations Research* **14**, 699–719.

LEE, J. & LEUNG, K.L. (2009). Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, ed. C. Boutilier, vol. 9, pp. 559–565. San Francisco: Morgan Kaufmann Publishers, Inc.

LEE, J.H.M. & LEUNG, K.L. (2012). Consistency techniques for global cost functions in weighted constraint satisfaction. *Journal of Artificial Intelligence Research* **43**, 257–292.

LI, S.Z. (2001). *Markov Random Field Modeling in Image Analysis*. Berlin: Springer-Verlag.

LIU, J.W.H. (1992). The multifrontal method for sparse matrix solution: Theory and practice. *SIAM Review* **34**, 82–109.

LIU, Q. & IHLER, A.T. (2011). Bounding the partition function using Holder inequality. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, eds. L. Getoor and T. Scheffer, pp. 849–856. Palo Lato: AAAI Press.

LIU, Y., CARBONELL, J., GOPALAKRISHNAN, V. & WEIGELE, P. (2009). Conditional graphical models for protein structural motif recognition. *Journal of Computational Biology* **16**, 639–657.

LOVÁSZ, L. (2005). Graph minor theory. *Bulletin of the American Mathematical Society* **43**, 75–86.

MAATHUIS, M., COLOMBO, D., KALISCH, M. & BÜHLMANN, P. (2010). Predicting causal effects in large-scale systems from observational data. *Nature Methods* **7**, 247–248.

MARINESCU, R. & DECHTER, R. (2006). Memory intensive branch-and-bound search for graphical models. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, ed. A. Cohn, pp. 1200–1205. Palo Alto: AAAI Press.

MESEGUER, P., ROSSI, F. & SCHIEX, T. (2006). Soft constraints. In *Handbook of Constraint Programming*, eds. F. Rossi, P. van Beek and T. Walsh, pp. 281–388. New York: Elsevier.

MINKA, T. (2001). A family of algorithms for approximate Bayesian inference. Ph.D. thesis, MIT.

MOOIJ, J.M. (2010). libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research* **11**, 2169–2173.

MOOIJ, J.M. & KAPPEN, H.J. (2007). Sufficient conditions for convergence of the sum-product algorithm. *IEEE Transactions on Information Theory* **53**, 4422–4437.

MURPHY, K. (2002). Dynamic Bayesian networks: representation, inference and learning. Ph.D. thesis, University of California, Berkeley.

MURPHY, K. (2012). *Machine Learning: A Probabilistic Perspective*. Cambridge: The MIT Press.

MURPHY, K., WEISS, Y. & JORDAN, M. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, eds. M. Meila and T. Heskes, pp. 467–475. San Francisco: Morgan Kaufmann Publishers, Inc.

NOCK, H. & OSTENDORF, M. (2003). Parameter reduction schemes for loosely coupled HMMs. *Computer Speech & Language* **17**, 233–262.

NOORSHAMS, N. & WAINWRIGHT, M.J. (2013). Belief propagation for continuous state spaces: stochastic message-passing with quantitative guarantees. *Journal of Machine Learning Research* **14**, 2799–2835.

Optimization Research Group, N. (2012). MiniZinc challenge 2012. Available from URL: http://www.minizinc.org/challenge2012/challenge.html.

OTTEN, L., IHLER, A., KASK, K. & DECHTER, R. (2012). Winning the PASCAL 2011 MAP challenge with enhanced AND/OR branch-and-bound. In *NIPS DISCML Workshop*, 2011.

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

PRALET, C., VERFAILLIE, G. & SCHIEX, T. (2007). An algebraic graphical model for decision with uncertainties, feasibilities, and utilities. *Journal of Artificial Intelligence Research*, **29**, 421–489.

RABINER, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**, 257–286.

REEVES, R. & PETTITT, A.N. (2004). Efficient recursions for general factorisable models. *Biometrika* **91**, 751–757.

ROBERT, C. & CASELLA, G. (2004). *Monte Carlo Statistical Methods*. New York: Springer-Verlag.

ROBERTSON, N. & SEYMOUR, P. (1986). Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms* **7**, 309–322.

ROSSI, F., VAN BEEK, P. & WALSH, T. (eds.) (2006). *Handbook of Constraint Programming*. New York: Elsevier.

SAEEDI, A., KULKARNI, T., MANSINGHKA, V. & GERSHMAN, S.J. (2017). Variational particle approximations. *Journal of Machine Learning Research* **18**, 1–29.

SCHIEX, T. (1999). A note on CSP graph parameters. Technical report 1999/03, INRA.

