Journal of Artificial Intelligence Research 27 (2006) 1-23 Submitted 02/06; published 09/06

A Variational Inference Procedure Allowing Internal Structure for Overlapping Clusters and Deterministic Constraints

Dan Geiger DANG@CS.TECHNION.AC.IL
Computer Science Dept., Technion,
Haifa, 32000, Israel

Christopher Meek MEEK@MICROSOFT.COM
Microsoft Research, Microsoft Corporation,
Redmond, WA 98052, USA

Ydo Wexler YWEX@CS.TECHNION.AC.IL
Computer Science Dept., Technion,
Haifa, 32000, Israel

Abstract

We develop a novel algorithm, called VIP*, for structured variational approximate inference. This algorithm extends known algorithms to allow efficient multiple potential updates for overlapping clusters, and overcomes the difficulties imposed by deterministic constraints. The algorithm's convergence is proven and its applicability demonstrated for genetic linkage analysis.

1. Introduction

Probabilistic graphical models are an elegant framework to represent joint probability distributions in a compact manner. The independence relationships between random variables which are nodes in the graph are represented through the absence of arcs in the model. This intuitively appealing presentation also naturally enables the design of efficient general-purpose algorithms for computing marginal probabilities, called inference algorithms.

The general inference problem is NP-hard (Cooper, 1990; Dagum & Luby, 1993), and although there are many cases where the model is small (or, more precisely, has a small treewidth) and exact inference algorithms are feasible, there are others in which the time and space complexity makes the use of such algorithms infeasible. In these cases fast yet accurate approximations are desired.

We focus on variational algorithms: a powerful tool for efficient approximate inference that offers guarantees in the form of a lower bound on the marginal probabilities. This family of approaches aims to minimize the KL divergence between a distribution Q and the target distribution P by finding the best distribution Q from some family of distributions for which inference is feasible. In particular, we have a joint distribution $P(X)$ over a set of discrete variables X and our goal is to compute the marginal probability $P(Y = y)$ where $Y \subseteq X$. Further assume that this exact computation is not feasible. The idea is to replace P with a distribution Q which can be used to compute a lower bound for $P(Y = y)$. We let $H = X \setminus Y$. Then, by using Jensen's inequality we get the following bound:

$$\log P(y) = \log \sum_h Q(h) \frac{P(y,h)}{Q(h)} \geq \sum_h Q(h) \log \frac{P(y,h)}{Q(h)} = -D(Q(H) \,\|\, P(Y=y,H))$$

where $D(\cdot \,\|\, \cdot)$ denotes the KL divergence between two probability distributions. The quantity $-D(Q \,\|\, P)$ is often called the free-energy where P and Q are possibly un-normalized distributions. Variational techniques aim to choose a distribution Q such that the lower bound is as high as possible, or equivalently, such that the KL divergence between $Q(h)$ and $P(h|Y=y)$ is minimized.

Variational approaches such as the mean field, generalized mean field, and structured mean field differ only with respect to the family of approximating distributions that can be used, with the structural mean field approach subsuming the remaining approaches as special cases. The research of several authors guided our work: Saul & Jordan (1996), Ghahramani & Jordan (1997), Wiegerinck (2000) and Bishop & Winn (2003).

The contributions of this paper are threefold. First we develop an extension to the algorithm by Wiegerinck (2000), which we call VIP*, that allows for a set potentials of the approximating distribution Q to be updated simultaneously even if the clusters of Q overlap. Algorithm VIP* is N-fold faster than Wiegerinck's algorithm for $N \times N$ grid-like models and yields two orders of magnitude improvement for large graphs such as genetic linkage analysis model of large pedigrees. Note that simultaneous updates were first presented for phylogenic trees by Jojic et al. (2004). Second, we prove the convergence of VIP* and of previous variational methods via a novel proof method, using properties of the KL divergence. Third, we extend VIP* to allow deterministic constraints in the model and demonstrate the applicability of this extension to genetic linkage analysis.

2. Background

This background section is based primarily on the paper by Weigerinck (2000), which in turn builds on pioneering works such as the papers of Saul & Jordan (1996) and Ghahramani & Jordan (1997). Our review provides a new exposition of this material.

We denote distributions by $P(x)$ and $Q(x)$ and related un-normalized distributions by $\tilde{P}(x) \propto P(x)$ and $\tilde{Q}(x) \propto Q(x)$. Let X be a finite set of variables and x be an instantiation of these variables. Let $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(d_i)$ where d_i is the projection of the instantiation x to the variables in $D_i \subseteq X$ and where Ψ_i is a non-negative function, commonly called a *potential*. The constant Z_P normalizes the product of potentials and the subsets $\{D_i\}_{i=1}^I$ are allowed to overlap. We often suppress the arguments of a potential and of a distribution, using Ψ_i instead of $\Psi_i(d_i)$ and P instead of $P(X)$.

Our goal is to find a distribution Q that minimizes the KL divergence between Q and P. We further constrain Q to be of the form $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$ where Z_Q is a normalizing constant and where $C_1, \ldots, C_J$ are possibly overlapping subsets of X, which we call *clusters*. Finding an optimum Q, however, can be difficult. A more modest and common goal is devising iterative converging algorithms such that in each iteration the KL divergence between an approximating distribution Q and P decreases unless Q is a stationary point.

Throughout, we define $Q(w|u) = \frac{1}{|W \setminus U|}$ for instantiations $U = u$ for which $Q(u) = 0$. Consequently, all terms in the equality $Q(w,u) = Q(u)Q(w|u)$ are well defined even if

$Q(u) = 0$. Moreover, this convention maintains properties such as $\sum_{W \setminus U} Q(w|u) = 1$ and $Q(w,z|u) = Q(w|z,u)Q(z|u) = \frac{1}{|W \setminus \{U \cup Z\}|} \cdot \frac{1}{|Z \setminus U|} = \frac{1}{|\{W \cup Z\} \setminus U|}$. We also note that $Q(x) \log Q(x) = 0$ whenever $Q(x) = 0$ and thus the KL divergence

$$D(Q \,\|\, P) = \sum_x Q(x) \log \frac{Q(x)}{P(x)}$$

is not finite if and only if $P(x) = 0$ and $Q(x) > 0$ for some instance x.

Our starting point is the algorithm developed by Wiegerinck (2000). The algorithm finds such a distribution Q as follows: it iterates over the clusters C_j and their instantiations c_j to update the potentials $\Phi_j(c_j) = e^{\gamma_j(c_j)}$ via the following update equation:

$$\gamma_j(c_j) \leftarrow -\sum_{\{k:g_{kj}=1\}} \sum_{C_k \setminus C_j} Q(c_k|c_j) \log \Phi_k(c_k) + \sum_{\{i:f_{ij}=1\}} \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) \qquad (1)$$

where g_{kj} and f_{ij} are two indicator functions defined via $g_{kj} = 0$ if $Q(C_k|c_j) = Q(C_k)$ for every instance c_j of C_j and 1 otherwise, and $f_{ij} = 0$ if $Q(D_i|c_j) = Q(D_i)$ for every instance c_j of C_j and 1 otherwise. Wiegerinck (2000) proved convergence of this algorithm to a stationary point using Lagrangians. Throughout we call this iterative procedure, Wiegerinck's algorithm.

Wiegerinck's algorithm relies at each step on an algorithm to compute the conditional probabilities $Q(c_k|c_j)$ and $Q(d_i|c_j)$ from an un-normalized distribution $\tilde{Q}$ represented by a set of potentials $\Phi_j(c_j)$. This can be accomplished by any inference algorithm such as *bucket elimination algorithm* or the *sum-product algorithm* described by Dechter (1999) and Kschischang, Frey & Loeliger (2001) . It is important to note that for $\tilde{Q}(x) = \prod_j \Phi_j(c_j)$ the computation of these conditionals is not affected by multiplying any Φ_j by a constant α.

Wiegerinck's algorithm generalizes the mean field (MF) algorithm and the generalized mean field (GMF) algorithm (Xing, Jordan & Russell, 2003, 2004). The mean field algorithm is the special case of Wiegerinck's algorithm in which each C_j contains a single variable. Similarly, the generalized mean field algorithm is the special case in which the C_j are disjoint subsets of variables. When C_j are disjoint clusters, the formula for γ_j in Eq. 1 simplifies to the GMF equations as follows (first term drops out):

$$\gamma_j(c_j) \leftarrow \sum_{\{i:f_{ij}=1\}} \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i). \qquad (2)$$

The term $Q(d_i|c_j)$ can be made more explicit when C_j are disjoint clusters (Bishop & Winn 2003). In particular, the set $D_i \setminus C_j$ partitions into $D_i^k = (D_i \setminus C_j) \cap C_k$ for $k = 1, \ldots, J$ where $k \neq j$. Note that $D_i^k = D_i \cap C_k$. Using this notation, $Q(d_i|c_j) = \prod_k Q(d_i^k)$ where $Q(d_i^k) = 1$ whenever $D_i^k = \emptyset$. This factorization further simplifies the formula for γ_j as follows:

$$\gamma_j(c_j) \leftarrow \sum_{\{i:f_{ij}=1\}} \sum_{D_i^1} Q(d_i^1) \ldots \sum_{D_i^J} Q(d_i^J) \log \Psi_i(d_i). \qquad (3)$$

This simplification is achieved automatically when using bucket elimination for computing γ_j. The iterated sums in Eq. 3 are in fact the buckets formed by bucket elimination when C_j are disjoint.

