PGM Seminar
Inference as Optimization

Daniel Carmon

Background: Exact Inference via Calibration

Algorithm 10.2 Calibration using sum-product message passing in a clique tree

Procedure CTree-SP-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)

1 Initialize-Cliques
2 **while** exist i, j such that i is ready to transmit to j
3 $\quad \delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow$ SP-Message(i, j)
4 **for** each clique i
5 $\quad \beta_i \leftarrow \psi_i \cdot \prod_{k \in \text{Nb}_i} \delta_{k \to i}$
6 **return** $\{\beta_i\}$

Procedure Initialize-Cliques (
)

1 **for** each clique $\boldsymbol{C}_i$
2 $\quad \psi_i(\boldsymbol{C}_i) \leftarrow \prod_{\phi_j \,:\, \alpha(\phi_j)=i} \phi_j$
3

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)

1 $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$
3 **return** $\tau(\boldsymbol{S}_{i,j})$

Background: Exact Inference via Calibration

At the end of algorithm run:

$$\beta_i = \psi_i \cdot \prod_{k \in \mathrm{Nb}_i} \delta_{k \to i}.$$

$$\begin{aligned}
\mu_{i,j}(\boldsymbol{S}_{i,j}) &= \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i) \\
&= \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k \in \mathrm{Nb}_i} \delta_{k \to i} \\
&= \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \delta_{j \to i} \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i} \\
&= \delta_{j \to i} \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i} \\
&= \delta_{j \to i} \delta_{i \to j},
\end{aligned}$$

Background: Exact Inference via Calibration

What was all this good for?

Calculates cluster marginals for each node in clique tree, using only two tree scans.

Correctness & no deadlocks since working over a clique tree.

So are we done?

Complexity of Calibrating a Clique Tree

Problems:

Clique tree messages can be *very* expensive.

Canonical example:

The Complete Bipartite Graph (as a pairwise markov model).

How would a clique-tree for a complete bipartite graph look like?

Complexity of Calibrating a Clique Tree

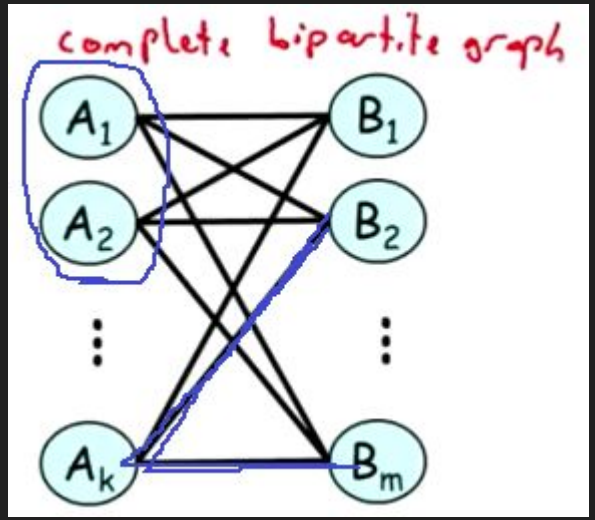
Remember that in clique-trees, each edge (i,j) induces a disjoint decomposition of the random variables to 3 sets: Wij+Wji+Sij, where (Wij⊥Wji|Sij).

In a complete bipartite graph,
given a set of rvs, the other rvs are not independent.

(X⊥Y|Z) iff X=Z or Y=Z.

We convince ourselves this results in sepsets, and/or cliques of size O(n).

Messages thus exponential in n.



Complexity of Calibrating a Clique Tree

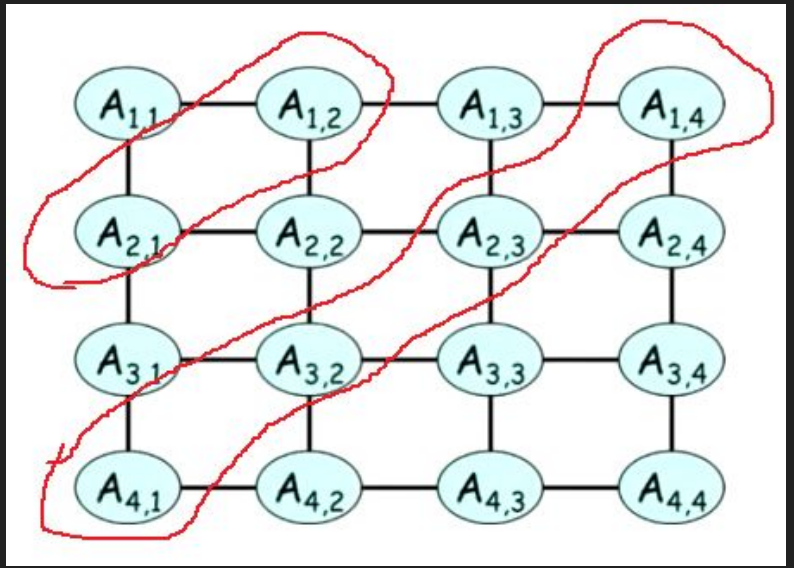
Another example: the nxn grid.

How would a sepset separate k rvs?

Takes at least min(k,n^2-k) rvs in the sepset.

Eventually we'll have sepsets of size O(n).

Messages exponential in n.



Why These Problems?

These difficulties are somewhat inherent.

Algorithm Performs *exact* inference, an NP-Hard problem.

We want to consider cheap “approximations” .

NP-Hardness reminder:

Relative error approx is also NP-Hard, and so is an additive error approx for queries with evidence.

But, cases for which exact inference is hard aren’t necessarily hard to approx.

How to Approximate?

Maybe approximate by using general cluster graphs (avoids previous problems)? maybe something completely different?

Need to develop theory to evaluate, design and understand these approximations.

Approximation methods usually divide to optimization based methods and sampling based methods (next week).

Before describing various optimization based methods, let’s revisit exact inference based on clique tree calibration, and try to understand how this corresponds to an optimization problem.

A Calibrated Clique Tree as a Distribution

We define the measure *induced by a calibrated tree* $\mathcal{T}$ *to be:*

$$Q_{\mathcal{T}} = \frac{\prod_{i \in \mathcal{V}_{\mathcal{T}}} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j) \in \mathcal{E}_{\mathcal{T}}} \mu_{i,j}(\boldsymbol{S}_{i,j})},$$

where

$$\mu_{i,j} = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j(\boldsymbol{C}_j).$$

Let $\mathcal{T}$ *be a clique tree over* Φ*, and let* $\beta_i(\boldsymbol{C}_i)$ *be a set of calibrated potentials for* $\mathcal{T}$*. Then,* $\tilde{P}_\Phi(\mathcal{X}) \propto Q_{\mathcal{T}}$ *if and only if, for each* $i \in \mathcal{V}_{\mathcal{T}}$*, we have that* $\beta_i(\boldsymbol{C}_i) \propto \tilde{P}_\Phi(\boldsymbol{C}_i)$.

Thus, we can view the clique tree as an alternative representation of the joint measure, one that directly reveals the clique marginals.

Let $\mathcal{T}$ be a clique tree over Φ, and let $\beta_i(\boldsymbol{C}_i)$ be a set of calibrated potentials for $\mathcal{T}$. Then, $\tilde{P}_\Phi(\mathcal{X}) \propto Q_\mathcal{T}$ if and only if, for each $i \in \mathcal{V}_\mathcal{T}$, we have that $\beta_i(\boldsymbol{C}_i) \propto \tilde{P}_\Phi(\boldsymbol{C}_i)$.