SCHIEX, T. (2000). Arc consistency for soft constraints. In *Proceedings of the Sixth International Conference on Principles and Practice of Constraint Programming*, ed. R. Dechter, pp. 411–424. Berlin: LNCS, Springer.

SCHIEX, T., FARGIER, H. & VERFAILLIE, G. (1995). Valued constraint satisfaction problems: hard and easy problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, ed. C.S. Mellish, pp. 631–637. San Francisco: Morgan Kaufmann Publishers, Inc.

SERANG, O. (2014). The probabilistic convolution tree: Efficient exact bayesian inference for faster LC-MS/MS protein inference. *PLoS ONE* **9**(3), e91507. https://doi.org/10.1371/journal.pone.0091507.

SHAFER, G. & SHENOY, P. (1988). Local computations in hyper-trees. Working paper 201, School of Business, University of Kansas.

SHENOY, P. & SHAFER, G. (1990). Axioms for probability and belief-function propagation. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, eds. P. Bonissone, M. Henrion, L. Kanal and J. Lemmer, pp. 169–198. San Francisco: Morgan Kaufmann Publishers, Inc.

SMIDI, V. & QUINN, A. (2006). *The Variational Bayes Method in Signal Processing*. Signal and Communication Technologies, Palo Alto: Springer.

SOLOMON, C. & BRECKON, T. (2011). *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. Chichester: Wiley-Blackwell.

TAMURA, T. & AKUTSU, T. (2014). Theory and method of completion for a Boolean regulatory network using observed data. In *Biological Data Mining and Its Applications in Healthcare*, eds. X. Li, S.-K. Ng and J.T.L. Wang, pp. 123–146. Singapore: World Scientific.

TARJAN, R. & YANNAKAKIS, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing* **13**, 566–579.

TARLOW, D., GIVONI, I. & ZEMEL, R. (2010). Hop-map: Efficient message passing with high order potentials. In *Proceedings of the Thirteenth International Workshop on Artificial Intelligence and Statistics*, eds. Y.W. Teh and M. Titterington, pp. 812–819. Brookline: PMLR.

TOPKIS, D. (1978). Minimizing a submodular function on a lattice. *Operations Research* **26**, 305–321.

VANDENBERGHE, L. & ANDERSEN, M.S. (2014). Chordal graphs and semidefinite optimization. *Foundations and Trends in Optimization* **1**, 241–433.

VICENTE, S., KOLMOGOROV, V. & ROTHER, C. (2008). Graph cut based image segmentation with connectivity priors. In *Proceedings of the Twenty-Sixth IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8. Piscataway: IEEE.

VIRICEL, C., SIMONCINI, D., BARBE, S. & SCHIEX, T. (2016). Guaranteed weighted counting for affinity computation: Beyond determinism and structure. In *Proceedings of the Twenty-Second International Conference on Principles and Practice of Constraint Programming*, ed. M. Rueher, pp. 733–750. Berlin:

LNCS, Springer.

WAINWRIGHT, M.J. & JORDAN, M.I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* **1**, 1–305.

WAINWRIGHT, M., JAAKKOLA, T. & WILLSKY, A. (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory* **51**, 2313–2335.

WALTZ, D.L. (1972). Generating semantic descriptions from drawings of scenes with shadows. Technical report AI271, MIT, Cambridge MA.

WANG, B. & TITTERINGTON, D. (2005). Inadequacy of interval estimates corresponding to variational Bayesian approximations. In *Proceedings of the Tenth International Workshop in Artificial Intelligence and Statistics*, eds. R. Cwell and Z. Ghahramani, pp. 373–380. Brookline: PMLR.

WANG, B. & TITTERINGTON, D. M. (2006). Convergence properties of a general algorithm for calculating variational Bayesian estimates for a normal mixture model. *Bayesian Analysis* **1**, 625–650.

WEISS, Y. (2000). Correctness of local probability propagation in graphical models with loops. *Neural Computation* **12**, 1–41.

WERNER, T. (2008). High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *Proceedings of the Twenty-Sixth IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 109–116. Piscataway: IEEE.

XING, P. (2004). Probabilistic graphical models and algorithms for genomic analysis. Ph.D. thesis, University of California, Berkeley, CA, USA.

YEDIDIA, J., FREEMAN, W. & WEISS, Y. (2005). Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory* **51**, 2282–2312.

ZHONG, S. & GHOSH, J. (2002). HMMs and coupled HMMs for multi-channel EEG classification. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 1254–1159. Piscataway: IEEE.