Eq. 1 requires repeated computation of the quantities $Q(c_k|c_j)$ and $Q(d_i|c_j)$. This repetition can be significant because there could be many indices k such that $Q(C_k|c_j) \neq Q(C_k)$, and many indices i such that $Q(D_i|c_j) \neq Q(D_i)$. As these computations share many sub-computations it is therefore reasonable to add a data structure to facilitate a more efficient implementation for these function calls. In particular, it is possible to save computations if the sets $C_1, \ldots, C_J$ form a junction tree.

A set of clusters $C_1, \ldots, C_J$ forms a *junction tree* iff there exists a set of trees JT having one node, called C_j, for each cluster of variables C_j, and for every two nodes C_i and C_j of JT, which are connected with a path in JT, and for each node C_k on this path, $C_i \cap C_j \subseteq C_k$ holds. By a set of trees we mean an undirected graph, not necessarily connected, with no cycles. Note that this definition allows a junction tree to be a disconnected graph. When $C_1, \ldots, C_J$ form a junction tree, $Q(x)$ has the decomposable form $Q(x) = \prod_j \Phi_j(c_j) / \prod_e \Phi_e(s_e)$, where Φ_j are marginals on the subsets C_j of X, and where Φ_e are the marginals on intersections $S_e = C_i \cap C_j$, one for each two neighboring clusters in the junction tree (Jensen 1996).

Wiegerinck (2000) enhanced his basic algorithm so that it maintains a consistent junction tree JT for the distribution $Q(x)$. Consistency means that $\sum_{C_j \setminus C_k} \Phi_j = \sum_{C_k \setminus C_j} \Phi_k$ for every two clusters. In a consistent junction tree, each potential $\Phi_j(C_j)$ is proportional to $Q(C_j)$. An update of a potential during the algorithm may yield an inconsistent junction tree, however, consistency is maintained by applying DistributeEvidence(Φ_j) (Jensen 1996) after each update of a potential. The procedure DistributeEvidence(Φ'_j) accepts as input a consistent junction tree and a new cluster marginal Φ'_j for C_j, and updates the potential of every neighboring cluster C_k of C_j via

$$\Phi'_k(c_k) \leftarrow \Phi_k(c_k) \frac{\sum_{C_j \setminus C_k} \Phi'_j(c_j)}{\sum_{C_j \setminus C_k} \Phi_j(c_j)} \tag{4}$$

and each neighboring cluster recursively propagates the update by applying Eq. 4 to all its neighbors except the one from which the update came. The output of this procedure is a consistent junction tree, having the same clusters, where Φ'_j is the (possibly un-normalized) marginal probability of Q on C_j, and where the conditional probability $Q(X|C_j)$ remains unchanged (Jensen 1996, pp. 74).

Wiegerinck's enhanced algorithm, which uses a junction tree, iteratively updates the potential of each cluster (node in the junction tree), using the potentials of other clusters and separators. However, since the junction tree may not be consistent after the update, the algorithm applies the procedure DistributeEvidence(Φ_j) to the junction tree, after each update. Note that our description omits a normalization step in Wiegerinck (2000) that is not needed for convergence.

The most time consuming computation in variational algorithms is computing conditional probabilities of the form $Q(c_k|c_j)$ and $Q(d_i|c_j)$. We distinguish among these conditional probabilities as follows.

Definition: A conditional probability $Q(A|c_j)$ is *subsumed* by Q if the set of target variables A is a subset of some cluster C_k of Q (i.e., $(A \setminus C_j) \subseteq C_k$).

Wiegerinck's enhanced algorithm has substantial computational benefits when the conditional probabilities are subsumed. In such cases the needed quantities in Eq. 1, $Q(d_i|c_j)$ and $Q(c_k|c_j)$, are obtained by a mere lookup in the junction tree, and only one call to DistributeEvidence is made for each update.

Weigerinck's basic and enhanced algorithms do not assume any structure for Φ_j, namely, the algorithms hold tables Φ_j with an explicit entry for every instantiation of C_j. Since the computations $Q(c_k|c_j)$ and $Q(d_i|c_j)$ grow exponentially in the size of D_i and C_k, the algorithms become infeasible for large cliques or clusters. For simplification, additional structure to Φ_j was suggested by Wiegerinck (2000, Section 4) of the form,

$$\Phi_j(c_j) = \prod_{l=1}^{n_j} \Phi_{jl}(c_{jl}), \tag{5}$$

where the sets C_{jl}, $l = 1, \ldots, n_j$, are possibly overlapping subsets of C_j, and c_{jl} is the projection of the instantiation c_j on the variables in C_{jl}. Using such structure it is sufficient to hold tables for the subsets C_{jl} which are considerably smaller. Note that when Φ_j has an entry to each instantiation c_j, then $n_j = 1$ and $\Phi_j(c_j) = \Phi_{j1}(c_{j1})$. Weigerinck uses this structure for the potentials Φ_j under the following assumptions:

Definition [Self compatibility]: A distribution Q with clusters C_j and subsets C_{jl} of the form $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$, with clusters that factor according to Eq. 5, is *self compatible* if for every C_j and C_k the set of indices $N_{jk} = \{l : Q(C_k|c_j) = Q(C_k|c_{jl})\}$ is non-empty regardless of the values of the potentials Φ_j, where c_j is an arbitrary instantiation of C_j and c_{jl} is the projection of c_j on C_{jl}.

Definition [Compatibility wrt P]: A distribution Q with clusters C_j and subsets C_{jl} of the form $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$, with clusters that factor according to Eq. 5, is *compatible* wrt a distribution P with sets D_i of the form $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(d_i)$ if for every D_i and C_j the set of indices $M_{ij} = \{l : Q(D_i|c_j) = Q(D_i|c_{jl})\}$ is non-empty, where c_j is an arbitrary instantiation of C_j and c_{jl} is the projection of c_j on C_{jl}.

Note that self-compatibility and compatibility wrt P depend on the form of Q and not on a particular realization of the potentials Φ_j.

Under these assumptions Weigerinck states that considerable simplifications can be deduced, and provides some examples for this statement.

We note that the algorithms of Bishop & Winn (2003) and Jojic et al. (2004) use a stronger assumption that the clusters C_j of the approximating distribution Q are disjoint and that $Q(C_k|c_j) = Q(C_k)$. This assumption, which implies that $Q(C_k|c_j) = Q(C_k|c_{jl})$ and $Q(D_i|c_j) = Q(D_i|c_{jl})$ for every index l, is relaxed by requiring each of these equalities to hold for a single index l (but possibly for multiple indices).

3. Multiple Potential Update using Overlapping Clusters

In this section we develop a new algorithm, called vip*, that uses the additional structure of potentials offered by Eq. 5 to speed up computations. In particular, rather than updating each potential Φ_{jl} separately, we offer a way to update the set of potentials $\{\Phi_{jl}\}_{l=1}^{n_j}$ simultaneously, saving considerable computations. Furthermore, this simultaneous update is enhanced by using a junction tree, despite the fact that the sets $\{C_{jl}\}$ need not form a junction tree, and only $\{C_j\}$ form a junction tree.

Our algorithm uses the definitions of self compatibility and compatibility wrt P, defined earlier, and the following definition of indices.

Definition: Let the indicator function $g_{jk}(l)$ equal 1 for a single fixed index $l \in N_{jk}$ and 0 for all other indices in N_{jk} when $Q(C_k|c_j) \neq Q(C_k)$, and equal 0 otherwise. Let the indicator function $f_{ij}(l)$ equal 1 for a single fixed index $l \in M_{ij}$ and 0 for all other indices in M_{ij} when $Q(D_i|c_j) \neq Q(D_i)$, and equal 0 otherwise.

Algorithm vip* is given in Figure 1. Its convergence is proved in Section 4. The proof requires Q to be self-compatible, compatible wrt P, and in addition, to satisfy $(P(x) = 0) \Rightarrow (Q(x) = 0)$. Note that $D(Q \,\|\, P) = \infty$ for distributions Q which do not satisfy the last assumption.

The main improvement of the algorithm is an efficient update of potentials. For potentials Φ_j that factorize into smaller potentials Φ_{jl} according to Eq. 5, algorithm vip* only updates Φ_{jl} instead of updating the whole potential Φ_j, as done in Weigerinck's algorithms. The update of the potentials Φ_{jl} as done by vip* is equivalent to updating Φ_j according to Eq. 1, up to an irrelevant constant, but does not require to compute the update equation for each instance of the cluster C_j. The proposed change considerably speeds up the previous algorithms.

The algorithm gets as input a target distribution P with sets D_i of the form $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ and an approximating distribution Q with clusters C_j which is self-compatible, compatible wrt P and satisfies the condition $(P(x) = 0) \Rightarrow (Q(x) = 0)$. Distribution Q is of the form $Q(x) = \frac{1}{Z_Q}\prod_j \Phi_j(c_j)$ where the potential of every cluster C_j factors according to $\Phi_j(c_j) = \prod_{l=1}^{n_j} \Phi_{jl}(c_{jl})$ and the clusters form a consistent junction tree. The algorithm iterates over the clusters, updating the potential of every instantiation of each of the subsets C_{jl} according to Eq. 6. To apply the update equation, the quantities $Q(d_i|c_{jl})$ are computed via variable propagation (Jensen, pp 69-80) on the junction tree. When these quantities are subsumed, they are obtained by a mere lookup in the junction tree. Then, after updating the potentials of all subsets C_{jl} for a cluster C_j, procedure DistributeEvidence is applied once to make the junction tree consistent with respect to Φ_j. Since the clusters C_j form a junction tree only via their subsets C_{jl}, Eq. 4 is replaced with Eq. 7. After convergence, algorithm vip* outputs the approximating distribution Q with its revised potentials.