PROOF Let r be any clique in $\mathcal{T}$, which we choose to be the root. Define the descendant cliques of a clique $\boldsymbol{C}_i$ to be the cliques that are downstream from $\boldsymbol{C}_i$ relative to $\boldsymbol{C}_r$; the nondescendant cliques are then the remaining cliques (other than $\boldsymbol{C}_i$). Let $\boldsymbol{X}$ be the variables in the scope of the nondescendant cliques. It follows immediately from theorem 10.2 that

$$\tilde{P}_\Phi \models (\boldsymbol{C}_i \perp \boldsymbol{X} \mid \boldsymbol{S}_{i,p_r(i)}).$$

From this, we obtain, using the standard chain-rule argument, that:

$$\tilde{P}_\Phi(\mathcal{X}) = \tilde{P}_\Phi(\boldsymbol{C}_r) \cdot \prod_{i \neq r} \tilde{P}_\Phi(\boldsymbol{C}_i \mid \boldsymbol{S}_{i,p_r(i)}).$$

We can rewrite equation (10.11) as a similar product, using the same root:

$$Q_\mathcal{T}(\mathcal{X}) = \beta_r(\boldsymbol{C}_r) \cdot \prod_{i \neq r} \beta_i(\boldsymbol{C}_i \mid \boldsymbol{S}_{i,p_r(i)}).$$

The "if" direction now follows from direct substitution of β_i for each $\tilde{P}_\Phi(\boldsymbol{C}_i)$.

To prove the "only if" direction, we note that each of the terms $\beta_i(\boldsymbol{C}_i \mid \boldsymbol{S}_{i,p_r(i)})$ is a conditional distribution; hence, if we marginalize out the variables not in $\boldsymbol{C}_r$ in the distribution $Q_\mathcal{T}$, each of these conditional distributions marginalizes to 1, and so we are left with $Q_\mathcal{T}(\boldsymbol{C}_r) = \beta_r(\boldsymbol{C}_r)$. It now follows that if $\tilde{P}_\Phi \propto Q_\mathcal{T}$, then $\tilde{P}_\Phi(\boldsymbol{C}_r) \propto Q_\mathcal{T}(\boldsymbol{C}_r) = \beta_r(\boldsymbol{C}_r)$. Because this argument applies to any choice of root clique, we have proved that this equality holds for every clique. ■

Exact Inference Revisited

Given this representation, the problem of exact inference can be viewed as searching for a calibrated set Q of beliefs for cliques and sepsets, which induces a distribution that equals our original normalized distribution.

Exact Inference Revisited

The problem of exact inference can be viewed as searching for a calibrated set Q of beliefs for cliques and sepsets, which induces a distribution that equals our original normalized distribution.

Intuitively, we can search for a calibrated set of beliefs which induces a distribution *closest* to our original one.

(Already the problem gets an optimization “flavor”).

Suppose now we do so,

How should we measure “distance” between two probability distributions?

Choosing a Distance Measure for Distributions

On information-theoretic grounds, it is best in this context to use the relative entropy:

Recall that the relative entropy between P_1 and P_2 is defined as

$$\boldsymbol{D}(P_1 \| P_2) = \boldsymbol{E}_{P_1}\left[\ln \frac{P_1(\mathcal{X})}{P_2(\mathcal{X})}\right].$$

Also recall that the relative entropy is always nonnegative, and equal to 0 if and only if $P_1 = P_2$.

This distance measure is generally not symmetric, so there are two different measures we can use:

D(P|Q)

D(Q|P)

I-Projection and M-Projection

We want to find a distribution Q (induced from a calibrated belief set for our tree) which is closest to the original distribution P, in the sense of relative entropy.

Which option to use? maybe a mix?

I-projection

- *The* I-projection *(information projection) of* P *onto* $\mathcal{Q}$ *is the distribution*

$$Q^I = \arg\min_{Q \in \mathcal{Q}} \boldsymbol{D}(Q \| P).$$

M-projection

- *The* M-projection *(moment projection) of* P *onto* $\mathcal{Q}$ *is the distribution*

$$Q^M = \arg\min_{Q \in \mathcal{Q}} \boldsymbol{D}(P \| Q).$$

I-Projection and M-Projection

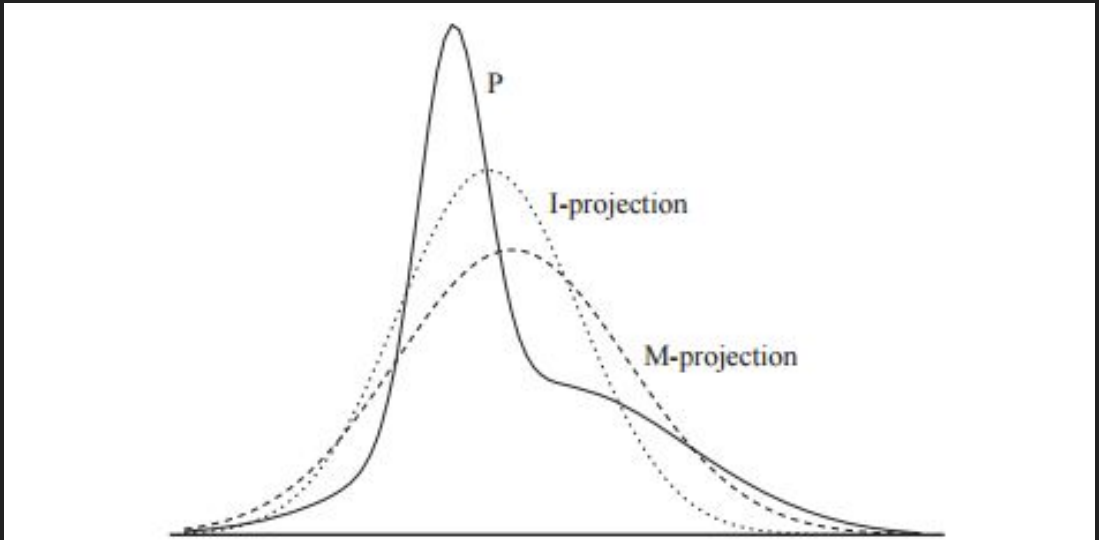


Figure 8.1 Example of M- and I-projections into the family of Gaussian distributions

I-Projection and M-Projection

Again from information-theoretic justifications, it is generally better to use the M-Projection.

But apparently (section 8.5 of KF[09]), computing an M-Projection of P is equivalent to computing probability marginals, so if we later generalize using this direction, we could have problems.

We thus resort to using the function D(Q|P) as our distance measure.

Exact Inference as Optimization - Take I

With these definitions in hand, we can now view exact inference as maximizing $-\boldsymbol{D}(Q\|P_\Phi)$ over the space of calibrated sets $\boldsymbol{Q}$.

CTree-Optimize-KL:

Find $\boldsymbol{Q} = \{\beta_i : i \in \mathcal{V}_T\} \cup \{\mu_{i,j} : (i\text{-}j) \in \mathcal{E}_T\}$

maximizing $-\boldsymbol{D}(Q\|P_\Phi)$

subject to

$$\mu_{i,j}[\boldsymbol{s}_{i,j}] = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{c}_i) \quad \forall (i\text{-}j) \in \mathcal{E}_T, \forall \boldsymbol{s}_{i,j} \in Val(\boldsymbol{S}_{i,j})$$

$$\sum_{\boldsymbol{c}_i} \beta_i(\boldsymbol{c}_i) = 1 \quad \forall i \in \mathcal{V}_T.$$

The Energy Functional

a-priori, D(Q|P) may also seem intractable to directly use, since it includes summation over all possible assignments to rv's.

We Introduce the first simplification, the Energy Functional.

Theorem:

$$\boldsymbol{D}(Q\|P_\Phi) = \ln Z - F[\tilde{P}_\Phi, Q]$$

where $F[\tilde{P}_\Phi, Q]$ *is the* energy functional

$$F[\tilde{P}_\Phi, Q] = \boldsymbol{E}_Q\left[\ln \tilde{P}(\mathcal{X})\right] + \boldsymbol{H}_Q(\mathcal{X}) = \sum_{\phi\in\Phi} \boldsymbol{E}_Q[\ln \phi] + \boldsymbol{H}_Q(\mathcal{X}).$$

The Energy Functional

Proof:

$$\boldsymbol{D}(Q\|P_\Phi) = \boldsymbol{E}_Q[\ln Q(\mathcal{X})] - \boldsymbol{E}_Q[\ln P_\Phi(\mathcal{X})]. \tag{11.4}$$

Using the product form of P_Φ, we have that

$$\ln P_\Phi(\mathcal{X}) = \sum_{\phi\in\Phi} \ln \phi(\boldsymbol{U}_\phi) - \ln Z.$$

Moreover, recall that $\boldsymbol{H}_Q(\mathcal{X}) = -\boldsymbol{E}_Q[\ln Q(\mathcal{X})]$. Plugging these into equation (11.4), we get

$$\begin{aligned}\boldsymbol{D}(Q\|P_\Phi) &= -\boldsymbol{H}_Q(\mathcal{X}) - \boldsymbol{E}_Q\left[\sum_{\phi\in\Phi} \ln \phi(\boldsymbol{U}_\phi)\right] + \boldsymbol{E}_Q[\ln Z] \\ &= -F[\tilde{P}_\Phi, Q] + \ln Z.\end{aligned}$$

■

Optimizing the Energy Functional

The term ln(Z) doesn’t depend on the induced distribution Q.

Therefore:

Minimizing the relative entropy = Maximizing the energy functional.

Notice the E.F has two terms, a expectations of factors term, and an entropy term.

Still contains terms which involve summation over all possible assignments.

Intuitively, we should perhaps simplify further.

The Factored Energy Functional

Given a cluster tree $\mathcal{T}$ with a set of beliefs $\boldsymbol{Q}$ and an assignment α that maps factors in P_Φ to clusters in $\mathcal{T}$, we define the factored energy functional:

$$\tilde{F}[\tilde{P}_\Phi, \boldsymbol{Q}] = \sum_{i \in \mathcal{V}_\mathcal{T}} \boldsymbol{E}_{\boldsymbol{C}_i \sim \beta_i}[\ln \psi_i] + \sum_{i \in \mathcal{V}_\mathcal{T}} \boldsymbol{H}_{\beta_i}(\boldsymbol{C}_i) - \sum_{(i-j) \in \mathcal{E}_\mathcal{T}} \boldsymbol{H}_{\mu_{i,j}}(\boldsymbol{S}_{i,j}), \tag{11.6}$$

An important reformulation of the original functional.

Notice the components of this functional are *local*.

Terms depend only on current beliefs, not on all possible events.

The Factored Energy Functional

Theorem:

If $\mathbf{Q}$ *is a set of calibrated beliefs for* $\mathcal{T}$*, and* Q *is defined by equation (11.2), then*

$$\tilde{F}[\tilde{P}_\Phi, \mathbf{Q}] = F[\tilde{P}_\Phi, Q].$$

Proof: hw

This means that if we optimize beliefs over a tree, we can use this simplified functional, without losing any accuracy.

Exact Inference as Optimization - Take II

Our target function is now the factorized energy functional.

CTree-Optimize:

Find $\boldsymbol{Q} = \{\beta_i : i \in \mathcal{V}_T\} \cup \{\mu_{i,j} : (i\text{–}j) \in \mathcal{E}_T\}$

maximizing $\tilde{F}[\tilde{P}_\Phi, \boldsymbol{Q}]$

subject to

$$\mu_{i,j}[\boldsymbol{s}_{i,j}] = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{c}_i) \qquad (11.7)$$

$$\forall (i\text{–}j) \in \mathcal{E}_T, \forall \boldsymbol{s}_{i,j} \in Val(\boldsymbol{S}_{i,j})$$

$$\sum_{\boldsymbol{c}_i} \beta_i(\boldsymbol{c}_i) = 1 \qquad \forall i \in \mathcal{V}_T \qquad (11.8)$$

(break)

Characterizing the Optimal Belief Set

The factored energy functional enables us to tackle this conveniently constrained optimization problem via Lagrange Multipliers.

We define the following Lagrangian:

$$
\begin{aligned}
\mathcal{J} \;=\; & \tilde{F}[\tilde{P}_\Phi, \boldsymbol{Q}] \\
& - \sum_{i \in \mathcal{V}_\mathcal{T}} \lambda_i \left(\sum_{\boldsymbol{c}_i} \beta_i(\boldsymbol{c}_i) - 1 \right) \\
& - \sum_{i} \sum_{j \in \mathrm{Nb}_i} \sum_{\boldsymbol{s}_{i,j}} \lambda_{j \to i}[\boldsymbol{s}_{i,j}] \left(\sum_{\boldsymbol{c}_i \sim \boldsymbol{s}_{i,j}} \beta_i(\boldsymbol{c}_i) - \mu_{i,j}[\boldsymbol{s}_{i,j}] \right)
\end{aligned}
$$

Characterizing the Optimal Belief Set

The partial derivatives of this Lagrangian are:

$$\frac{\partial}{\partial \beta_i(\boldsymbol{c}_i)}\mathcal{J} = \ln \psi_i[\boldsymbol{c}_i] - \ln \beta_i(\boldsymbol{c}_i) - 1 - \lambda_i - \sum_{j \in \mathrm{Nb}_i} \lambda_{j \to i}[\boldsymbol{s}_{i,j}]$$

$$\frac{\partial}{\partial \mu_{i,j}[\boldsymbol{s}_{i,j}]}\mathcal{J} = \ln \mu_{i,j}[\boldsymbol{s}_{i,j}] + 1 + \lambda_{i \to j}[\boldsymbol{s}_{i,j}] + \lambda_{j \to i}[\boldsymbol{s}_{i,j}].$$

\+ The original optimization constraints (from deriving w.r.t constraint variables) .