Example 1 *The target distribution P is an $N \times N$ grid of pairwise potentials (see Figure 2a) and the approximating family is defined by a single row and the set of columns in the grid, each augmented with edges to the middle vertex (see Figure 2b) where C_7 is a row of the grid and C_i $(i = 1, \ldots, N = 6)$ are the columns. Using the notation $X_{i,j}$ to denote the vertex at row i and column j in the grid, cluster C_7 is associated with $N - 1$ subsets*

Algorithm VIP*(Q,P)

Input: Two probability distributions $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ and $Q(x) = \frac{1}{Z_Q}\prod_j \Phi_j(c_j)$ where the initial potentials $\Phi_j(c_j) = \prod_{l=1}^{n_j}\Phi_{jl}(c_{jl})$ form a consistent junction tree, and where Q is self-compatible, compatible wrt P, and satisfies $(P(x) = 0) \Rightarrow (Q(x) = 0)$.
Output: A revised set of potentials $\Phi_{jl}(c_j)$ defining a probability distribution Q via $Q(x) \propto \prod_{j,l}\Phi_{jl}(c_{jl})$ such that Q is a stationary point of $D(Q\,\|\,P)$.

Iterate over all clusters C_j until convergence
Step 1.
For $l = 1, \ldots, n_j$:

For every instantiation c_{jl} of C_{jl} apply the following *update equation*:

$$\gamma_{jl}(c_{jl}) \leftarrow -\sum_{\{k:g_{jk}(l)=1\}}\sum_{C_k\setminus C_{jl}} Q(c_k|c_{jl})\log\Phi_k(c_k) + \sum_{\{i:f_{ij}(l)=1\}}\sum_{D_i\setminus C_{jl}} Q(d_i|c_{jl})\log\Psi_i(d_i) \quad (6)$$

$\Phi_{jl}(c_{jl}) \leftarrow e^{\gamma_{jl}(c_{jl})}$

Note: *$Q(d_i|c_{jl})$ is computed via variable propagation (Jensen, pp 69-80) on the junction tree JT. However, when these quantities are subsumed, they are obtained by a mere lookup in JT.*

Step 2. Make JT consistent with respect to Φ_j: DistributeEvidence(JT, Φ_j)

DistributeEvidence(JT, Φ'_j)

Input: A junction tree JT with nodes C_k and potentials $\Phi_k(c_k) = \prod_{l=1}^{n_k}\Phi_{kl}(c_{kl})$. A starting node C_j with a revised potential Φ'_j.
Output: A consistent junction tree.

initialization *source*$(j) \leftarrow 0$; UPDATED$\leftarrow \{j\}$

While (UPDATED$\neq \emptyset$)
$\beta \leftarrow$ first element in UPDATED; UPDATED$\leftarrow$ UPDATED$\setminus\{\beta\}$
For all neighboring nodes C_k of C_β in JT such that $k \neq$ *source*(β)

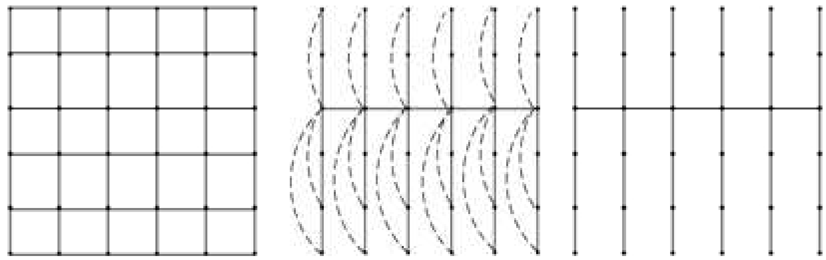
$$\Phi'_{km}(c_{km}) \leftarrow \Phi_{km}(c_{km})\frac{\sum_{C_\beta\setminus C_k}\prod_{l=1}^{n_\beta}\Phi'_{\beta l}(c_{\beta l})}{\sum_{C_\beta\setminus C_k}\prod_{l=1}^{n_\beta}\Phi_{\beta l}(c_{\beta l})} \quad (7)$$

for a single subset m of C_k for which $(C_\beta \cap C_k) \subseteq C_{km}$

source$(k) \leftarrow \beta$

UPDATED$\leftarrow$ UPDATED$\cup\{k\}$

Figure 1: Algorithm VIP*



(a) (b) (c)

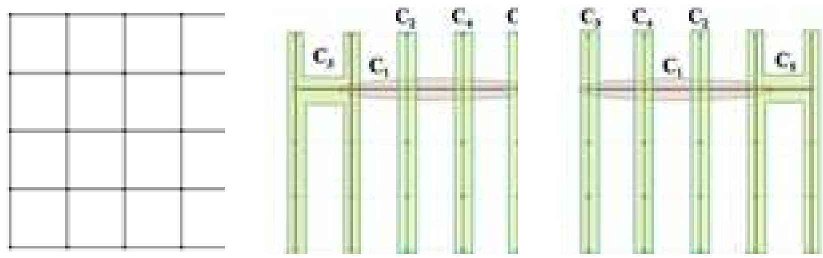
Figure 2: (a) Grid-like P distribution (b) & (c) Approximating distributions Q.

$C_{7l} = \{X_{3,l}, X_{3,l+1}\}$. *Each column cluster* C_j *is associated with 2N-4=8 subsets* C_{jl} *from which* $C_{jl} = \{X_{l,j}, X_{l+1,j}\}$ *for N-1 subsets (*$l = 1, \ldots, 5$*),* $C_{jl} = \{X_{1,j}, X_{3,j}\}$ *for* $l = N$*, and* $C_{jl} = \{X_{l-N+4,j}, X_{3,j}\}$ *for each of the additional N-4 subsets (*$l = 7, 8$*).*

This choice induces a self-compatible approximating distribution Q; every column cluster C_j is independent of another cluster given a subset that contains $X_{3,j}$ (such as C_{j2}). In addition, the row cluster C_7 is independent of every column cluster C_j given C_{7j}. The induced distribution is also compatible wrt P; for each vertical edge $D_v = \{X_{i,j}, X_{i+1,j}\}$ of P, distribution Q satisfies $Q(D_v|c_k) = Q(D_v|c_{k2})$ for a column cluster C_k such that $k \neq j$, and $Q(D_v|c_7) = Q(D_v|c_{7j})$. In addition, for each horizontal edge $D_h = \{X_{i,j}, X_{i,j+1}\}$ in P, distribution Q satisfies $Q(D_h|c_7) = Q(D_h|c_{7j})$, and $Q(D_h|c_k) = Q(D_h|c_{k2})$ for $k \neq j, j+1$. Finally, for the edge D_h and $k = j, j+1$, the approximating distribution satisfies $Q(D_h|c_k) = Q(D_h|c_{kl})$ where $C_{kl} = \{X_{i,k}, X_{3,k}\}$, due to the additional $N-3$ edges added to each column cluster.

Like Wiegerinck's enhanced algorithm, algorithm VIP* has substantial computational benefits when the conditional probabilities $Q(d_i|c_{jl})$ are subsumed. In such cases the needed quantities, $Q(d_i|c_{jl})$ and $Q(c_k|c_{jl})$, are obtained by a mere lookup in the junction tree in step 1 of the algorithm, and only one call to DistributeEvidence is made in step 2, as demonstrated in the next paragraph. The computational efficiency of VIP* is achieved even if the quantities $Q(d_i|c_{jl})$ are not subsumed but only factor to subsumed probabilities. Disjoint clusters is one such special case, in which the quantities $Q(d_i|c_{jl})$ can factor into subsumed probabilities $Q(d_i^k|c_{jl})$, where $D_i^k = D_i \cap C_k$, which are obtainable by lookup in a junction tree.

Consider Example 1 to compare the computational cost of VIP* versus Wiegerinck's basic and enhanced algorithms. Assume Wiegerinck's basic algorithm (Eq. 1), uses the distribution Q given in Figure 2c, with 35 clusters C'_j and 60 sets D_i. Therefore, when a junction tree is not used, $4 \cdot (60 + 34) = 376$ conditionals are computed for each cluster C'_j (edge) not on the boundary of the grid, 94 for each of the four possible values for the edge cluster C'_j. Clearly, if additional clusters are introduced, as shown for example by Figure 2b, then the computational cost grows. By using a junction tree, as done by Wiegerinck's enhanced algorithm, the subsumed conditional probabilities, which are computed separately by Wiegerinck's basic algorithm, are computed with a single call to DistributeEvidence. These computation covers all subsets in Figure 2c. The only conditionals that are not sub-



(a) (b) (c)

Figure 3: The target distribution P is a grid of pairwise potentials as in (a). Two different partitions of the grid into clusters are shown in Figures (b) and (c), both contain the same subsets.

sumed are $Q(d_i|c_j)$ of horizontal edges D_i that are not contained in a single cluster, namely, edges in 2a but not in 2c. These factor to two subsumed probabilities, one computed by the single call described earlier and the other requires a second call to DistributeEvidence. For example, let $D_i = \{X_1, X_2\}$ be a horizontal edge in P which does not overlap with C'_j, then $Q(x_1, x_2|c'_j) = Q(x_1|c'_j, x_2)Q(x_2|c'_j)$. The two conditionals are subsumed, but a second call to DistributeEvidence is needed to obtain $Q(x_1|c'_j, x_2)$. This yields 25 calls to DistributeEvidence. However, in Example 1, one call to DistributeEvidence is sufficient to compute the conditionals for two adjacent horizontal edges, yielding the need only for 15 calls. Therefore, since there are $4 \cdot 15 = 60$ calls to DistributeEvidence, and since the cost of the junction tree algorithm is typically twice the cost of computing conditional probabilities without using a junction tree, this yields a 3-fold speedup for Wiegerinck's enhanced algorithm versus Wiegerinck's basic algorithm. For edges on a boundary, the speedup factor is less than 3. As the size of the grid grows, a smaller fraction of the edges are on the boundary, and, thus, the speedup approaches a 3-fold speedup.