Characterizing the Optimal Belief Set

After equating to zero, rearranging terms, and exponentiating, we get:

$$\begin{aligned}\beta_i(\boldsymbol{c}_i) &= \exp\{-1-\lambda_i\}\,\psi_i[\boldsymbol{c}_i]\prod_{j\in \mathrm{Nb}_i}\exp\{-\lambda_{j\to i}[\boldsymbol{s}_{i,j}]\}\\ \mu_{i,j}[\boldsymbol{s}_{i,j}] &= \exp\{-1\}\exp\{-\lambda_{i\to j}[\boldsymbol{s}_{i,j}]\}\exp\{-\lambda_{j\to i}[\boldsymbol{s}_{i,j}]\}\,.\end{aligned}$$

hmm.. familiar?

Background: Exact Inference via Calibration

At the end of algorithm run:

$$\beta_i = \psi_i \cdot \prod_{k \in \mathrm{Nb}_i} \delta_{k \to i}.$$

$$\begin{aligned}
\mu_{i,j}(\boldsymbol{S}_{i,j}) &= \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i) \\
&= \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k \in \mathrm{Nb}_i} \delta_{k \to i} \\
&= \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \delta_{j \to i} \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i} \\
&= \delta_{j \to i} \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i} \\
&= \delta_{j \to i} \delta_{i \to j},
\end{aligned}$$

Characterizing the Optimal Belief Set

After equating to zero, rearranging terms, and exponentiating, we get:

$$\begin{aligned}\beta_i(\boldsymbol{c}_i) &= \exp\{-1-\lambda_i\}\,\psi_i[\boldsymbol{c}_i]\prod_{j\in \mathrm{Nb}_i}\exp\{-\lambda_{j\to i}[\boldsymbol{s}_{i,j}]\}\\ \mu_{i,j}[\boldsymbol{s}_{i,j}] &= \exp\{-1\}\exp\{-\lambda_{i\to j}[\boldsymbol{s}_{i,j}]\}\exp\{-\lambda_{j\to i}[\boldsymbol{s}_{i,j}]\}\,.\end{aligned}$$

hmm.. familiar?

Define:

$$\delta_{i\to j}[\boldsymbol{s}_{i,j}] = \exp\left\{-\lambda_{i\to j}[\boldsymbol{s}_{i,j}] - \frac{1}{2}\right\}.$$

Characterizing the Optimal Belief Set

Combining this with the calibration constraint:

$$
\begin{aligned}
\delta_{i\to j}[\boldsymbol{s}_{i,j}] &= \frac{\mu_{i,j}[\boldsymbol{s}_{i,j}]}{\delta_{j\to i}[\boldsymbol{s}_{i,j}]} \\
&= \frac{\sum_{\boldsymbol{c}_i \sim \boldsymbol{s}_{i,j}} \beta_i(\boldsymbol{c}_i)}{\delta_{j\to i}[\boldsymbol{s}_{i,j}]} \\
&= \exp\left\{-\lambda_i - 1 + \frac{1}{2}|\mathrm{Nb}_i|\right\} \sum_{\boldsymbol{c}_i \sim \boldsymbol{s}_{i,j}} \psi_i(\boldsymbol{c}_i) \prod_{k \in \mathrm{Nb}_i - \{j\}} \delta_{k\to i}[\boldsymbol{s}_{i,k}].
\end{aligned}
$$

Each “message” is defined in terms of other “messages”.

Characterizing the Optimal Belief Set

Theorem:

A set of beliefs $\boldsymbol{Q}$ *is a stationary point of* CTree-Optimize *if and only if there exists a set of factors* $\{\delta_{i\to j}[\boldsymbol{S}_{i,j}] : (i\text{–}j) \in \mathcal{E}_{\mathcal{T}}\}$ *such that*

$$\delta_{i\to j} \propto \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \left(\prod_{k\in \mathrm{Nb}_i - \{j\}} \delta_{k\to i} \right) \tag{11.10}$$

and moreover, we have that

$$\begin{aligned} \beta_i &\propto \psi_i \left(\prod_{j\in \mathrm{Nb}_i} \delta_{j\to i} \right) \\ \mu_{i,j} &= \delta_{j\to i} \cdot \delta_{i\to j}. \end{aligned}$$

From Fixed-Point Equations to Algorithms

Intuitively, a change in the belief set that reduces the difference between the LHS and RHS of the above equations, will get us closer to the maximum.

Idea: **use equations as iterative update rules**.

A particular order of applying these equations results exactly in the original calibration algorithm.

However, when we will consider variants of the above optimization problem, the fixed-point equations will result in *new* algorithms.

Approximation Methods Based on Optimization

The methods covered in this chapter can be divided into three categories:

1) Using the above for cluster graphs which aren't clique trees.
2) Using “approximate messages”.
3) Mean field methods: using the exact E.F and exact messages, but projecting P onto a much simpler distribution class.

Each method has it’s own fixed-point equations characterizing the optimal object in it’s context, and these result in different algorithms.

We will now broadly cover the first method.

Loopy Belief Propagation

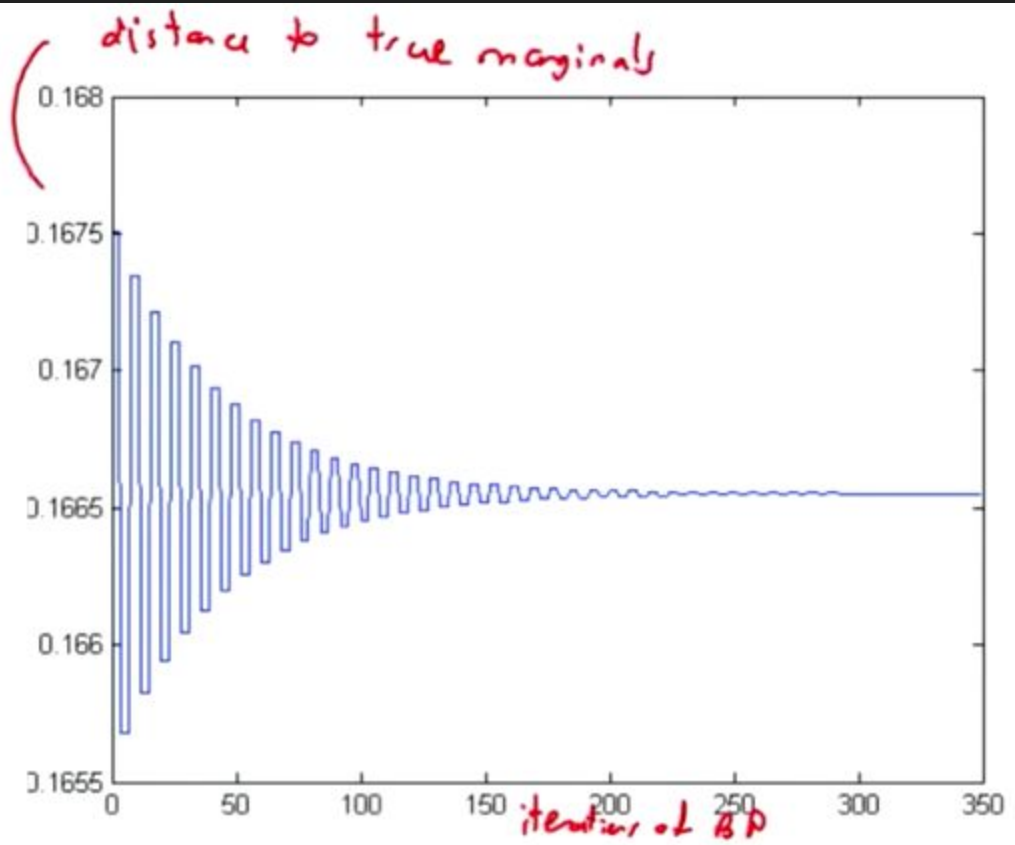
Consider using the above idea for general cluster graphs.

Nothing in the iterative algorithm relies on cliques to be “ready” etc.

So no deadlocks, yet..

Convergence? Correctness? these did relied on input being a clique tree...

Let’s observe an example case.

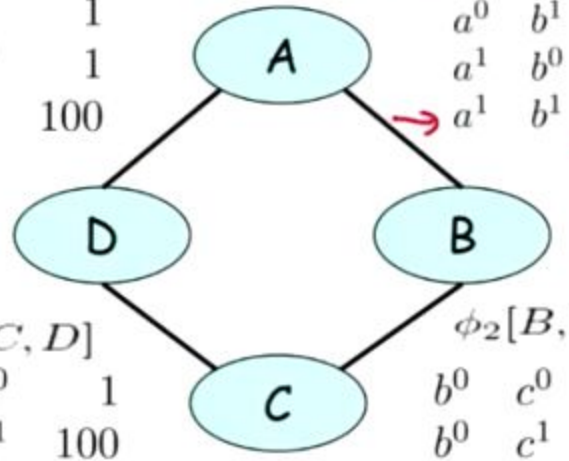


$\phi_4[D, A]$

d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

$\phi_1[A, B]$

a^0	b^0	100
a^0	b^1	2
a^1	b^0	1
a^1	b^1	100



$\phi_3[C, D]$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2[B, C]$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

Daphne Koller

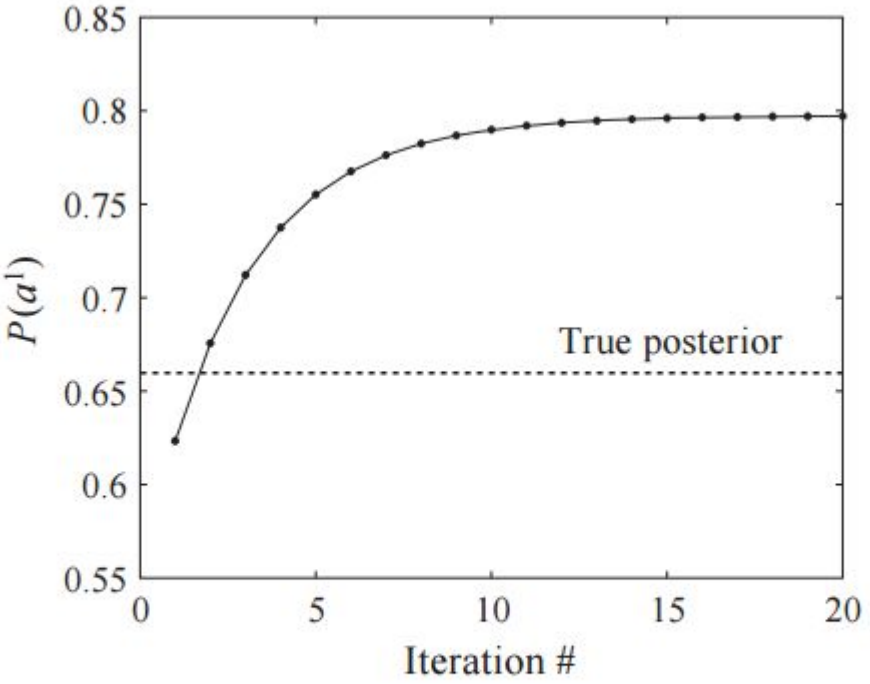


Figure 11.2 An example run of loopy belief propagation in the simple network of figure 11.1a. In this run, all potentials prefer consensus assignments over nonconsensus ones. In each iteration, we perform message passing for all the edges in the cluster graph of figure 11.1b.

Loopy Belief Propagation

Unlike in clique-trees, the process may not converge in two-passes.

It is far from clear that the propagation of beliefs necessarily converges at all.

Also, even if the process converged, there is the question of the relationship between the resulting beliefs and the original probability distribution.

Local Consistency Polytope

Notice that since the beliefs are not over a clique-tree, the measure induced by them is even not necessarily a distribution.

Beliefs for a calibrated cluster graph might not represent marginals of any single coherent distribution.

We then say the beliefs are “Pseudo-Marginals”, and denote the set we are optimizing over as the “Local Consistency Polytope” of the cluster graph:

$$Local[\mathcal{U}] = \left\{ \begin{array}{c|lcll} \{\beta_i : i \in \mathcal{V}_\mathcal{U}\} \cup & \mu_{i,j}[\boldsymbol{s}_{i,j}] & = & \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{c}_i) & \forall (i\text{-}j) \in \mathcal{E}_\mathcal{U}, \forall \boldsymbol{s}_{i,j} \in Val(\boldsymbol{S}_{i,j}) \\ \{\mu_{i,j} : (i\text{-}j) \in \mathcal{E}_\mathcal{U}\} & 1 & = & \sum_{\boldsymbol{c}_i} \beta_i(\boldsymbol{c}_i) & \forall i \in \mathcal{V}_\mathcal{U} \\ & \beta_i(\boldsymbol{c}_i) & \geq & 0 & \forall i \in \mathcal{V}_\mathcal{U}, \boldsymbol{c}_i \in Val(\boldsymbol{C}_i). \end{array} \right\} \quad (11.16)$$

Loopy Belief Propagation

Overall, we use here several approximations:

1) We use the Factored Energy Functional instead of the original Energy Functional. They are no longer equal and thus we have an approximated target function.

   Why is it impossible to resort to our original distribution distance measure?

2) We approximate the original distribution with an element from the local consistency polytope.
3) Different cluster graphs will result in different polytopes and thus generally in different approximations.

Analyzing Convergence

Key question: whether and when propagation converges.

Indeed, there are many networks for which it doesn’t converge.

Convergnce → Calibration (messages don’t change).

Analysis of convergence is complex.

In the simple case of a pairwise markov network, KF[09] offers a proof of convergence by viewing synchronous updates as alpha-contractions of message space, and applying the Banach fixed-point theorem.

Practical Heuristics to Achieve Convergence

1) Intelligent message scheduling:

Don’t use synchronous updates (since these are less informed updates).

Tree reparameterization: calibrate sub-tree after sub-tree. spanning is better.

Residual belief propagation: pass critical messages first.

Critical messages = where adjacent clusters disagree the most (need to use a priority queue).