A significant speedup is obtained by using algorithm vip* with clusters C_j and subsets as described in Figure 2b. Note that the additional subsets are needed to meet the compatibility assumption of vip*. Algorithm vip* makes one call to DistributeEvidence per cluster C_j for each non-subsumed conditional, rather than for every edge cluster C'_j. Since vip* uses $N + 1$ clusters C_j, the speedup when compared to Wiegerinck's enhanced algorithm approaches N as the the $N \times N$ grid grows. This $O(N)$ speedup is confirmed in the experiments section (Figure 9).

Another potential benefit of vip* is the possibility of alternating between different choices of clusters which contain identical subsets C_{jl}. A simple example is the grid in Figure 3a, where two such choices are illustrated in Figures 3b and 3c. The two sets of clusters update of the potentials Φ_j differently and therefore can yield better approximations as the distance $D(Q \,\|\, P)$ is reduced with every alternation. In general, we can iterate through a set of choices for clusters and execute vip* for one choice using as initial potentials the potentials Φ_{jl} found for an earlier choice of clusters. The practical benefit of this option of added flexibility remains to be tested in an application.

4. Proof of Convergence

We develop several lemmas that culminate with a proof of convergence of algorithm VIP*. Lemma 1 states that for two un-normalized probability distributions $P(x)$ and $Q(x)$ the KL divergence is minimized when $Q(x)$ is proportional to $P(x)$. Lemma 2 rewrites the KL divergence $D(Q \,\|\, P)$ in terms of the potentials of Q and P using the quantity $\upsilon_j(c_j)$ which, according to Lemma 3, differs from $\gamma_j(c_j) = \sum_l \gamma_{jl}(c_{jl})$ only by a constant. Finally, Theorem 1 asserts that the KL divergence between Q and P decreases with each iteration of Algorithm VIP* unless Q is a stationary point. The proof exploits the new form of $D(Q \,\|\, P)$ provided by Lemma 2, and replaces the term $\upsilon_j(c_j)$ with the terms $\gamma_{jl}(c_{jl})$ used in the update equation of VIP*. A final observation, which uses Lemma 1, closes the proof showing that when the potentials are updated as in algorithm VIP*, the KL divergence is minimized wrt Φ_j.

The first lemma provides a variant of a well known property of KL. Recall that for every two probability distributions $Q(x)$ and $P(x)$, the KL divergence $D(Q(x) \,\|\, P(x)) \geq 0$ and equality holds if and only if $Q(x) = P(x)$ (Cover & Thomas 1991; Theorem 2.6.3). A similar result holds also for un-normalized probability distributions.

Lemma 1 *Let $\tilde{Q}(x)$ and $\tilde{P}(x)$ be non-negative functions such that $\sum_x \tilde{P}(x) = Z_P > 0$, and let*

$$\hat{Q}(x) = \min_{\{\tilde{Q} \mid \sum_x \tilde{Q}(x) = Z_Q\}} D(\tilde{Q}(x) \,\|\, \tilde{P}(x))$$

where Z_Q is a positive constant. Then $\hat{Q}(x) = \frac{Z_Q}{Z_P}\tilde{P}(x)$.

Proof. We observe that

$$D(\tilde{Q}(x) \,\|\, \tilde{P}(x)) = Z_Q \cdot D(\tfrac{\tilde{Q}(x)}{Z_Q} \,\|\, \tfrac{\tilde{P}(x)}{Z_P}) + Z_Q \log \frac{Z_Q}{Z_P}$$

which implies, using the cited result about normalized distributions, that the minimum is obtained when $\frac{\tilde{Q}(x)}{Z_Q} = \frac{\tilde{P}(x)}{Z_P}$, yielding the desired claim. □

The next lemma rewrites the KL divergence so that an optimizing update equation for cluster C_j becomes readily available.

Lemma 2 *Let $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ and $Q(x) = \frac{1}{Z_Q}\prod_j \Phi_j(c_j)$ be two probability distributions. Then,*

$$D(Q \,\|\, P) = \sum_{C_j} Q(c_j) \log \frac{\Phi_j(c_j)}{\Upsilon_j(c_j)} + \log(Z_P) - \log(Z_Q) \tag{8}$$

where $\Upsilon_j(c_j) = e^{\upsilon_j(c_j)}$, and where

$$\upsilon_j(c_j) = -\sum_k \sum_{C_k \setminus C_j} Q(c_k|c_j) \log \Phi_k(c_k) + \sum_i \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) \tag{9}$$

Proof: Recall that

$$D(Q \,\|\, P) = \sum_X Q(x) \log \frac{Q(x)}{P(x)} = -\left[H(Q) + E_Q[\log P(x)]\right] \tag{10}$$

where $H(Q)$ denotes the entropy of $Q(x)$ and E_Q denotes expectation with respect to Q. The entropy term can be written as

$$\begin{aligned} H(Q) &= -\sum_{C_j} \sum_{X \setminus C_j} Q(c_j) Q(x|c_j) \left[\log Q(c_j) + \log Q(x|c_j)\right] \\ &= -\textstyle\sum_{C_j} Q(c_j) \log Q(c_j) - \sum_{C_j} Q(c_j) \sum_{X \setminus C_j} Q(x|c_j) \log Q(x|c_j). \end{aligned}$$

This is a variation of a well known form of $H(Q)$ which is derived by splitting summation over X into summation over C_j and $X \setminus C_j$, and using the fact that $Q(c_j) \sum_{X \setminus C_j} Q(x|c_j) = Q(c_j)$ which holds for every distribution. To split the sum over $X \setminus C_j$ for $Q(c_j) > 0$ we use

$$\log Q(x|c_j) = \log \frac{\frac{1}{Z_Q} \prod_k \Phi_k(c_k)}{Q(c_j)} = \sum_k \log \Phi_k(c_k) - \log Q(c_j) - \log(Z_Q)$$

Thus,

$$\begin{aligned} &\sum_{X \setminus C_j} Q(x|c_j) \log Q(x|c_j) = \\ &\quad \textstyle\sum_k \sum_{X \setminus C_j} Q(c_k|c_j) Q(x|c_k, c_j) \log \Phi_k(c_k) - \sum_{X \setminus C_j} Q(x|c_j) \left[\log Q(c_j) + \log(Z_Q)\right] \end{aligned}$$

and by using $Q(c_k, c_j) \sum_{X \setminus \{C_k \cup C_j\}} Q(x|c_k, c_j) = Q(c_k, c_j)$ this term is further rewritten as

$$\begin{aligned} H(Q) &= -\sum_{C_j} Q(c_j) \log Q(c_j) \\ &\quad - \textstyle\sum_{C_j} Q(c_j) \cdot \left[\sum_{k \neq j} \sum_{C_k \setminus C_j} Q(c_k|c_j) \log \Phi_k(c_k) + \log \frac{\Phi_j(c_j)}{Q(c_j)}\right] + \log(Z_Q) \end{aligned}$$

Note that when $Q(c_j) = 0$ the bracketed term is multiplied by zero, and due to the equality $0 \log 0 = 0$, the product is also zero.

The second term of Eq. 10 is similarly written as

$$\begin{aligned} E_Q[\log P(x)] &= \sum_i \sum_{C_j} Q(c_j) \sum_{X \setminus C_j} Q(x|c_j) \log \Psi_i(d_i) - \log(Z_P) \\ &= \sum_{C_j} Q(c_j) \sum_i \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) - \log(Z_P) \end{aligned} \tag{11}$$

Hence Eq. 10 is rewritten as

$$\begin{aligned} D(Q \,\|\, P) &= \sum_{C_j} Q(c_j) \log Q(c_j) \\ &\quad - \sum_{C_j} Q(c_j) \left[-\sum_k \sum_{C_k \setminus C_j} Q(c_k|c_j) \log \Phi_k(c_k) + \sum_i \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) \right] \end{aligned}$$

$$-\sum_{C_j} Q(c_j) \log \frac{Q(c_j)}{\Phi_j(c_j)} + \log(Z_P) - \log(Z_Q)$$

Denoting the bracketed term by $\upsilon_j(c_j)$, and letting $\Upsilon_j(c_j) = e^{\upsilon_j(c_j)}$, we get

$$D(Q \,\|\, P) = \sum_{C_j} Q(c_j) \log \frac{\Phi_j(c_j)}{\Upsilon_j(c_j)} + \log(Z_P) - \log(Z_Q). \quad \Box$$

The next lemma shows that $\upsilon_j(c_j)$, defined in Eq. 9, and $\gamma_j(c_j) = \sum_l \gamma_{jl}(c_{jl})$, used to update potentials of Q in VIP*, differ only by an additive constant which does not depend on c_j. As argued in Theorem 1, the fact that this difference is a constant enables VIP* to use the latter form, which is a more efficient representation.