Practical Heuristics to Achieve Convergence

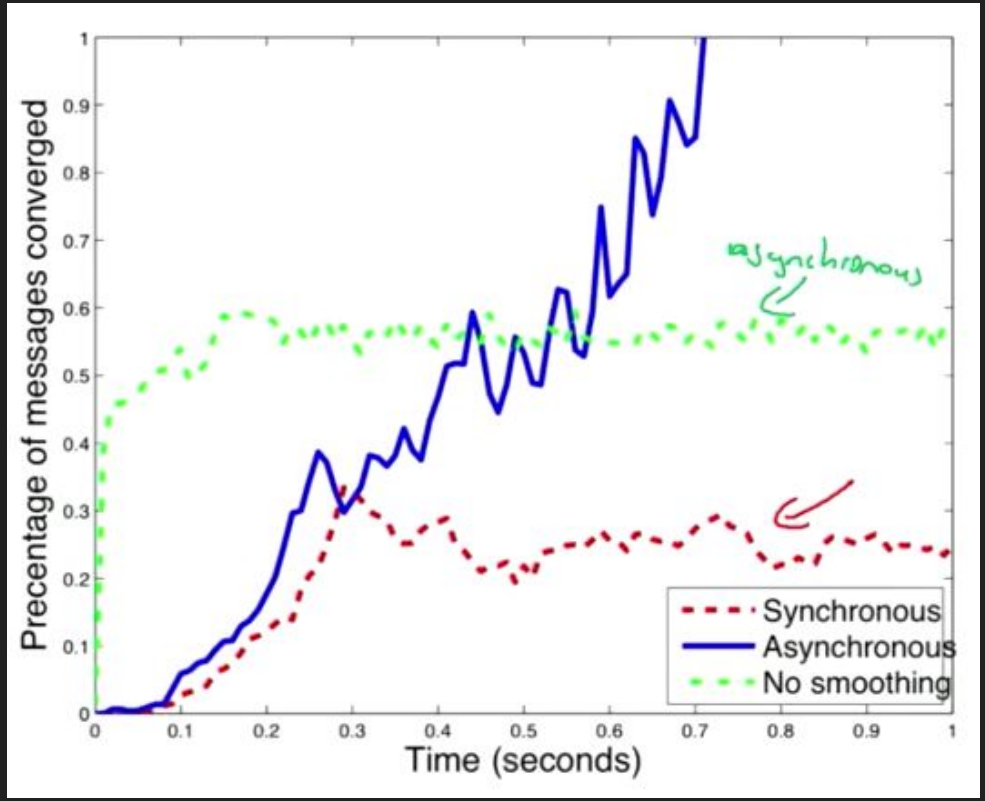
2) Message Damping:

Use weighted average updates.

$$\delta_{i\to j} \leftarrow \lambda \left(\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \prod_{k\neq j} \delta_{k\to i} \right) + (1-\lambda)\delta_{i\to j}^{\text{old}}$$

Just a heuristic? Fixed point solutions of original equations are also solutions of damped equations.

Method Comparison for Ising Model



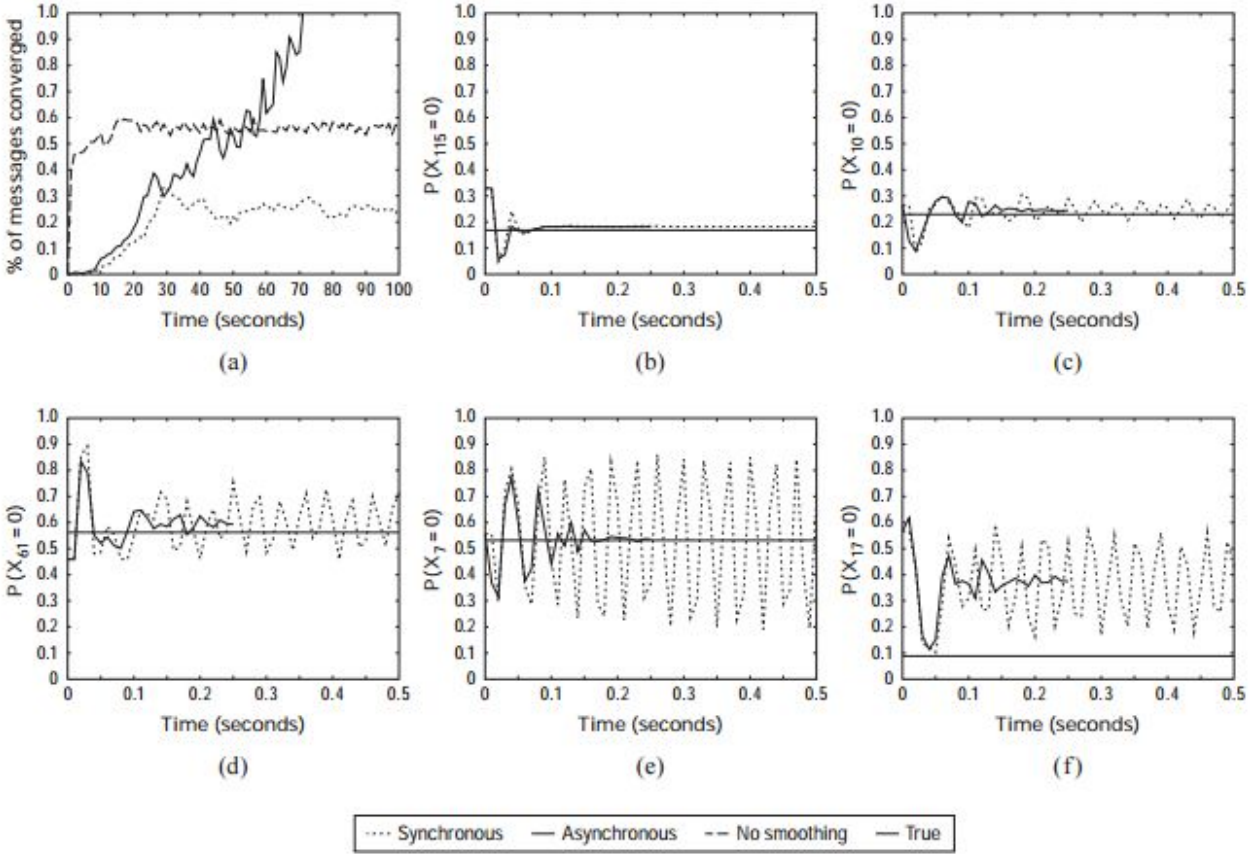


Figure 11.C.1 — Example of behavior of BP in practice on an 11 × 11 Ising grid. (a) Percentage of messages converged as a function of time for three different BP variants. (b) A marginal where both variants converge rapidly. (c–e) Marginals where the synchronous BP marginals oscillate around the asynchronous BP marginals. (f) A marginal where both variants are inaccurate.

Achieving Convergence

These qualitative differences between the BP variants are quite consistent across many random and real-life models.

Typically, the more complex the inference problem, the larger the gaps in performance.

For very complex real-life networks involving tens of thousands of variables and multiple cycles, even asynchronous BP is not very useful and more elaborate propagation methods or convergent alternatives must be adopted.

Correctness at Convergence?

Theorem (unproven):

All *stable* convergence points Q of the above algorithm are maxima of target function.

Other direction not generally true: might have *unstable maxima*, but this is not so common.

Correctness at Convergence: Tree Consistency

Consider a subtree of the cluster graph, which holds the running intersection property (and thus is a clique tree for a subset of our original rvs).

At convergence, beliefs are calibrated, and thus represent the marginals of the distribution induced by the beliefs of the tree.

Why is this not enough for exact inference?

Residual beliefs might change everything.

Correctness at Convergence: Tree Consistency

By analyzing the residual term (product of sepset beliefs divided by clique beliefs, for sepsets and cliques outside of the tree), we can get an *approximation bound*.

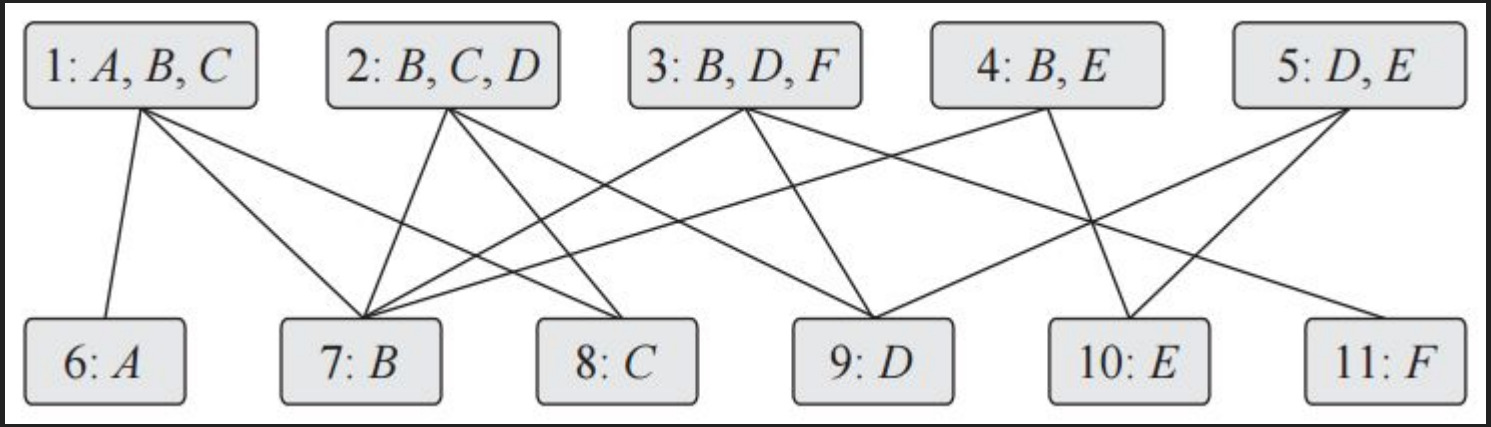
The possibility to conduct such an analysis varies from cluster graph to cluster graph.

More methods of correctness analysis appear later in chapter.

The Bethe Approximation

Remember that choosing the specific cluster graph also made a difference.

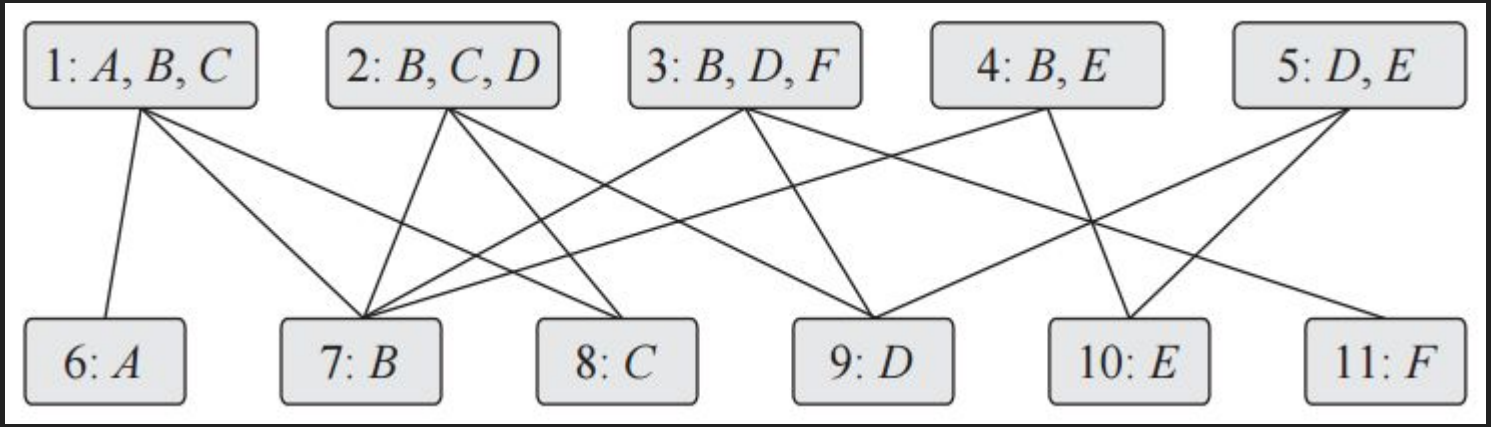
The Bethe graphs are common default constructions.



The Bethe Approximation

We have a cluster for each rv, a cluster for each original factor, and edges with singleton sepsets between factors and rv's within their scope.

Tradeoff: message costs vs structure preserving



Application: Turbocodes

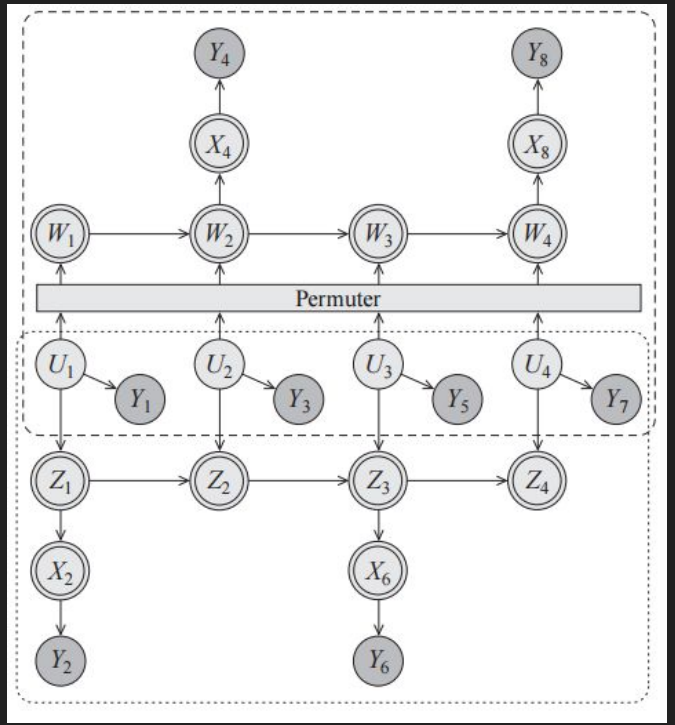
message-->code-->noisy code-->decoded message

The task of decoding a message is a probabilistic inference task.

The Shannon limit: error-rate per noise-level

Shannon gave *non-constructive* proof for existence of a code which reaches his theoretical limit.

Until Turbocodes, no proposed mechanism even came close to this limit.