Lemma 3 *Let $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(d_i)$ and $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$ where $\Phi_j(c_j) = \prod_l \Phi_{jl}(c_{jl})$, be two probability distributions such that Q is self-compatible and compatible wrt P. Let*

$$\gamma_{jl}(c_{jl}) = -\sum_{\{k: g_{jk}(l)=1\}} \sum_{C_k \setminus C_{jl}} Q(c_k|c_{jl}) \log \Phi_k(c_k) + \sum_{\{i: f_{ij}(l)=1\}} \sum_{D_i \setminus C_{jl}} Q(d_i|c_{jl}) \log \Psi_i(d_i) \quad (12)$$

Then, the difference between $\upsilon_j(c_j)$ defined by Eq. 9 and $\gamma_j(c_j) = \sum_l \gamma_{jl}(c_{jl})$ is a constant that does not depend on c_j.

Proof: We first argue that each term of the form $\sum_{C_k \setminus C_j} Q(c_k|c_j) \log \Phi_k(c_k)$ and each term of the form $\sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i)$ in Eq. 9 that depends on c_j appears exactly once for a single subset C_{jl} in Eq. 12. We then argue that every term in Eq. 12 appears once in Eq. 9.

Since Q is self-compatible, it follows that for every cluster C_k that depends on C_j the function $g_{kj}(l)$ equals one for a single subset C_{jl}, namely $Q(c_k|c_j) = Q(c_k|c_{jl})$, in which case the expression $\sum_{C_k \setminus C_j} Q(c_k|c_{jl}) \log \Phi_k(c_k)$ appears in Eq. 12. Similarly, since Q is compatible wrt P it follows that for every set D_i that depends on C_j the function $f_{ij}(l)$ equals one for a single subset C_{jl}, namely $Q(d_i|c_j) = Q(d_i|c_{jl})$, in which case the expression $\sum_{D_i \setminus C_j} Q(d_i|c_{jl}) \log \Psi_i(d_i)$ appears in the second term of Eq. 12.

It remains to show that every term in Eq. 12 appears once in Eq. 9. Since Q is self-compatible, it is implied that $C_k \cap C_j \subseteq C_{jl}$ and thus summing over $C_k \setminus C_j$ is equivalent to summing over $C_k \setminus C_{jl}$. Therefore, for every k such that $g_{jk}(l) = 1$, the first term of Eq. 12 appears once in Eq. 9. Similarly, since Q is compatible wrt P, it is implied that $D_i \cap C_j \subseteq C_{jl}$ and thus summing over $D_i \setminus C_j$ is equivalent to summing over $D_i \setminus C_{jl}$ and therefore for every i such that $f_{ij}(l) = 1$ the second term of Eq. 12 appears once in Eq. 9. $\Box$

Theorem 1 (Convergence of VIP*) *Let the initial approximating distribution Q be self-compatible and compatible wrt a given distribution P, and assume that $(P(x) = 0) \Rightarrow (Q(x) = 0)$. Then, the revised distribution Q retains these properties, and in each iteration of Algorithm VIP* the KL divergence between Q and P decreases unless Q is a stationary point.*

Proof. Let $Q(x) = \frac{1}{Z_Q}\prod_{jl}\Phi_{jl}(c_{jl})$ where $\Phi_{jl}(c_{jl}) = e^{\gamma_{jl}(c_{jl})}$. We need to show that at the start of each iteration of VIP* the function Q' defined by the revised potentials $\Phi'_j(c_j) = \prod_l \Phi'_{jl}(c_{jl})$ is a probability distribution with the listed properties and that it is closer to P in KL divergence than Q at the start of the previous iteration.

First, we show that Q' maintains the properties listed in the theorem throughout the updates done in VIP*. The properties of self-compatibility and compatibility wrt P are derived from the form of Q and thus are not affected by the updates done in VIP*. For the property $(P(x) = 0) \Rightarrow (Q(x) = 0)$, consider an instance x for which $P(x) = 0$. Since $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ there exists a potential Ψ_i of P for which $\Psi_i(d_i) = 0$, where d_i is the projection of x on the set D_i. Since $Q(x) = 0$ and $Q(x) = \frac{1}{Z_Q}\prod_{jl}\Phi_{jl}(c_{jl})$ there exists a subset C_{jl} for which $Q(c_{jl}) = 0$, where c_{jl} is the projection of x on C_{jl}. Algorithm VIP* updates $\gamma_{jl}(c_{jl})$ to $-\infty$ because $\log\Psi_i(d_i) = -\infty$ and $Q(d_i|c_{jl}) = \frac{1}{|D_i \setminus C_{jl}|}$ by convention, yielding $Q'(x) = 0$, as claimed.

Now, we show that no additional zeroes are introduced into Q whenever $(P(x) = 0) \Rightarrow (Q(x) = 0)$. Hence, the normalizing constant $Z_Q > 0$ and therefore the revised Q' is a probability distribution. For instances c_{jl} for which $Q(c_{jl}) > 0$ the terms $Q(c_k|c_{jl})\log\Phi(c_k)$ and $Q(d_i|c_{jl})\log\Psi(d_i)$ are finite as long as $(P(x) = 0) \Rightarrow (Q(x) = 0)$. This implies $\Phi_{jl}(c_{jl})$ is updated to a positive value and thus, no additional zeroes are introduced into Q.

Using the given form of Q, we have

$$Q(c_j) = \frac{1}{Z_Q}\left[\sum_{X\setminus C_j}\prod_{k\neq j}\Phi_k(c_k)\right]\Phi_j(c_j). \tag{13}$$

We denote the bracketed coefficient of $\Phi_j(c_j)$ by $B_j(c_j)$ and note that it is constant in the sense that it does not depend on the quantity Φ_j being optimized.

We now use Eq. 13 to rewrite the KL divergence as justified by Eq. 8 in Lemma 2:

$$D(Q\,\|\,P) = \frac{1}{Z_Q}\left[\sum_{C_j}\Phi_j(c_j)B_j(c_j)\log\frac{\Phi_j(c_j)B_j(c_j)}{\Upsilon_j(c_j)B_j(c_j)}\right] + \log(Z_P) - \log(Z_Q). \tag{14}$$

Due to Lemma 3, the distance $D(Q\,\|\,P)$ only changes by a constant when replacing $\Upsilon_j(c_j)$ with $\Gamma_j(c_j) = e^{\gamma_j(c_j)}$, where $\gamma_j(c_j) = \sum_l \gamma_{jl}(c_{jl})$ as computed via Eq. 6 of VIP*. Note that $\Gamma_j(c_j)$ does not depend on $\Phi(c_j)$ and is a function of $Q(x)$ only through the conditional distribution of $X \setminus C_j$ given C_j (via $Q(c_k|c_j)$). Hence, Lemma 1 states that the minimum of $D(Q\,\|\,P)$ wrt Φ_j is achieved when $\Phi_j(c_j)$ is proportional to $\Gamma_j(c_j)$. As the potential Φ_j is only held implicitly through the partial potentials Φ_{jl}, step 1 of VIP* updates $\Phi_j(c_j)$ via Eq. 6 to be proportional to $\Gamma_j(c_j)$ by setting each potential $\Phi_{jl}(c_{jl})$ to be proportional to $\gamma_{jl}(c_{jl})$. The proportionality constant does not matter because if Φ_j is multiplied by α, and the arbitrary constraining constant Z_Q is also multiplied by α, these influences cancel in Eq. 14. For simplicity, the algorithm uses $\alpha = 1$ and therefore $\Phi_j(c_j) \leftarrow e^{\gamma_j(c_j)}$. Algorithm VIP* implicitly computes $\Phi_j(c_j)$ according to this formula and hence decreases $D(Q\,\|\,P)$ at each iteration by improving $\Phi_j(c_j)$ while holding all other cluster potentials fixed. Since the KL divergence is lower bounded by zero, VIP* converges. □

The properties of Q required by Theorem 1 of self-compatibility and compatibility wrt P are derived from the form of Q and are satisfied by setting the clusters C_j appropriately. In addition, the condition $(P(x) = 0) \Rightarrow (Q(x) = 0)$ is trivially satisfied for strictly positive distributions P.

Note that the difference between $\upsilon_j(c_j)$ defined by Eq. 9 and $\gamma_j(c_j)$ defined by Eq. 1 is a constant that does not depend on c_j. Consequently, our convergence proof also applies to Wiegerinck's algorithm, because this algorithm is a special case of VIP* where every cluster C_j has a single subset $C_{j1} = C_j$.

5. Handling Deterministic Potentials

When the distribution P is not strictly positive the property $(P(x) = 0) \Rightarrow (Q(x) = 0)$ must hold for the convergence proof of VIP* to apply. In this section we provide a sufficient condition for Q to retain this property.

Definition: An instantiation $W = w$ is *feasible* (wrt to a distribution P) if $P(W = w) > 0$. Otherwise, the instantiation is *infeasible*.

Definition: A *constraining set* wrt a distribution $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(d_i)$ with sets D_i is a minimal set of variables $\Lambda \subseteq D_i$ which has an infeasible instantiation λ.

Definition: A distribution $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$ with clusters C_j is *containable* wrt a distribution $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(d_i)$, if for every constraining set Λ of P there exists at least one cluster C_j such that $\Lambda \subseteq C_j$.

Theorem 2 *Let $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(d_i)$ and $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$ be two distributions where $\Phi_j(c_j) = \prod_l \Phi_{jl}(c_{jl})$ and where Q is containable and compatible wrt P and strictly positive. Then, after* VIP* *iterates once over all clusters C_j, the revised distribution Q satisfies $(P(x) = 0) \Rightarrow (Q(x) = 0)$.*

Proof: From the definition of a constraining set, for every infeasible instantiation x, there exists an infeasible instantiation λ of a constraining set Λ such that λ is a projection of x on Λ. We show that VIP* updates $Q(x) = 0$ for such instantiations. Since Q is containable wrt P there exists a cluster C_j which contains Λ. Furthermore, since $\Lambda \subseteq D_i$ where D_i is a set of P and since Q is compatible wrt P, there exists a subset C_{jl} that contains Λ. For every instantiation c_{jl} which is a projection of λ on C_{jl} the expression $\gamma_{jl}(c_{jl})$ is updated to $-\infty$ according to Eq. 6 of VIP*. This is true because $Q(d_i|c_{jl}) > 0$ and $\log \Psi(d_i) = -\infty$. Since $Q(x) = \frac{1}{Z_Q} \prod_{j,l} \Phi_{jl}(c_{jl})$ this update implies $Q(x) = 0$.$\square$

Whenever the first two compatibility conditions of Theorem 1 hold, it follows that VIP* converges for containable distributions. Note that since every iteration of VIP* decreases the KL divergence, following iterations can not change Q to be greater than zero for instantiation which are infeasible wrt P, as this leads to an infinite distance. However, containability implies a stronger property stated in the next theorem.

Theorem 3 *Let $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ and $Q(x) = \frac{1}{Z_Q}\prod_j \Phi_j(c_j)$ be two distributions where $\Phi_j(c_j) = \prod_l \Phi_{jl}(c_{jl})$ and where Q is containable wrt P and $(P(x) = 0) \Rightarrow (Q(x) = 0)$. Then, after* VIP* *iterates once over all clusters C_j, the revised distribution Q satisfies $(Q(x) = 0) \Rightarrow (P(x) = 0)$.*

Proof: Consider an instantiation x for which $P(x) > 0$. We show that Eq. 6 of VIP* updates $Q(x)$ to a positive value. Since $Q(x) = \frac{1}{Z_Q}\prod_{j,l}\Phi_{jl}(c_{jl})$, it is sufficient to show that the revised potential $\Phi_{jl}(c_{jl})$ is positive for each subset C_{jl} and an instance c_{jl} which is the projection of x on the C_{jl}. For such instances c_{jl} the value $\gamma_{jl}(c_{jl})$ is set by Eq. 6 to a finite value since $\Psi_i(d_i) > 0$ for every instance d_i which is a projection of x on a set D_i, and $\Phi_k(c_k) = 0$ implies $Q(c_k|c_{jl}) = 0$. Therefore, $\Phi_{jl}(c_{jl}) = e^{\gamma_{jl}(c_{jl})} > 0$ and $Q(x) > 0$.□

The consequence of Theorem 3 is that $Q(x) = 0$ iff $P(x) = 0$. Conditions weaker than containability may be sufficient to ensure the requirement needed for convergence, however, containability is easily satisfiable in applications of variational techniques as explicated in the next section.

6. Genetic Linkage Analysis via Variational Algorithms

Genetic linkage analysis takes as input a family pedigree in which some individuals are affected with a genetic disease, affection status of members of the pedigree, marker readings across the genome, and mode of inheritance. The output is the likelihood of data as a function of the location of a disease gene and the given pedigree. Locations yielding maximum or close to maximum likelihood are singled out as suspect regions for further scrutiny. The exact computation of this likelihood is often too complex and approximations are needed.

Algorithm VIP* has been developed to facilitate such likelihood computations. In particular, VIP* allows there to be overlapping clusters which minimizes the loss of valuable information and, more importantly, can handle the deterministic constraints that are common in these models. In this section, we describe the standard probabilistic model for genetic linkage, several approximate distributions that we use when applying VIP* to the genetic linkage model, and demonstrate VIP* on a real-world data set of a large pedigree with 115 individuals.

The standard probabilistic model for genetic linkage is based on a pedigree which contains several variables for each person at each location and conditional probability tables for each variable X_m given a set of variables called *parents* of X_m and denoted by $\pi(X_m)$. The distribution $P(x)$ that represents the joint distribution of the variables in the pedigree is written using multiple indices; one set of indices for persons (i), one for loci (j), and another for the type of variable (t) as follows:

$$P(x) = \prod_j \prod_i \prod_{t\in\{ps,ms,pg,mg,f\}} P(x_j^{i,t}|\pi(x_j^{i,t})) = \qquad (15)$$

$$\frac{1}{Z_P}\prod_j \prod_i \prod_{t\in\{ps,ms,pg,mg,f\}} \Psi_j^{i,t}(x_j^{i,t}|\pi(x_j^{i,t}))$$

where the five possible types of variables are: paternal selector (*ps*), maternal selector (*ms*), paternal genotype (*pg*), maternal genotype (*mg*) and phenotype (*f*). Thus, the set $D_j^{i,t}$ equals $\{X_j^{i,t}, \pi(X_j^{i,t})\}$.

We denote the variables of the different types, *ps*, *ms*, *pg*, *mg* and *f*, of individual i at locus j by $S_j^{i,p}, S_j^{i,m}, G_j^{i,p}, G_j^{i,m}$ and F_j^i respectively. In this notation the possible potentials of P are $\Psi(G_j^{i,p}, G_j^{a,p}, G_j^{a,m}, S_j^{i,p}), \Psi(G_j^{i,m}, G_j^{b,p}, G_j^{b,m}, S_j^{i,m}), \Psi(S_j^{i,p}, S_{j-1}^{i,p}), \Psi(S_j^{i,m}, S_{j-1}^{i,m})$ and $\Psi(F_j^i, G_j^{i,p}, G_j^{i,m})$ where a and b are i's father and mother in the pedigree, respectively. Some exemplifying sets are $D_j^{i,pg} = \{G_j^{i,p}, S_j^{i,p}, G_j^{a,p}, G_j^{a,m}\}$, $D_j^{i,ms} = \{S_j^{i,m}, S_{j-1}^{i,m}\}$, and $D_j^{i,f} = \{F_j^i, G_j^{i,p}, G_j^{i,m}\}$, where a is the father of individual i in the pedigree. We note that the first two types of potentials and possibly the last one are deterministic potentials which equal zero for some instantiations.

A directed acyclic graph along with a probability distribution R that factors according to $R(Z) = \prod_i R(z_i|\pi(z_i))$ is called a *Bayesian network*. A Bayesian network defined by Eq. 15, that describes parents-offspring interaction in a simple genetic analysis problem with two siblings and their parents across 3 loci, is given in Figure 4. The dashed boxes contain all variables that describe the variables in a single location. In this example we assume that each phenotype variable depends on the genotype at a single locus. This is reflected by the fact that only edges from a single locus point into each phenotype variable. The full semantics of all variables and more details regarding the conditional probability tables can be found in the paper by Fishelson & Geiger (2002); these details are not needed here.

We use several choices to cluster the Bayesian network for $P(x)$ such that Q is self-compatible and compatible wrt P. In addition, since some potentials in P are constrained (e.g. $\Psi_j^{i,pg}$), we choose clusters such that Q is containable wrt P. According to Theorem 2 this choice ensures that Q satisfies all conditions necessary for convergence of VIP*, and in particular $(P(x) = 0) \Rightarrow (Q(x) = 0)$.

Consider a partition of the network into slots, each containing a set of consecutive loci. In the most simple case every slot is a single locus and each of the subsets C_{ji} contains variables related to one individual i and the genotypes of his parents in that slot. We set $C_{ji} = \{G_j^{\rho(i)}, S_j^i\}$ and $C_j = \bigcup_i \{C_{ji}\}$ where $\rho(i)$ denotes the union of i and his parents. An illustration of such a setting in a pedigree of two siblings and their parents over three loci is given in Figure 5. In this setting, self-compatibility is trivially satisfied because the clusters C_j of Q are disjoint, and Q is containable wrt P since only sets $D_j^{i,ps}$ and $D_j^{i,ms}$, the potentials of which are not constrained, span across more than a single locus. It remains to show that compatibility of Q wrt P is satisfied. For sets $D_j^{i,t}$ contained in a single subset C_{ji} this is trivial as $Q(D_j^{i,t}|c_j) = Q(D_j^{i,t}|c_{ji}) = 1$. Otherwise, t equals *ps* or *ms* and without loss of generality, $Q(S_{j-1}^{i,p}, S_j^{i,p}|c_j) = Q(S_{j-1}^{i,p}|c_j) = Q(S_{j-1}^{i,p}|c_{ji}) = Q(S_{j-1}^{i,p}, S_j^{i,p}|c_{ji})$.

In a more complex setup, which is similar to Example 1, we add a cluster C_{J+1} which cuts across the loci for selector variables of an individual r, as shown in Figure 6. The subset C_{ji} are set to $C_{ji} = \{G_j^{\rho(i)}, S_j^i, S_j^r\}$ and the clusters are $C_j = \bigcup_i \{C_{ji}\}$, for $j = 1 \dots J$. In addition, we set $C_{J+1} = \bigcup_l \{C_{J+1,l}\}$ and the subsets $C_{J+1,l} = \{S_{l,l+1}^r\}$, for a single chosen individual r. We verify that Q still satisfies the conditions of Theorem 1. Self-compatibility is maintained since $Q(C_j|c_{J+1}) = Q(C_j|c_{J+1,j})$, $Q(C_{J+1}|c_j) = Q(C_{J+1}|c_{jr})$,

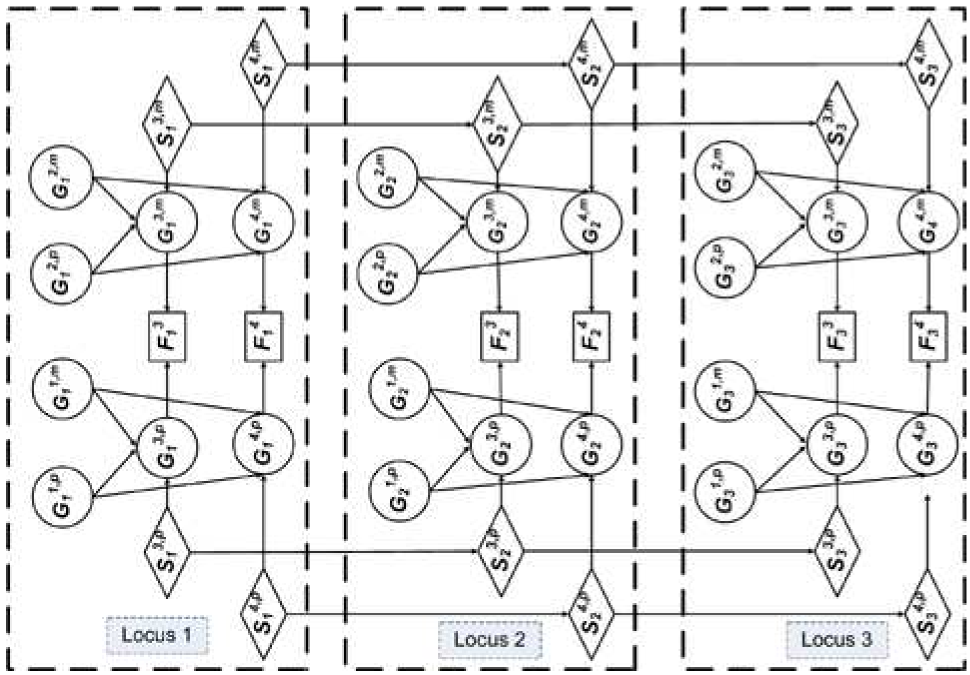


Figure 4: A Bayesian network representation of a pedigree of two siblings and their parents in a 3-loci model. The circles, squares and diamond shapes represent genotype, phenotype and selector variables respectively.

and $Q(C_k|c_j) = Q(C_k|c_{jr})$ for every two clusters C_j, C_k such that $j, k \leq J$. Sets $D_j^{i,t}$ such that $t \neq ms, ps$ are contained in cluster C_j and thus maintain independence given a subset. For $t = ms, ps$ the sets $D_j^{i,t}$ that connect two adjacent loci are independent of C_{J+1} given $C_{J+1,j}$, and are independent of other clusters C_j given the subset C_{ji}, maintaining compatibility of Q wrt P. Finally, Q is containable wrt P since all clusters from the previous option remain.

Immediate extensions of the above clustering schemes allow every slot to contain several consecutive loci and a set of possibly more than one individual R to cut across the loci. To maintain the compatibility of Q wrt P in the latter extension, the subsets C_{jR} are set to $C_{jR} = \{G_j^{\rho(R)}, S_j^R\}$, where $\rho(R)$ denotes the union of individuals in R and their parents.

We now describe the experiments performed using a large pedigree with 115 individuals spanning across 21 locations, which was studied by Narkis et al. (2004) to locate an area that contains a gene that causes a fatal neurological disorder (LCCS type 2). First, the proximity of the disease gene to each of the markers was tested through two-point analysis - a model-selection method in which only two loci are considered simultaneously, with one of them being the disease locus. In this method, loci which yield likelihood maxima suggest probable locations of the disease locus. Two-point analysis of the abovementioned pedigree took several days using the exact inference software superlink v1.5 designed for genetic linkage analysis.

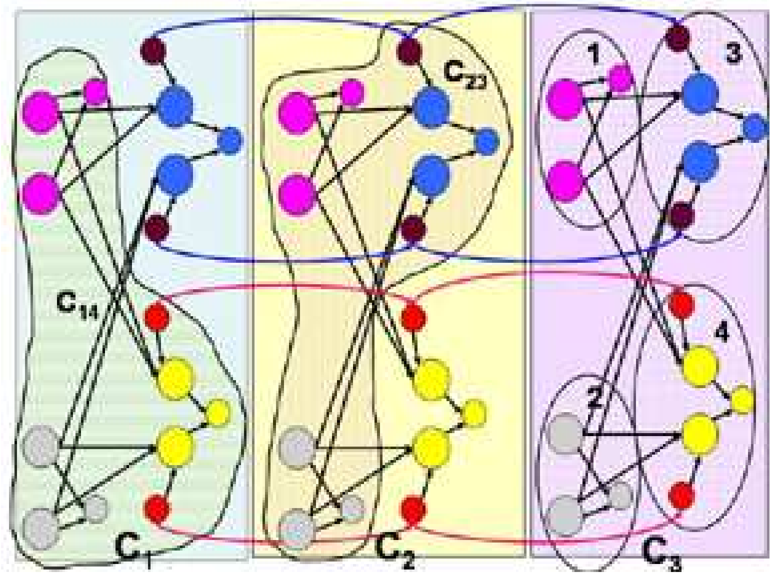


Figure 5: A schematic division of a pedigree into clusters, where each locus is a cluster. In cluster C_3 the variables of every individual are in a separate ellipse and the number of that individual is written. In the other two clusters the marked areas are C_{14} and C_{23}.

The log-likelihood probabilities obtained by VIP* using the abovementioned extended clustering scheme with 5 clusters across all loci and with no cluster cutting across the loci, are shown in Figure 7. The figure shows clearly that the exact and approximate log-likelihood curves have a similar shape and almost identical extremum points, but have a major difference in absolute value.

Next, we tested VIP* on three-point analysis problems on the same pedigree, where two markers are considered simultaneously along with one disease locus. We note that exact inference for this task on the specified pedigree is hard on a single PC, taking several weeks. Since the exact location of the disease gene is known with high probability, we wished to test whether or not the lower-bounds found by VIP* indicate this location. We considered two nearby markers (number 4 and 6) and seven models which differ by the location of the speculated disease gene: in the first two, the disease locus was positioned left to marker 4 at distances 0.01 and 0.02 centi-Morgan (cM), and in the remaining five it was positioned to the right of marker 4 at distances 0.01 to 0.05 cM with 0.01 cM difference between locations. The location of the disease gene is 0.01cM to the right of marker 4. The algorithm was run three times on each model with random initial potentials, taking into account only the maximum value obtained. The results of this test are plotted in Figure 8 versus the approximation found by the sampling software SIMWALK2 introduced by Sobel, Papp & Lange (2002) which is designed for approximate pedigree analysis. As shown, the probabilities found by VIP* are higher as we approach the location of the disease gene. The same in not true for the probabilities found by SIMWALK2. However, we note that VIP* is

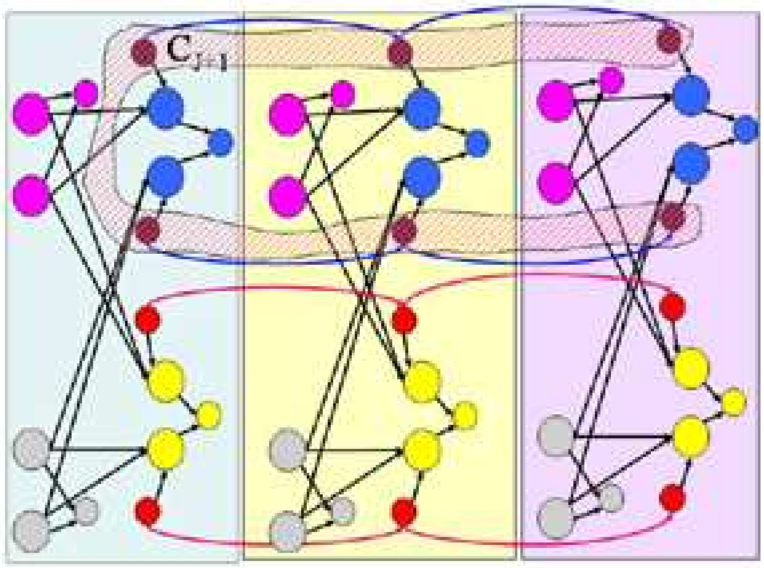


Figure 6: A pedigree partitioned into clusters, where each locus is a cluster and an additional cluster C_4, in the striped area, contains the variables of one of the individuals.

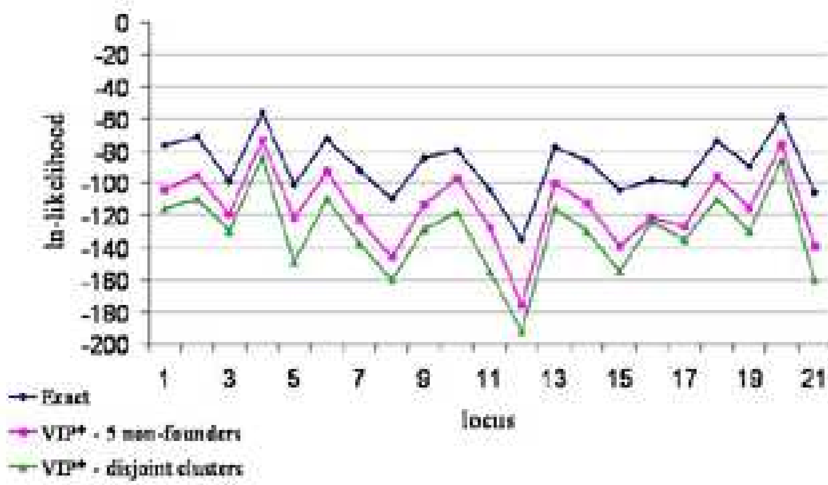
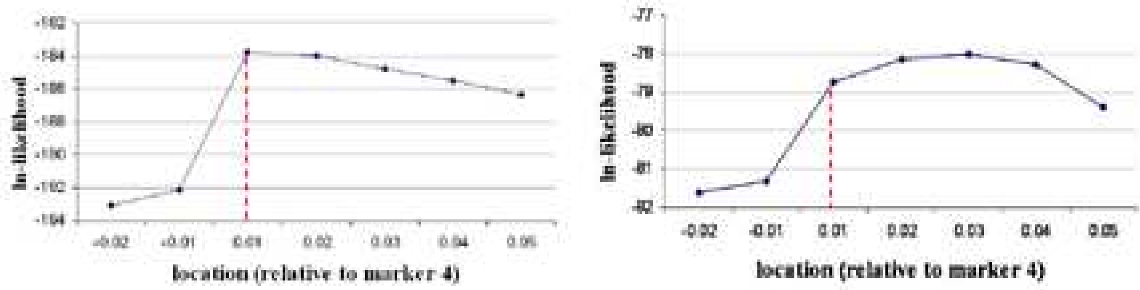


Figure 7: Approximations to the log-likelihood of two-point analysis using VIP*.



(a) using VIP* (b) using SIMWALK2

Figure 8: Approximations to the log-likelihood of three-point analysis.

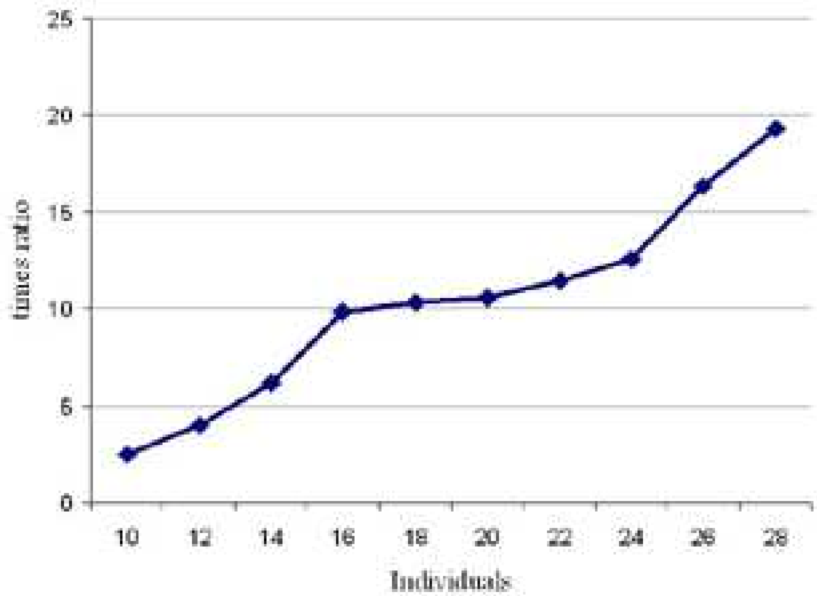


Figure 9: Speedup of VIP* over Wiegerinck's algorithm.

much slower on this problem than SIMWALK2, taking several hours for each run. In addition, we note that the ln-likelihood probabilities in Figures 8(a) and (b) are drawn on different scales due to the markedly different output of the two methods.

We compared the convergence times of VIP* and Wiegerinck's algorithm on various size problems of genetic linkage analysis. The original network on which we performed the test includes 332 variables and represents a pedigree with 28 individuals over four loci. To create various sized problems, subsets of the original pedigree with increasing number of individuals were considered. In addition, for a fair comparison, the clusters of Wiegerinck's algorithm were chosen to be a subset of the subsets used by VIP*. Since the number of iterations until convergence did not vary significantly between the two algorithms, we report the ratio of iteration times which also tests the theoretical speedup predicted in Section 3 of VIP* over Wiegerinck's algorithm. Figure 9 illustrates the ratio of time for an update iteration of the

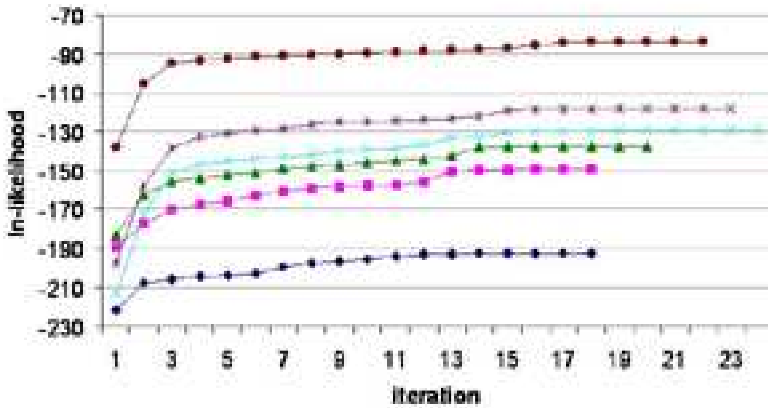


Figure 10: Convergence of VIP* for 6 runs of two-point analysis.

two algorithms, where it is evident that the ratio increases linearly with the problem size. The ratios indicated are averaged over 5 runs each with 10 iterations for every problem size.

Finally we examine the convergence of VIP* in Figure 10 using six representatives of the original 21 two-point analysis runs described earlier, each on a different network. The algorithm is halted when the change in the lower-bound is smaller than 10^{-5} for more than 3 iterations. Although not all runs converge at the same rate, it seems that they obey a certain pattern of convergence where the first few iterations show significant improvements in the lower-bound, followed by slow convergence to a local maximum, and then another moderate improvement to a better maximum point.

7. Discussion

In this paper we present an efficient algorithm called VIP* for structured variational approximate inference. This algorithm, which extends known algorithms, can handle overlapping clusters and overcome the difficulties imposed by deterministic constraints. We show that for $N \times N$ grid-like models, algorithm VIP* is N fold faster than Wiegerinck's algorithm, when a junction tree is used. In addition, we prove the convergence of VIP* and of previous variational methods via a novel proof method, using properties of the KL divergence.

Finally, algorithm VIP* is tested on Bayesian networks that model genetic linkage analysis problems. These graphs resemble grid-like models and are notoriously difficult to approximate due to the numerous deterministic constraints. The results show a linear improvement in speed of VIP* versus Wiegerinck's algorithm, and that the approximation follows the shape of the real likelihood probabilities. Nevertheless, Figure 7 shows that variational methods such as Wiegerinck's algorithm or VIP* are still not appropriate to produce accurate approximation of the likelihood for genetic linkage analysis.

Acknowledgments

This paper is an extension of a paper that originally appeared at the 10^{th} workshop on Artificial Intelligence and Statistics (Geiger & Meek 2005). We thank D. Heckerman, N.

Jojic and V. Jojic for helpful discussions. We also thank the two anonymous reviewers for correcting several errors that appeared in the early version as well as improving the presentation. Part of the work was done while the first author was a visitor at Microsoft Research. This work is supported by the Israeli Science Foundation and the Israeli Science Ministry.

References

Bishop, C. & Winn, J. (2003). Structured variational distributions in VIBES. In *Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.

Cooper, G. (1990). Probabilistic inference using belief networks is NP-hard. *Artificial Intelligence*, *42*, 393–405.

Cover, T. M. & Thomas, J. A. (1991). *Elements of Information Theory*. Wiley.

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

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

Fishelson, M. & Geiger, D. (2002). Exact genetic linkage computations for general pedigrees. *Bioinformatics*, *18*, S189–S198.

Geiger, D. & Meek, C. (2005). Structured variational inference procedures and their realizations. In *Proceedings of Tenth International Workshop on Artificial Intelligence and Statistics,* The Barbados. The Society for Artificial Intelligence and Statistics.

Ghahramani, Z. & Jordan, M. I. (1997). Factorial hidden Markov models. *Machine Learning*, *29*, 245–273.

Jensen, F. V. (1996). *An Introduction to Bayesian Networks*. Springer.

Jojic, V., Jojic, N., Meek, C., Geiger, D., Siepel, A., Haussler, D., & Heckerman, D. (2004). Efficient approximations for learning phylogenetic HMM models from data. *Bioinformatics*, *20*, 161–168.

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, *47*(2), 498–519.

Narkis, G., Landau, D., Manor, E., Elbedour, K., Tzemach, A., Fishelson, M., Geiger, D., Ofir, R., Carmi, R., & Birk, O. (2004). Homozygosity mapping of lethal congenital contractural syndrome type 2 (LCCS2) to a 6 cM interval on chromosome 12q13. *American Journal of Medical Genetics*, *130*(3), 272–276.

Saul, L. & Jordan, M. I. (1996). Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Sobel, E., Papp, J., & Lange, K. (2002). Detection and integration of genotyping errors in statistical genetics. *American Journal of Human Genetics*, *70*, 496–508.

Wiegerinck, W. (2000). Variational approximations between mean field theory and the junction tree algorithm. In *Uncertainty in Artificial Intelligence*, (pp. 626–633). Morgan Kaufmann.

Xing, E. P., Jordan, M. I., & Russell, S. (2003). A generalized mean field algorithm for variational inference in exponential families. In *Uncertainty in Artificial Intelligence*, (pp. 583–591). Morgan Kaufmann.

Xing, E. P., Jordan, M. I., & Russell, S. (2004). Graph partition strategies for generalized mean field inference. In *Uncertainty in Artificial Intelligence*, (pp. 602 – 610). Morgan Kaufmann.