Application: Turbocodes

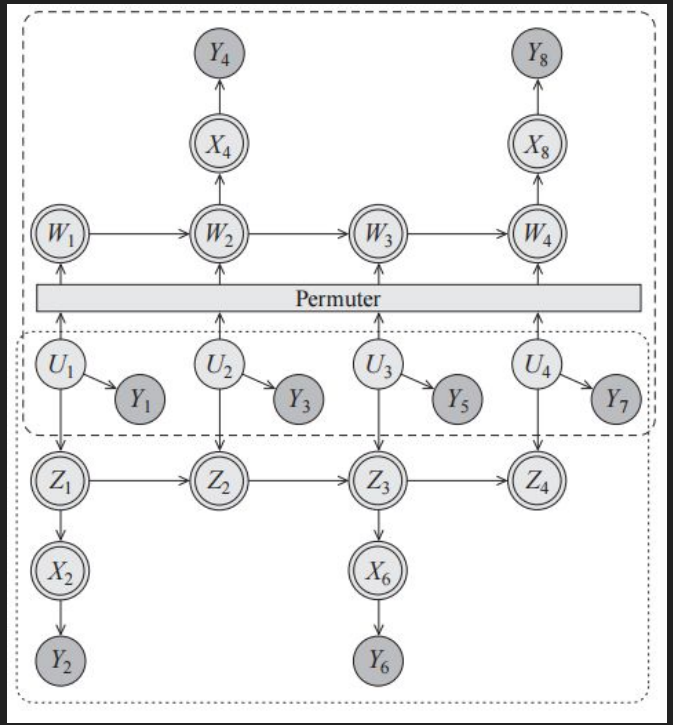
Coding scheme transmits two sets of bits:

<original bits, transformed bits>

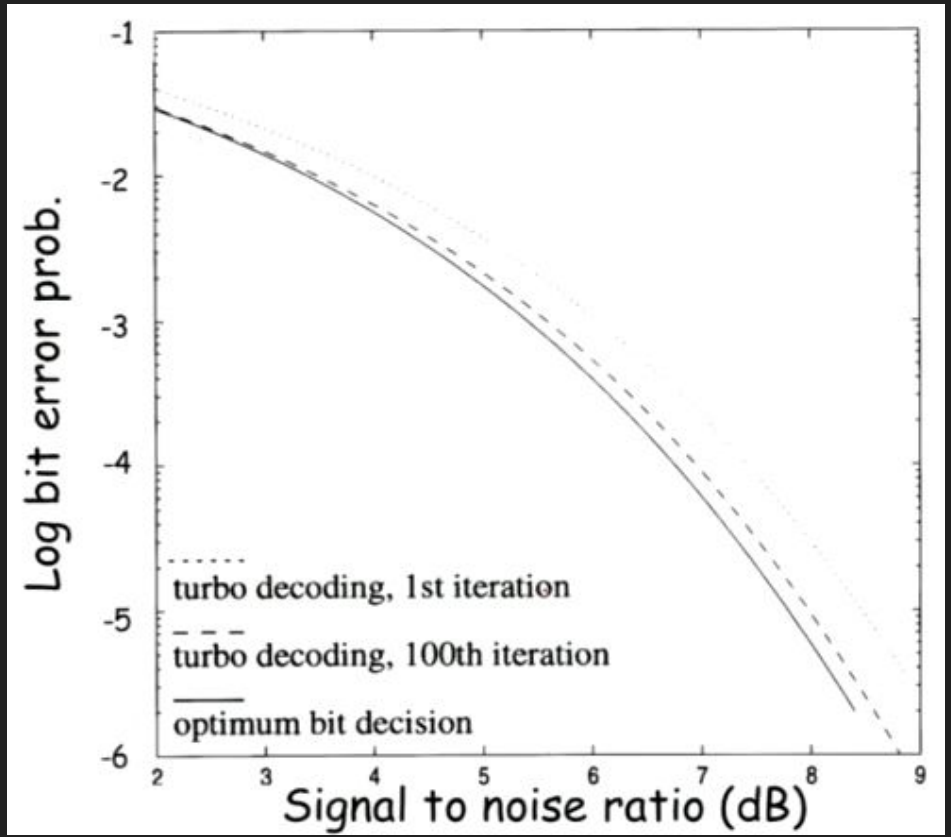
Each can be decoded (with some error rate).

Idea: use posterior of one set as prior of other, and vice versa, in a feedback loop.

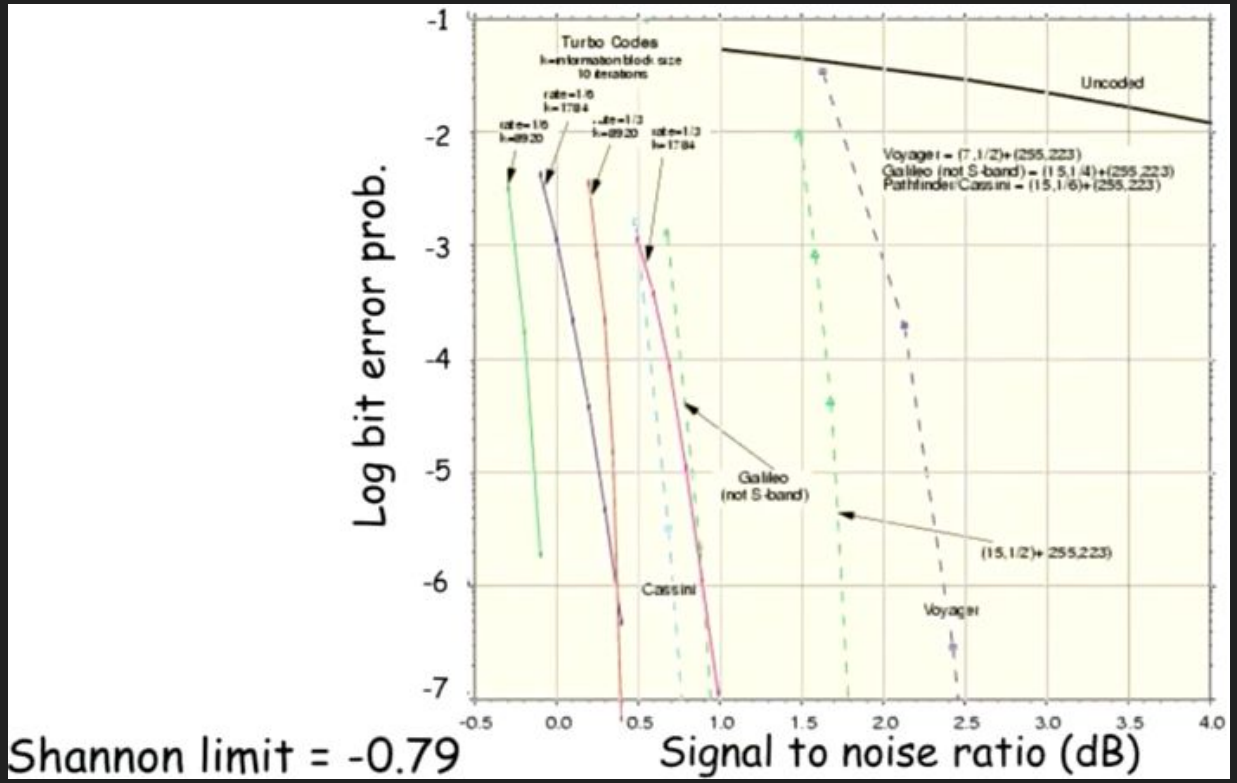
weak coding schemes “turbocharging” each other.



Application: Turbocodes



Application: Turbocodes



Application: Turbocodes

Analysis showed this decoding scheme is equivalent to a loopy belief propagation (with a specific message scheduling scheme) for the bayesian network representing the model.

Today based on these insights there are even better coding schemes named “Low-density parity checking codes”.

Used every day by cell phones, communication satellites, NASA, etc.

Thank you very much (: