
Loopy Belief Propagation

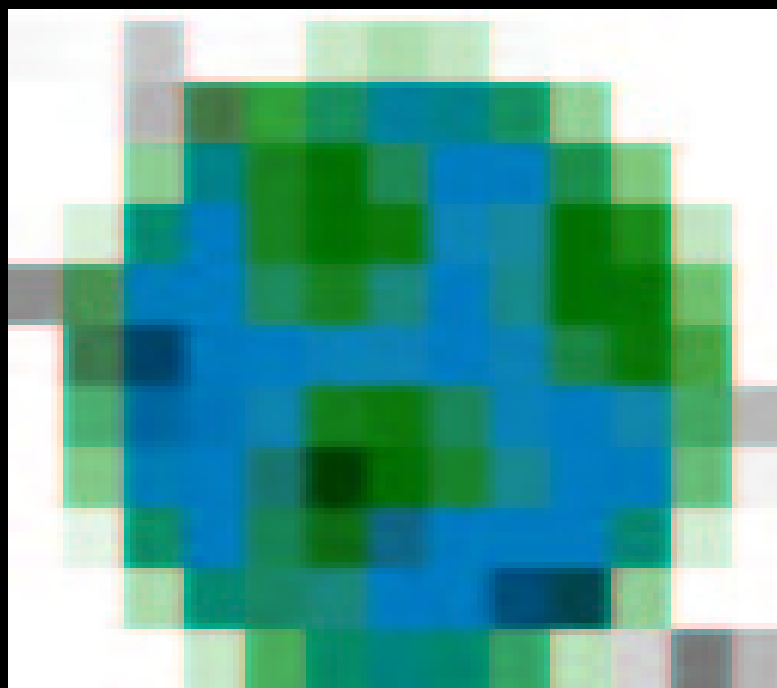
Research Exam

Kristin Branson
September 29, 2003

Problem Formalization

Reasoning about any real-world problem requires assumptions about the structure of the problem: the **relevant variables** and the **interrelationships** of these variables.

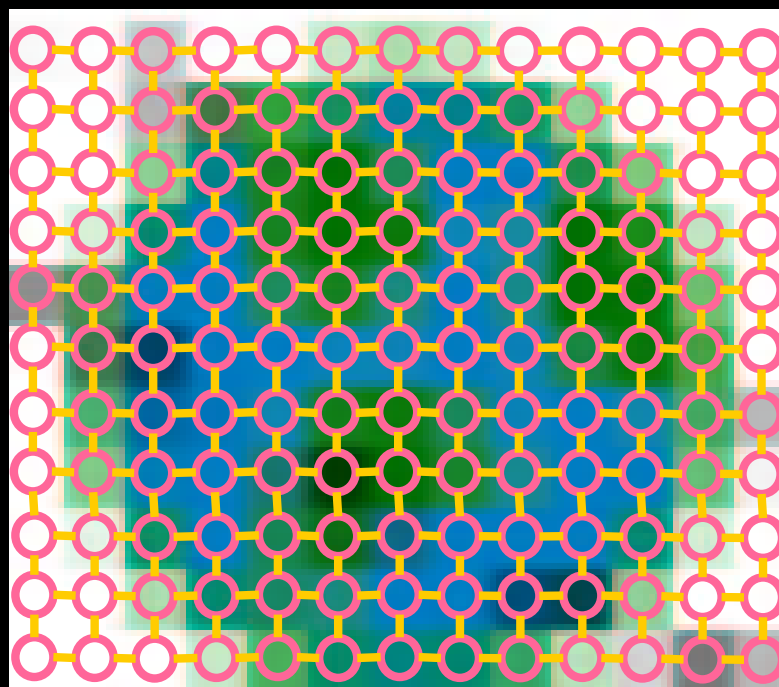
A **graphical model** is a formal representation of these assumptions.



Problem Formalization

Reasoning about any real-world problem requires assumptions about the structure of the problem: the **relevant variables** and the **interrelationships** of these variables.

A **graphical model** is a formal representation of these assumptions.



Probabilistic Model

These assumptions are a simplification of the problem's true structure.

The world appears **stochastic** in terms of the model.

Graphical models are interpreted as describing the **probability distribution** of random variables.

Probabilistic Inference

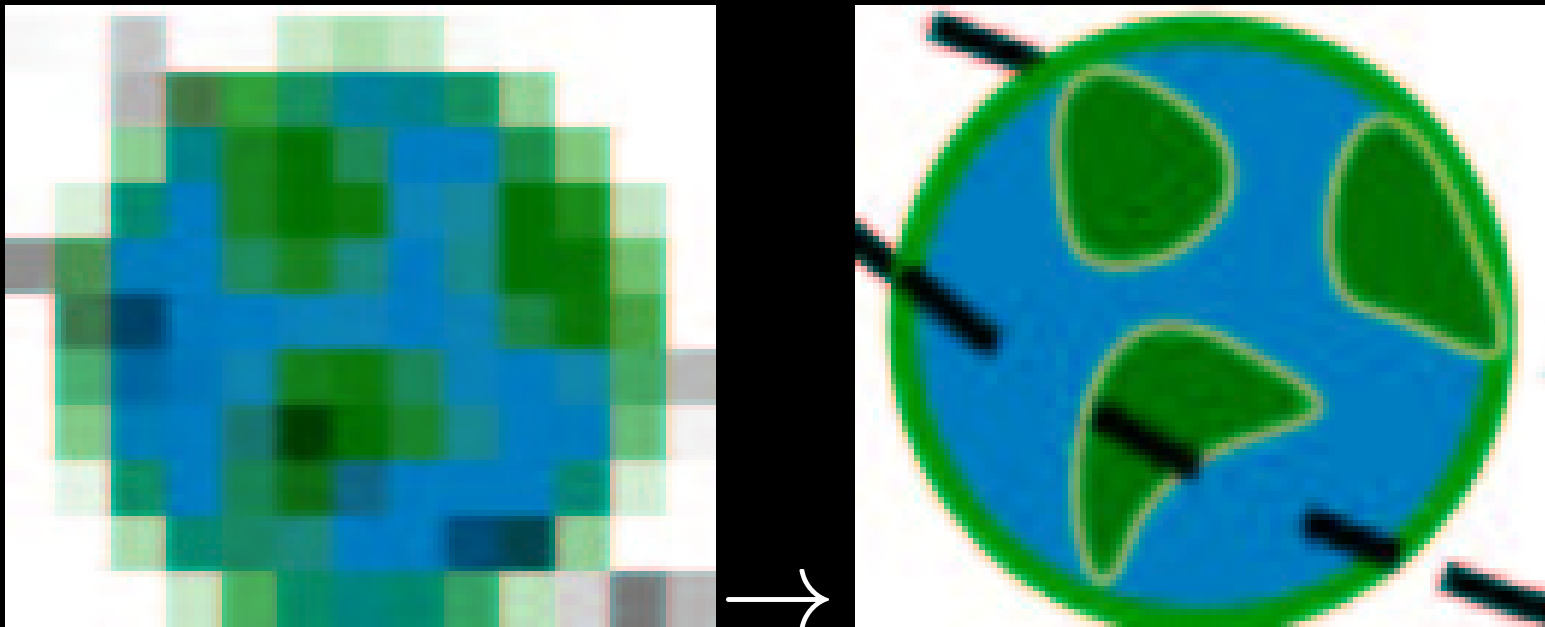
Reasoning about real-world problems can be modeled as **probabilistic inference** on a distribution described by a graph.

Probabilistic inference involves computing desired properties of a distribution:

- What is the most probable state of the variables, given some evidence?
- What is the marginal distribution of a subset of the variables, given some evidence?

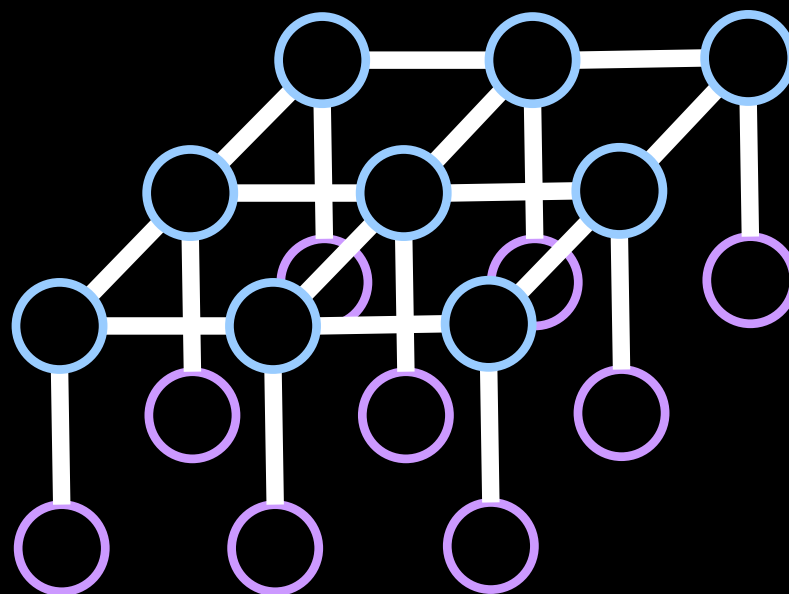
Inference Example

Estimate the intensity value of each pixel of an image given a corrupted version of the image.



Inference Example

Estimate the intensity value of each pixel of an image given a corrupted version of the image.



Assume $y_i \sim \mathcal{N}(x_i, \Sigma)$.

Assume relationship between uncorrupted pixel variables can be described by a local smoothness constraint.

Inference is Intractible

Assuming a sparse graphical model greatly simplifies the problem.

Still, probabilistic inference is in general **intractible**.

Exact inference algorithms are exponential in the graph size.

Pearl's Belief Propagation (BP) algorithm performs **approximate inference** on an arbitrary graphical model.

Loopy BP

The assumptions made by BP only hold for acyclic graphs.

For graphs containing cycles, **loopy BP** is not guaranteed to converge or be correct.

However, it has been applied with experimental success.

Experimental Results

- Impressive experimental results were first observed in graphical code schemes.
- The **Turbo code** error-correcting code scheme was described as “the most exciting and potentially important development in coding theory in many years” (McEliece et al., 1995).
- Murphy et al. experimented with loopy BP on graphical models that appear in **machine learning**.
- They concluded that loopy BP did converge to **good approximations** in many cases.
- Since then, loopy BP has successfully been applied to many applications in machine learning and computer vision.

Theoretical Analysis

When considering applying loopy BP, one would like to know whether it will converge to good approximations.

In this exam, I present recent analyses of loopy BP.

Outline

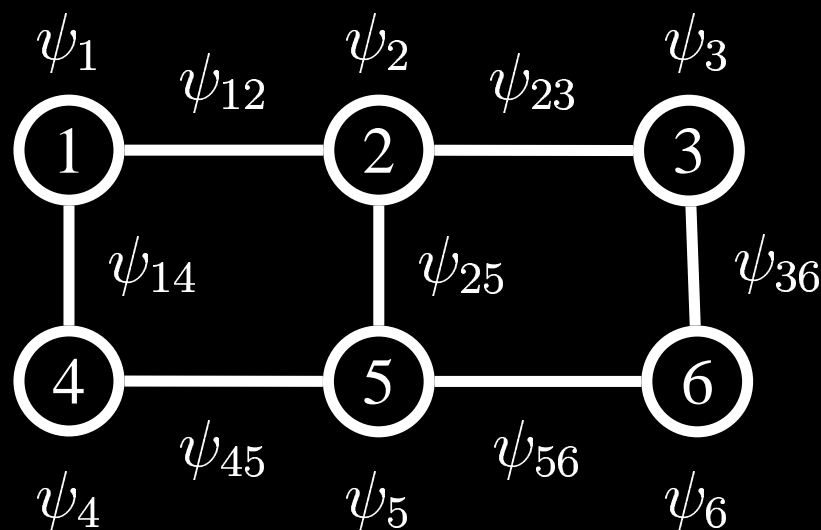
- Background.
 - Undirected graphical models.
 - Belief Propagation algorithm.
- Three techniques for analyzing loopy BP.
 - Algebraic analysis.
 - Unwrapped tree.
 - Reparameterization.
- Future work.

Outline

- Background.
 - Undirected graphical models.
 - Belief Propagation algorithm.
- Three techniques for analyzing loopy BP.
 - Algebraic analysis.
 - Unwrapped tree.
 - Reparameterization.
- Future work.

Markov Random Fields

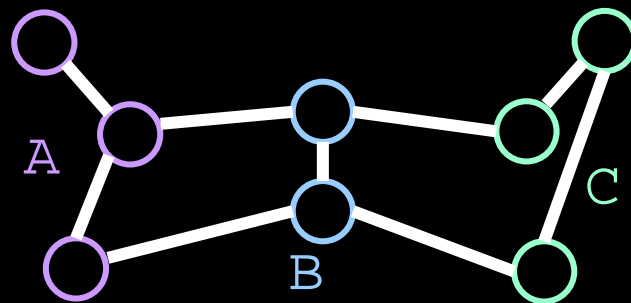
An undirected graphical model represents a distribution by an undirected graph.



- Nodes represent random variables.
- Edges represent dependencies.
- Each clique is associated with a potential function, $\psi_C(\mathbf{x}_C)$.

Markov Properties

Paths in the graph represent dependencies.
If two nodes are not connected, the variables are independent.



If nodes B separate nodes A from nodes C , then A and C are conditionally independent given B .

The **Hammersley-Clifford theorem**: the conditional independence assumptions hold if and only if the distribution factorizes as the product of potential functions over cliques: $p(\mathbf{x}) = \alpha \prod_C \psi_C(\mathbf{x}_C)$.

Pairwise MRFs

To simplify notation, we focus on pairwise MRFs. The largest clique size in a pairwise MRF is two. The distribution can therefore be represented as

$$p(\mathbf{x}) = \alpha \prod_{u \in V} \psi_u(x_u) \prod_{(u,v) \in E} \psi_{uv}(x_u, x_v).$$

A MRF with larger cliques can be converted into a pairwise MRF.

Probabilistic Inference

We discuss two inference problems:

- **Marginalization**: For each node u , compute

$$p_u(x_u | \mathbf{y}^*) = \sum_{\{\mathbf{x}' | x'_u = x_u\}} p(\mathbf{x}' | \mathbf{y}^*).$$

- **MAP assignment**: Find the maximum probability assignment given the evidence:

$$\mathbf{x}^{MAP} = \operatorname{argmax}_{\{\mathbf{x}'\}} p(\mathbf{x}' | \mathbf{y}^*).$$

Max-Marginals

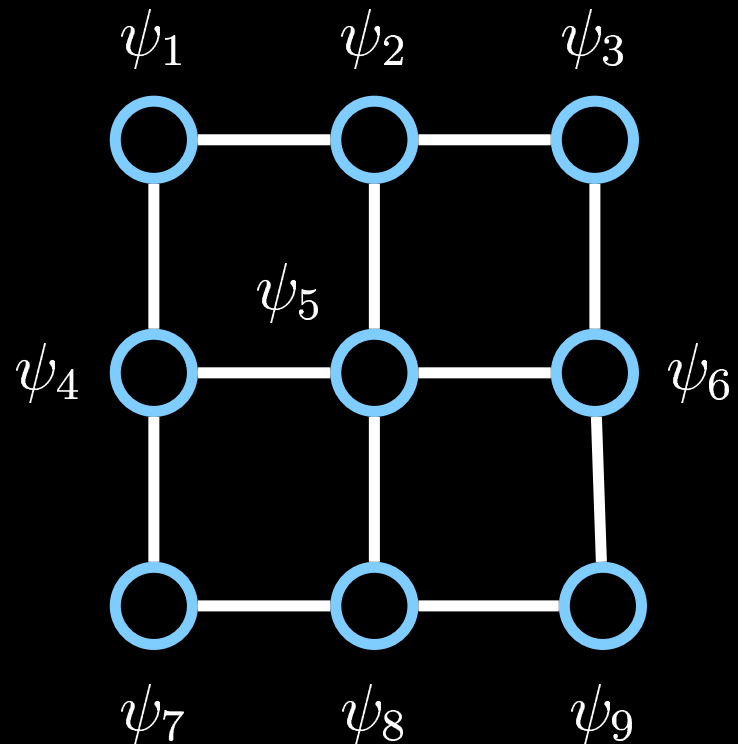
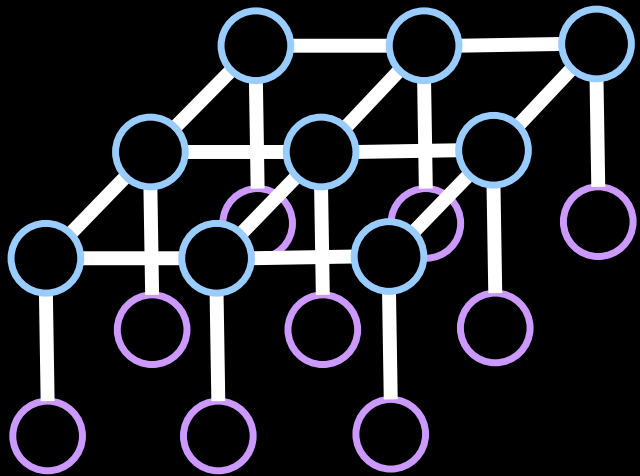
To find the MAP assignment, we will compute the **max-marginals**:

$$P_u(x_u | \mathbf{y}^*) = \max_{\{\mathbf{x}' | x'_u = x_u\}} p(\mathbf{x}' | \mathbf{y}^*).$$

The MAP assignment x_u^{MAP} maximizes $P_u(\cdot)$.

Notation

To simplify notation, we assume that effect of the observed data \mathbf{y}^* is encapsulated in the single-node potentials ψ_u .



Variable Elimination

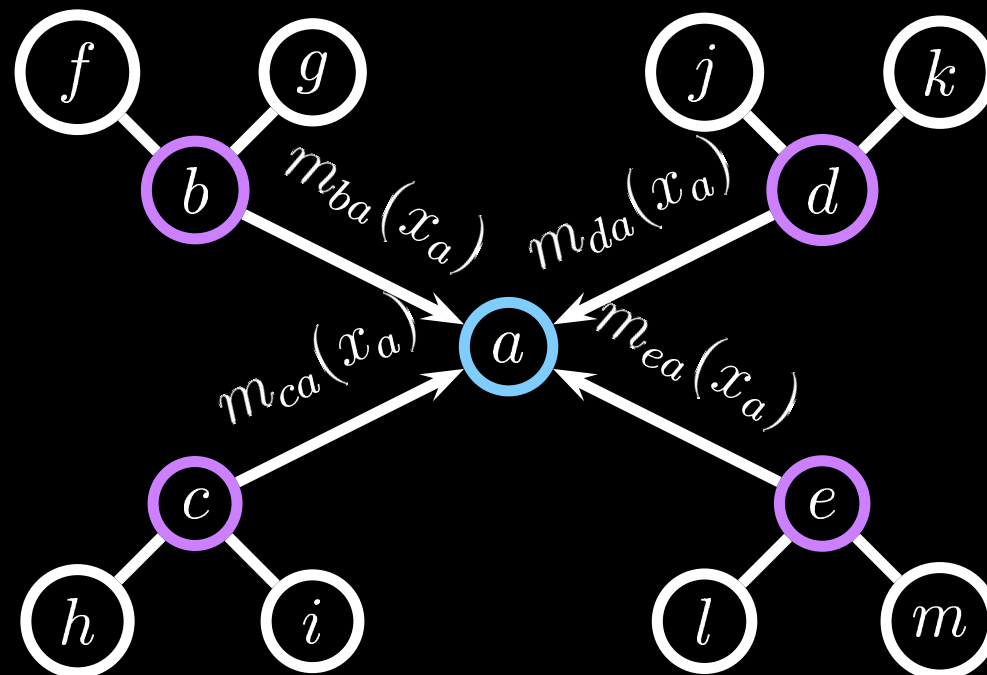
Exact inference can be performed by repeatedly eliminating variables:

$$\begin{aligned} p_u(x_u) &= \alpha \sum_{\{\mathbf{x}' | x'_u = x_u\}} p(\mathbf{x}') \\ &= \alpha \sum_{\{\mathbf{x}' | x'_u = x_u\}} f_1(\mathbf{x}_{V/v}) \sum_{x'_v} f_2(\mathbf{x}_{\{v, S(v)\}}) \\ &= \alpha \sum_{\{\mathbf{x}' | x'_u = x_u\}} f_1(\mathbf{x}_{V/v}) m(\mathbf{x}_{S(v)}) \end{aligned}$$

Outline

- Background.
 - Undirected graphical models.
 - **Belief Propagation algorithm.**
- Three techniques for analyzing loopy BP.
 - Algebraic analysis.
 - Unwrapped tree.
 - Reparameterization.
- Future work.

BP for Trees

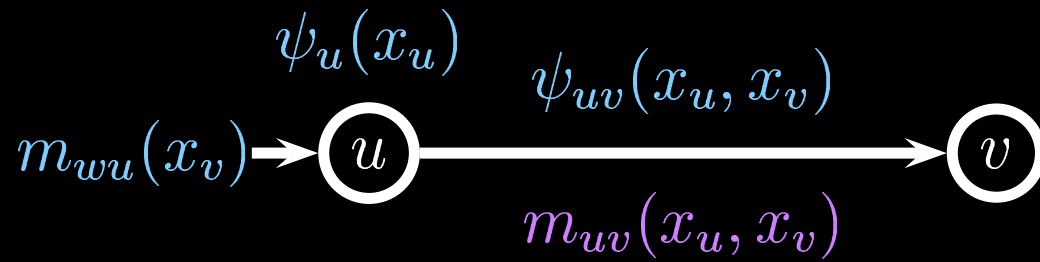


BP breaks the [max-]marginalization for a node a into **independent subproblems** corresponding to the subtrees rooted at the neighbors of a .

In each subproblem, BP **eliminates** all the variables except a .

The result is a **message** $m_{ua}(x_a)$.

BP for Trees



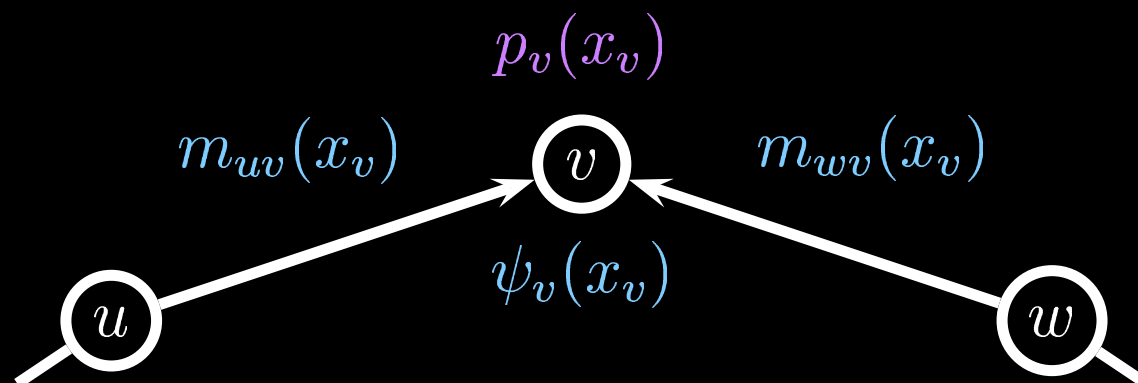
BP is a dynamic programming form of variable elimination.

The creation of a message is equivalent to repeatedly removing leaf nodes of the subtree:

$$m_{uv}(x_v) = \alpha \sum_{x_u} \psi_{uv}(x_u, x_v) \psi_u(x_u) \prod_{w \in N(u)/v} m_{wu}(x_u).$$

$$m_{uv}(x_v) = \alpha \max_{x_u} \psi_{uv}(x_u, x_v) \psi_u(x_u) \prod_{w \in N(u)/v} m_{wu}(x_u).$$

BP for Trees

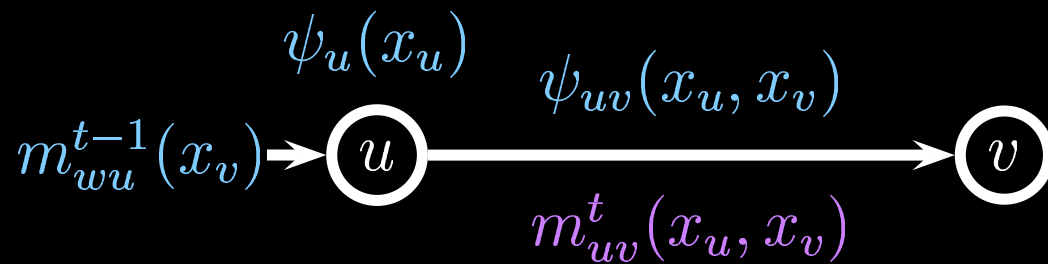


In terms of these messages, the [max-]marginals are

$$p_v(x_v) = \alpha \psi_v(x_v) \prod_{u \in N(v)} m_{uv}(x_v).$$

$$P_v(x_v) = \alpha \psi_v(x_v) \prod_{u \in N(v)} m_{uv}(x_v).$$

Parallel Message Passing



Instead of waiting for smaller problems to be solved before solving larger problems, we can iteratively pass messages in **parallel**.

- Initialize the messages $m_{uv}^0(x_v)$ for all $(u, v) \in E$.
- For $t = 1, 2, \dots$ until convergence,
 - Update $m_{uv}^t(x_v)$ using $m_{vu}^{t-1}(x_u)$ for all $(u, v) \in E$.

Loopy BP

The parallel BP algorithm can be applied to arbitrary graphs.

However, the assumptions made by BP **do not hold** for a loopy graph.

Loopy BP is not guaranteed to converge.

If it does converge, it is not guaranteed to converge to the correct [max-]marginals.

We will call the approximate [max-]marginals **beliefs** $b_u(x_u)$.

Theoretical Analysis

- When will loopy BP converge?
- How good an approximation are the [max-]marginals and max-product assignment?

I present three techniques for analyzing BP. The first two analyze the message-passing dynamics, while the third analyzes the steady-state beliefs directly.

Outline

- Background.
 - Undirected graphical models.
 - Belief Propagation algorithm.
- Three techniques for analyzing loopy BP.
 - Algebraic analysis.
 - Unwrapped tree.
 - Reparameterization.
- Future work.

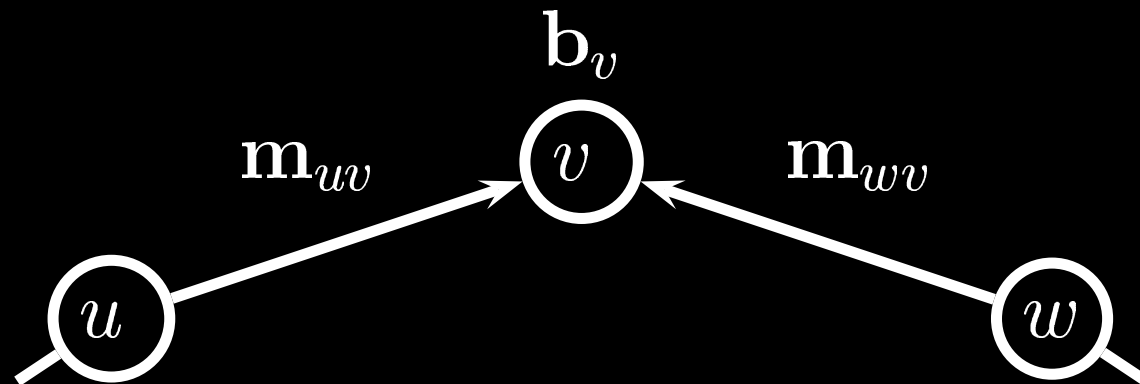
Algebraic Analysis Overview

- We first discuss an algebraic analysis of the **sum-product** algorithm for a single-cycle graph.
- We represent each message update of the sum-product algorithm as a **matrix multiplication**.
- We use linear algebra results to show the relationship between the steady-state beliefs and the true marginals, as well as convergence properties.
- The sum-product algorithm converges for a single-cycle graph.
- The convergence rate and the accuracy of the beliefs are related.

Matrix Representation

We represent the message and belief functions as vectors \mathbf{m}_{uv} and \mathbf{b}_u .

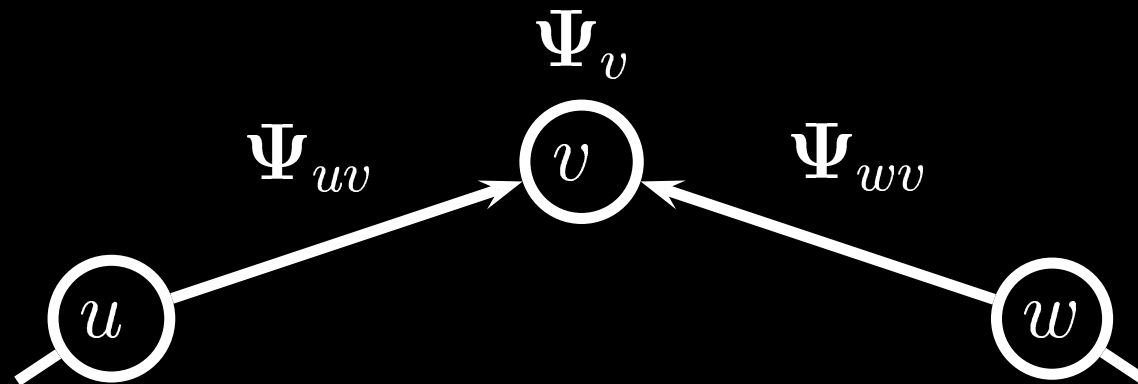
Similarly, we represent the single- and pair-node potentials as matrices Ψ_u and Ψ_{uv} .



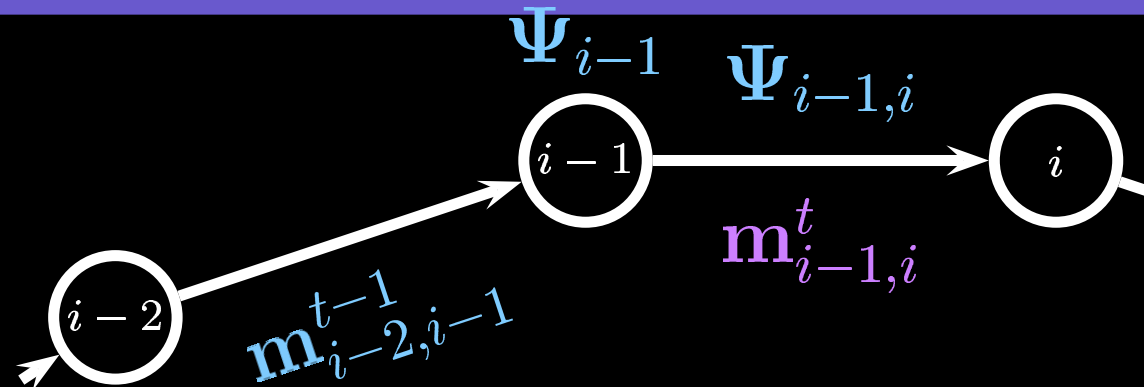
Matrix Representation

We represent the message and belief functions as vectors \mathbf{m}_{uv} and \mathbf{b}_u .

Similarly, we represent the single- and pair-node potentials as matrices Ψ_u and Ψ_{uv} .



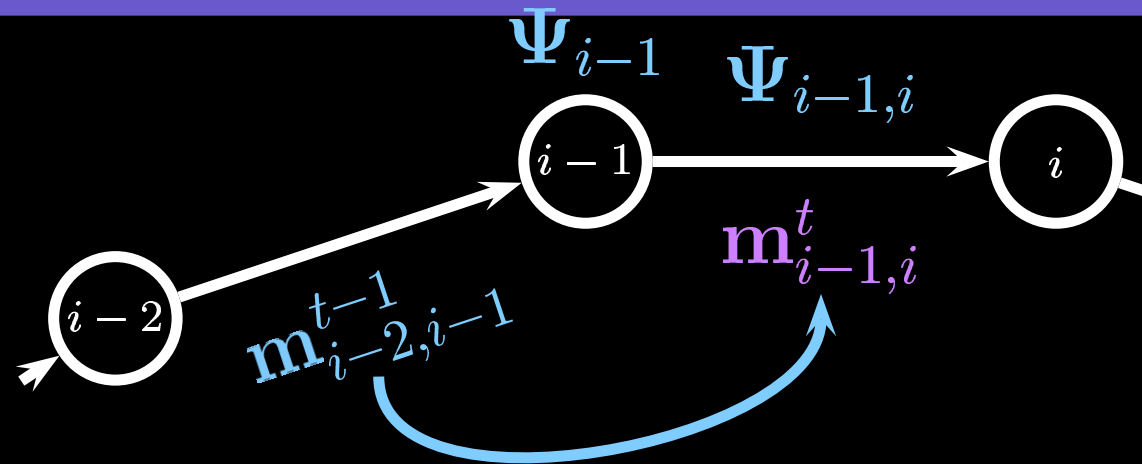
Matrix Message Updates



For a graph consisting of a single cycle, a message update is a matrix multiplication

$$m_{i-1,i}^t = \alpha \Psi_{i-1,i} \Psi_{i-1} m_{i-2,i-1}^{t-1}$$

Matrix Message Updates

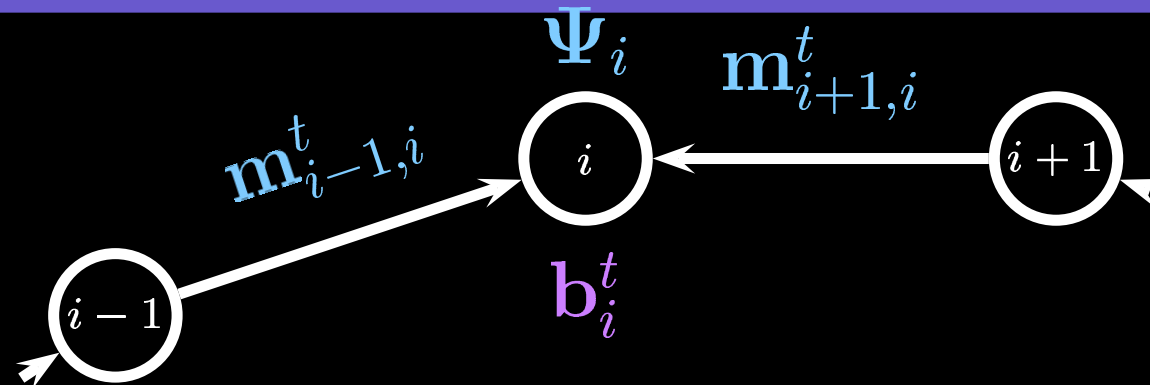


$$\Psi_{i-1,i} \Psi_{i-1}$$

For a graph consisting of a single cycle, a message update is a matrix multiplication

$$m_{i-1,i}^t = \alpha \Psi_{i-1,i} \Psi_{i-1} m_{i-2,i-1}^{t-1}$$

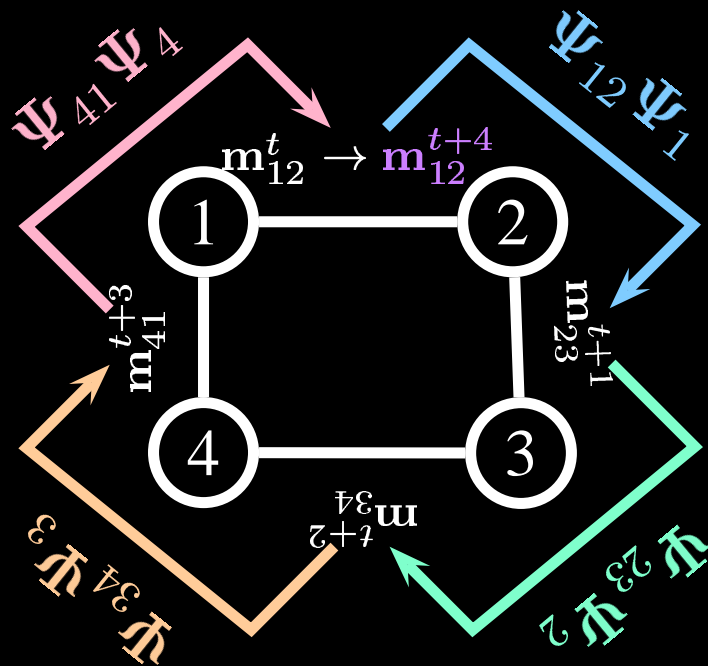
Matrix Belief Updates



For a graph consisting of a single cycle, a belief update is

$$\mathbf{b}_i = \alpha \text{diag}(\mathbf{m}_{i-1,i} \Psi_i \mathbf{m}_{i+1,i}^T).$$

Matrix Message Updates



A series of message-updates is a series of matrix multiplications.

$$\begin{aligned}
 \mathbf{m}_{i-1,i}^{t+l} &= \alpha(\Psi_{i-1,i} \Psi_{i-1})(\Psi_{i-2,i-1} \Psi_{i-2}) \dots (\Psi_{i,i+1} \Psi_i) \mathbf{m}_{i-1,i}^t \\
 &= \alpha \mathbf{C}_{i-1,i} \mathbf{m}_{i-1,i}^t
 \end{aligned}$$

Power Method Lemma

- $\mathbf{m}^{t+l} = \alpha \mathbf{C} \mathbf{m}^t$ converges to the principal eigenvector of \mathbf{C} , $\alpha \mathbf{u}_1$.
- The convergence rate is the ratio of the first two eigenvalues, $r = |\lambda_2/\lambda_1|$.
- This applies if
 - The eigenvalues follow $|\lambda_1| > |\lambda_2|$ (e.g. if the distribution is positive).
 - The initial vector \mathbf{m}^0 is not orthogonal to \mathbf{u}_1 .

True Marginals

The sums and multiplications performed when computing the marginals are a distributed version of the sums and multiplications performed when computing the diagonal elements of $\mathbf{C}_{i-1,i}$:

$$\mathbf{p}_i = \alpha \text{diag}(\mathbf{C}_{i-1,i}).$$

Beliefs

- $\Psi_i \mathbf{m}_{i+1,i}^T$ is the left eigenvector of $\mathbf{C}_{i-1,i}$, since

$$\mathbf{C}_{i-1,i} = \Psi_i^{-1} \mathbf{C}_{i+1,i}^T \Psi_i.$$

- The steady-state beliefs are therefore the diagonal elements of the outer product of the right and left principal eigenvectors of $\mathbf{C}_{i-1,i}$.

Beliefs vs True Marginals

- The diagonal elements of the outer product of the right and left principal eigenvectors is an approximation of the diagonal elements of $C_{i-1,i}$.
- The goodness of the approximation depends on the ratio $\lambda_1 / \sum_i \lambda_i$.
- Recall that the convergence rate depends on a similar ratio, λ_1 / λ_2 .
- The faster the convergence, the better the approximation.

Algebraic Analysis Recap

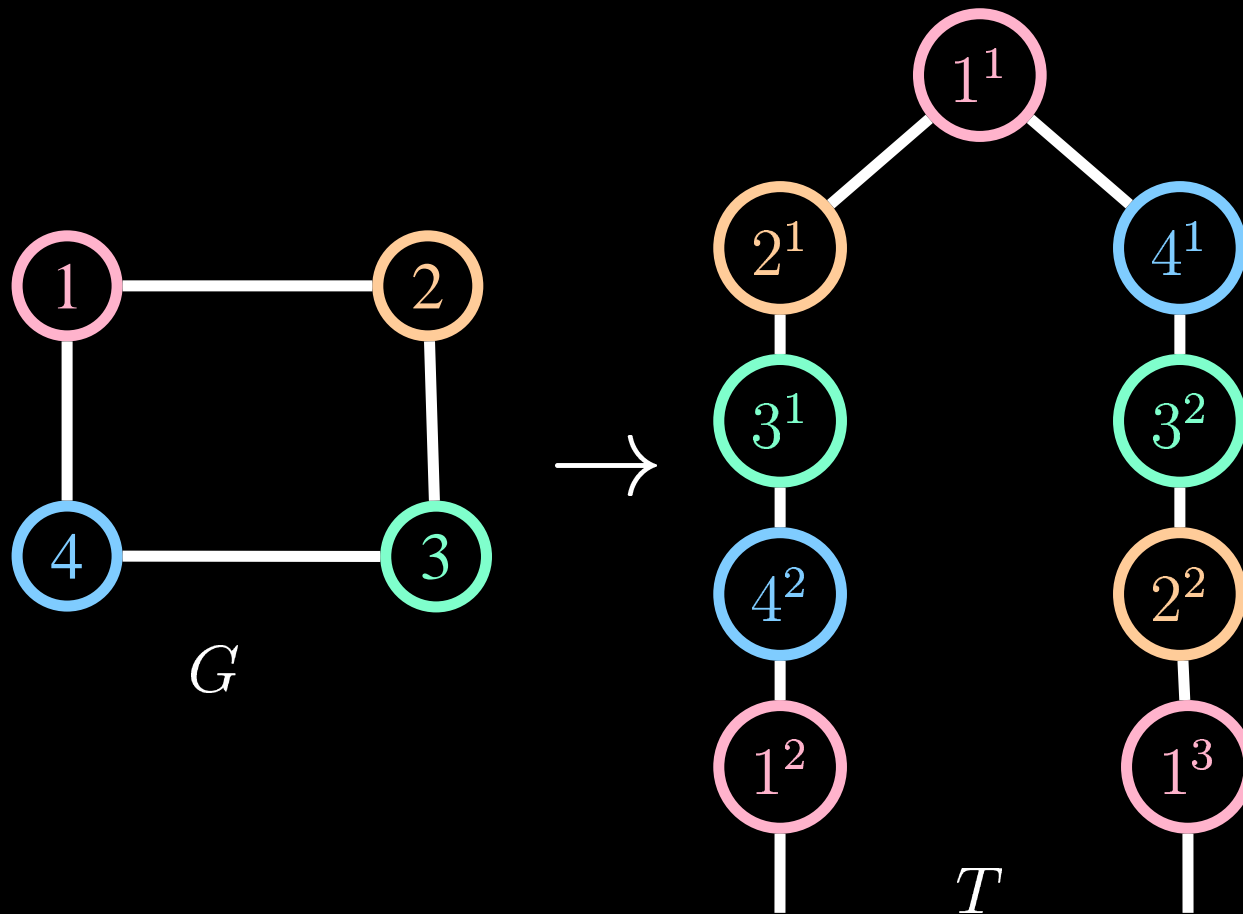
- By representing the sum-product algorithm on a single-cycle graph as a series of matrix multiplications, we showed the following results:
 - The sum-product algorithm converges for positive distributions.
 - Both the convergence rate and the accuracy of the steady-state beliefs depend on the relative size of the first eigenvalue of the same matrix C .

Outline

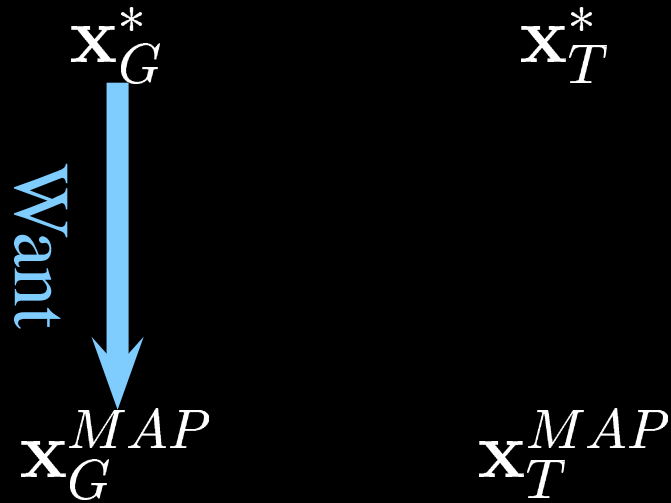
- Background.
 - Undirected graphical models.
 - Belief Propagation algorithm.
- Three techniques for analyzing loopy BP.
 - Algebraic analysis.
 - **Unwrapped tree.**
 - Reparameterization.
- Future work.

Unwrapped Tree

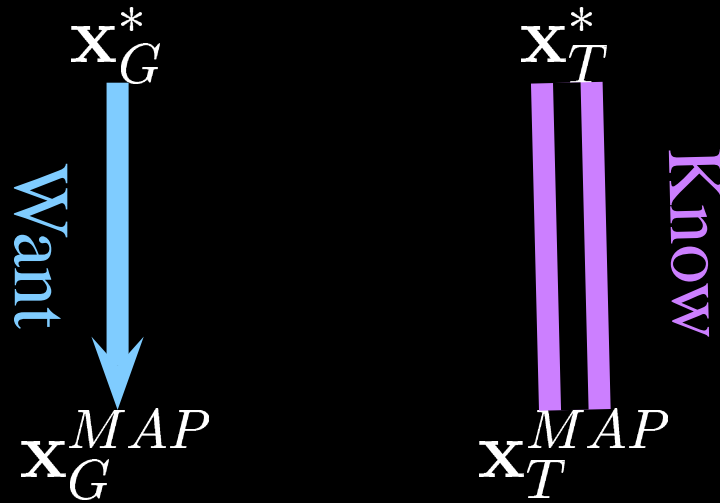
To analyze the BP algorithm, we construct the **unwrapped tree**, T , an acyclic graph that is **locally equivalent** to the original graph, G .



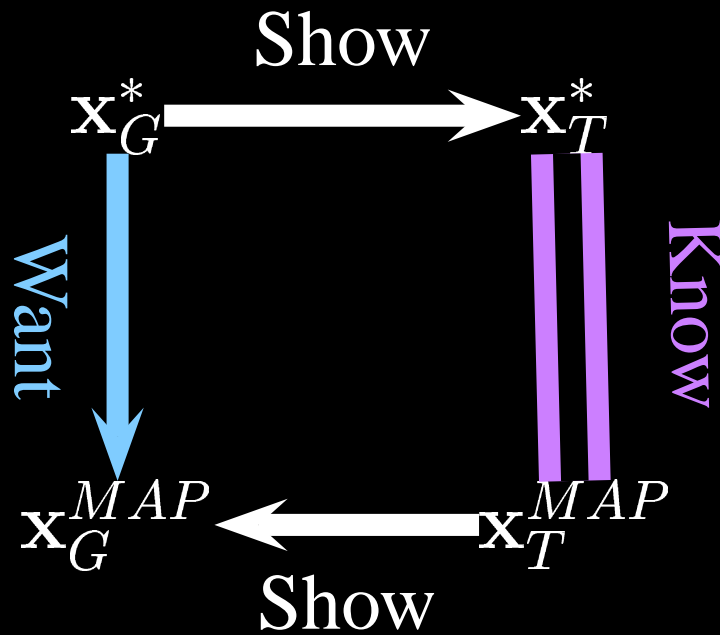
Unwrapped Tree Analysis



Unwrapped Tree Analysis



Unwrapped Tree Analysis

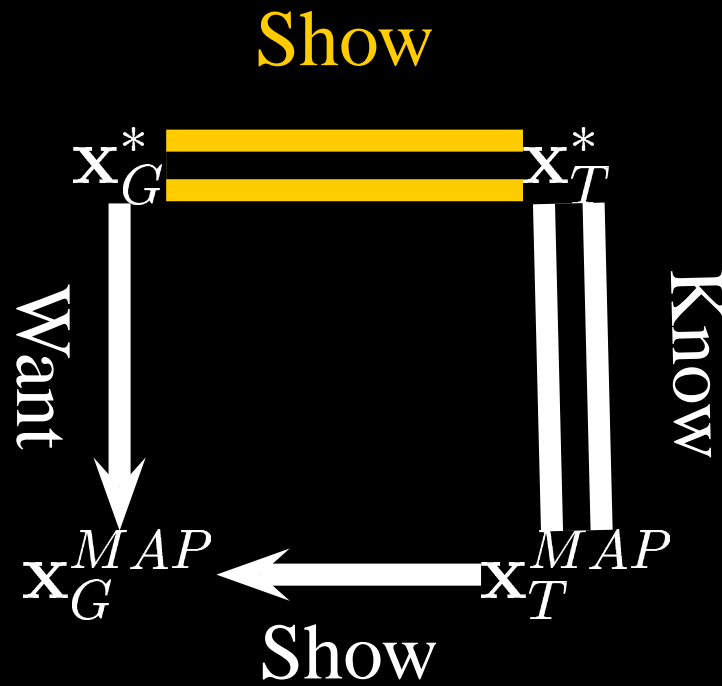


Unwrapped Tree Overview

The unwrapped tree was used to prove:

- The max-product assignment is exact for a graph containing a single cycle.
- The max-product assignment has a higher probability than any other assignment in a large neighborhood.

Unwrapped Tree Construction



Unwrapped Tree Construction

The unwrapped tree, T , is constructed from G as follows:

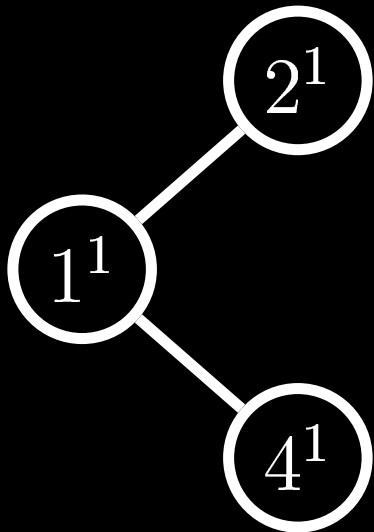
- Choose an arbitrary root node r . Initialize $T = r$.
- Repeat:
 - For each leaf u of T , find the neighbors of the corresponding node in G , other than the parent of u in T . Add these nodes to the tree.

1¹

Unwrapped Tree Construction

The unwrapped tree, T , is constructed from G as follows:

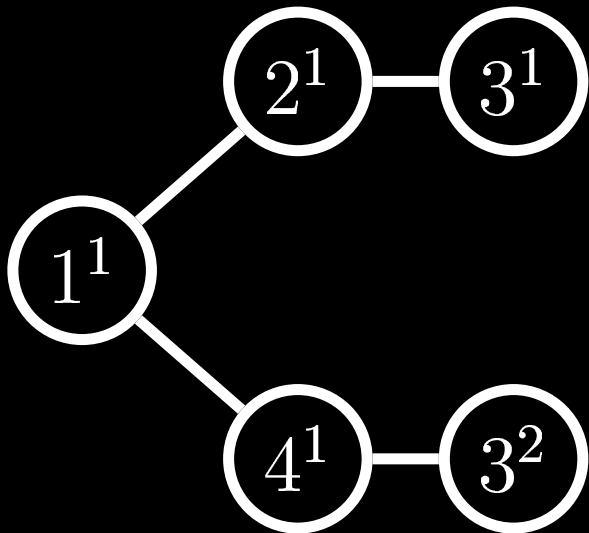
- Choose an arbitrary root node r . Initialize $T = r$.
- Repeat:
 - For each leaf u of T , find the neighbors of the corresponding node in G , other than the parent of u in T . Add these nodes to the tree.



Unwrapped Tree Construction

The unwrapped tree, T , is constructed from G as follows:

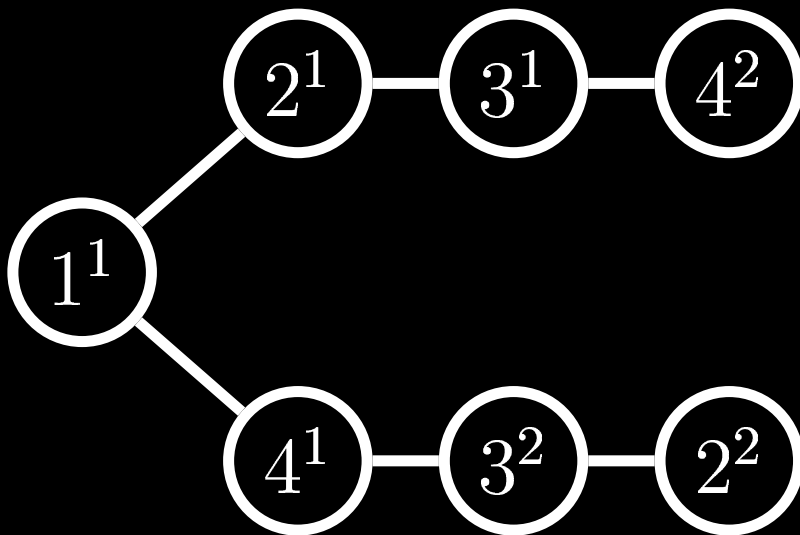
- Choose an arbitrary root node r . Initialize $T = r$.
- Repeat:
 - For each leaf u of T , find the neighbors of the corresponding node in G , other than the parent of u in T . Add these nodes to the tree.



Unwrapped Tree Construction

The unwrapped tree, T , is constructed from G as follows:

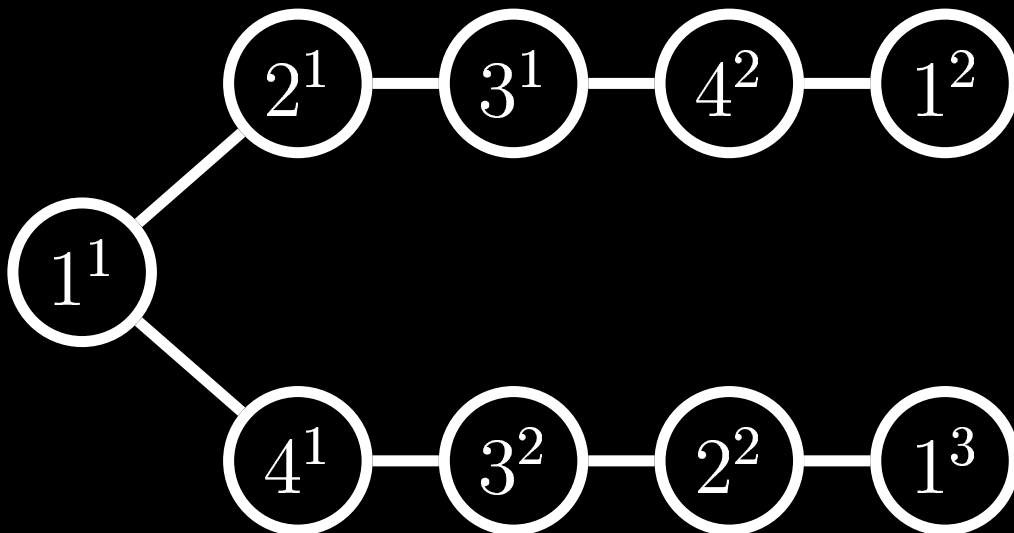
- Choose an arbitrary root node r . Initialize $T = r$.
- Repeat:
 - For each leaf u of T , find the neighbors of the corresponding node in G , other than the parent of u in T . Add these nodes to the tree.



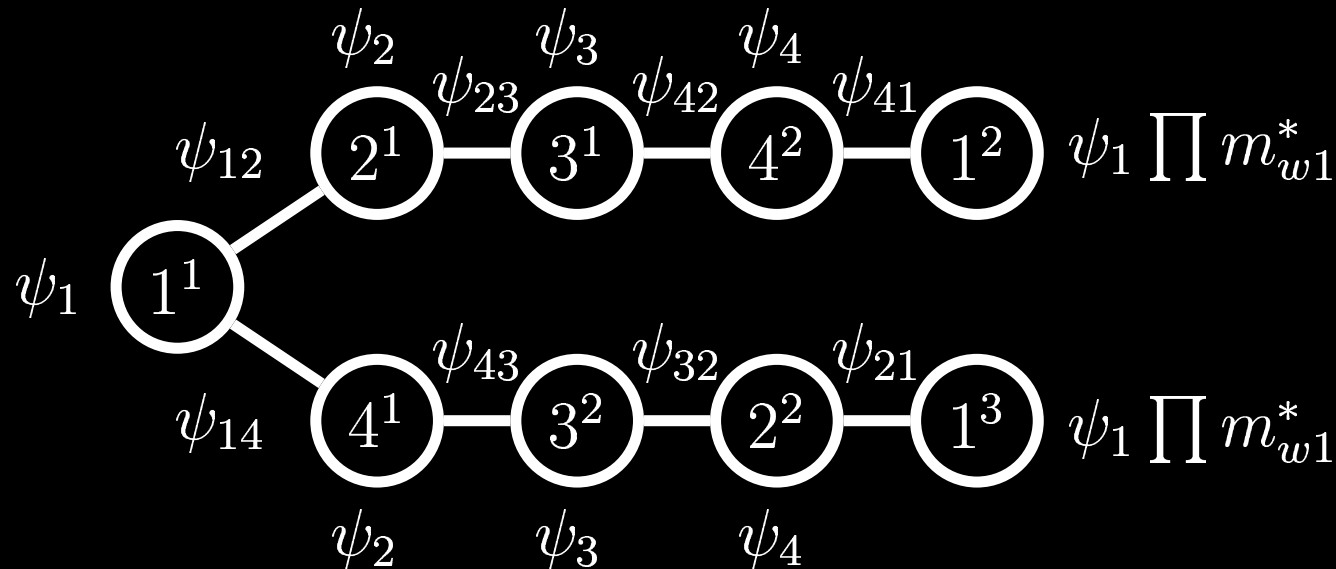
Unwrapped Tree Construction

The unwrapped tree, T , is constructed from G as follows:

- Choose an arbitrary root node r . Initialize $T = r$.
- Repeat:
 - For each leaf u of T , find the neighbors of the corresponding node in G , other than the parent of u in T . Add these nodes to the tree.



Unwrapped Tree Construction



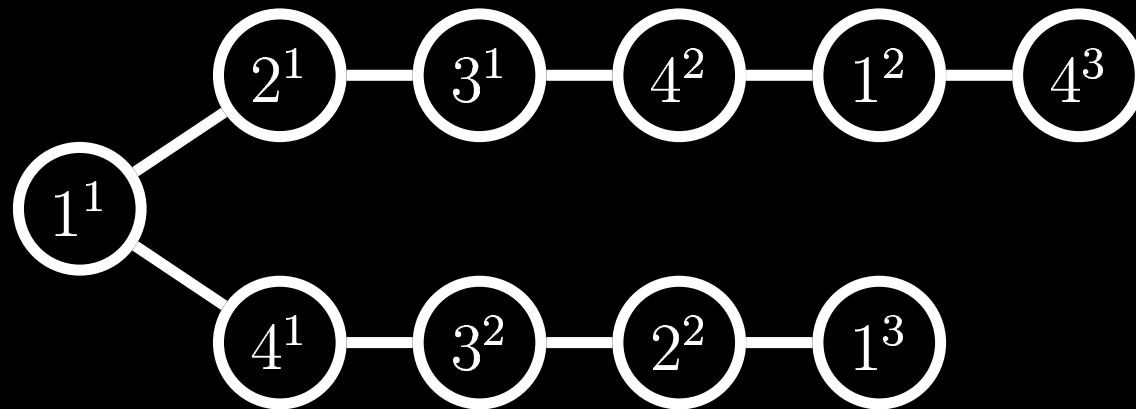
Copy the potentials from the corresponding nodes in G .
Modify the leaf single-node potentials to simulate the steady-state messages in G .

Because the graphs are locally the same, the \mathbf{x}_T^* will be replicas of \mathbf{x}_G^* .

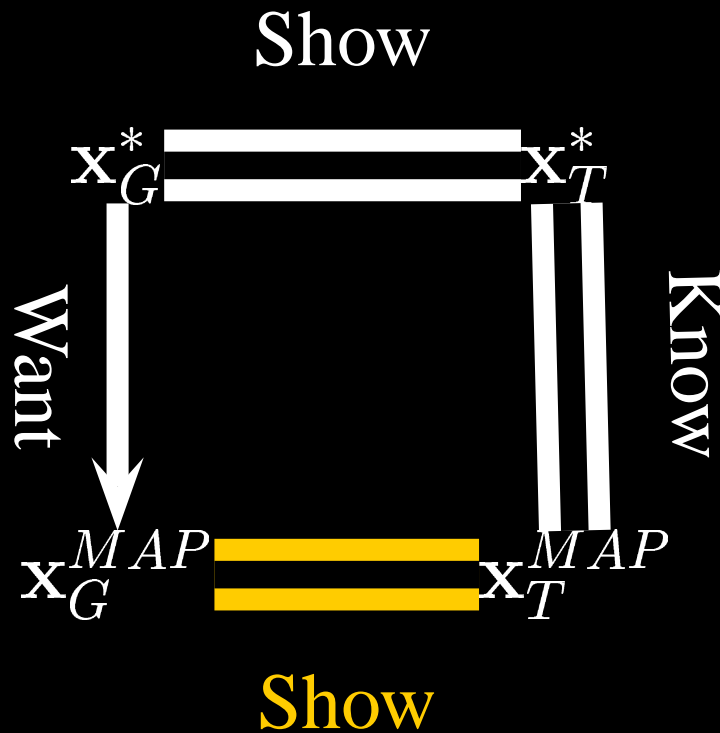
Graphs Containing a Single Cycle

For a graph containing a single cycle, the unwrapped tree is an **infinite chain**.

We can construct T so that each node is replicated n times in the interior.



Graphs Containing a Single Cycle



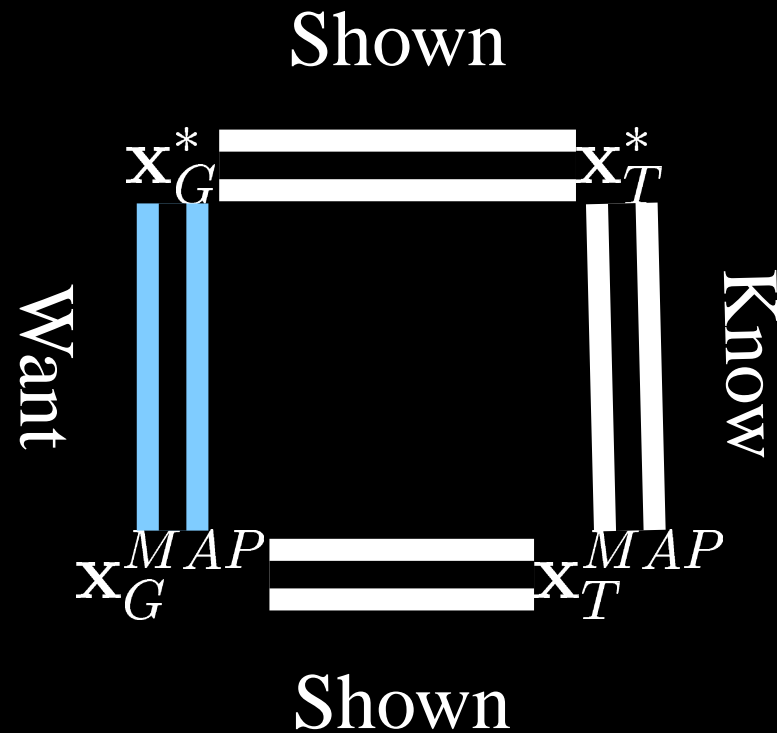
Graphs Containing a Single Cycle

Let $J_G(\mathbf{x})$ be the log-likelihood of assignment \mathbf{x} for G . Since the interior of T consists of n replicas of G , the log-likelihood for T is

$$J_T(\mathbf{x}) = nJ_G(\mathbf{x}) + \tilde{J}(\mathbf{x}_L),$$

where $\tilde{J}(\mathbf{x}_L)$ is the log-likelihood for the two leaf nodes. As $\tilde{J}(\mathbf{x}_L)$ does not depend on n , in the limit as $n \rightarrow \infty$, $J_T(\mathbf{x}) \propto J_G(\mathbf{x})$.

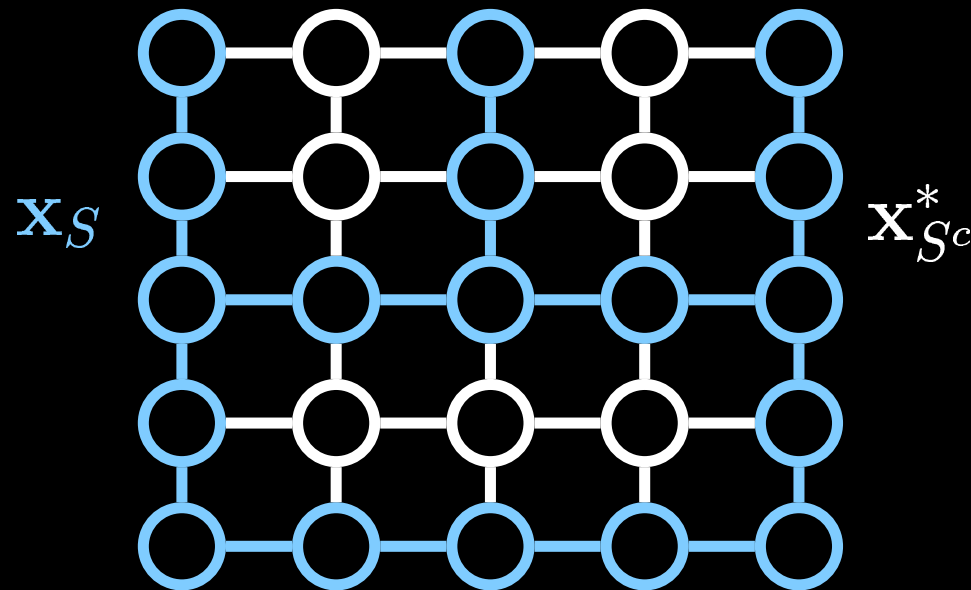
Graphs Containing a Single Cycle



Optimality for Arbitrary Graphs

Let S be a set of nodes whose induced subgraph contains at most one cycle per connected component.

We can show that \mathbf{x}^* has a higher probability than any $(\mathbf{x}_S, \mathbf{x}_{S^c}^*)$.



Outline

- Background.
 - Undirected graphical models.
 - Belief Propagation algorithm.
- Three techniques for analyzing loopy BP.
 - Algebraic analysis.
 - Unwrapped tree.
 - **Reparameterization.**
- Future work.

Reparameterization Analysis

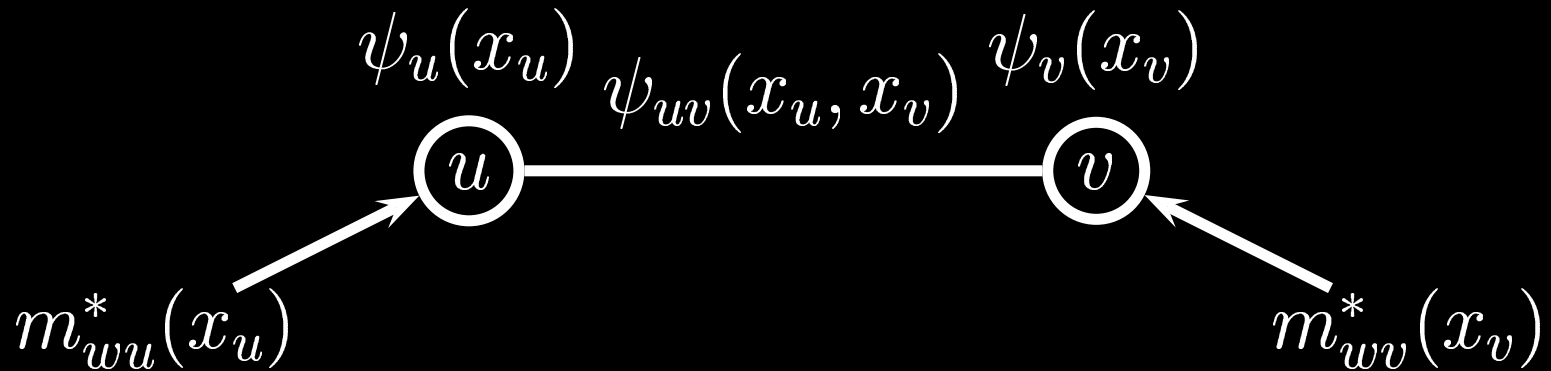
The past two analysis techniques analyzed the **message-passing** dynamics of BP.

The **reparameterization** technique analyzes the **steady-state beliefs**.

Reparameterization Overview

- We show that the beliefs define **another parameterization** of the distribution $p(\mathbf{x})$.
- In this parameterization,
 - We show that the steady-state beliefs are **consistent w.r.t every subtree**.
 - We show that the max-product assignment satisfies an **optimality condition w.r.t. every subgraph with at most one cycle per connected component**.

Steady-State Beliefs



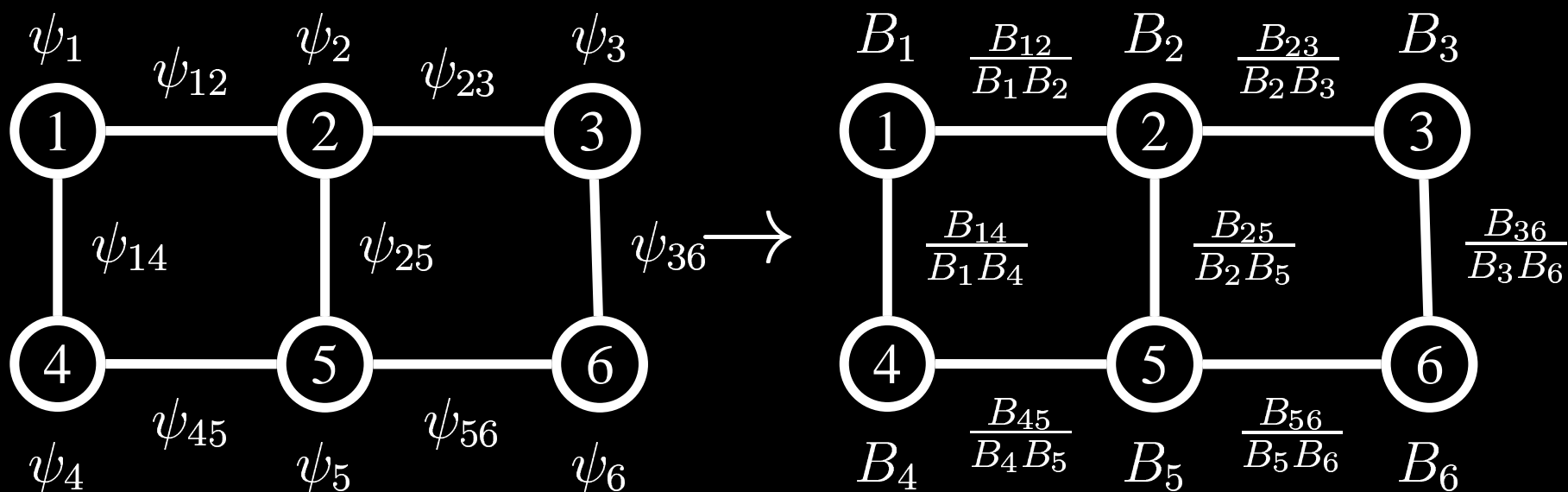
We analyze the steady-state single- and pair-node beliefs:

$$B_u^*(x_u) = \psi_u(x_u) \prod_{v \in N(u)} m_{vu}^*(x_u)$$

$$B_{uv}^*(x_u, x_v) = \psi_u(x_u) \psi_v(x_v) \psi_{uv}(x_u, x_v) \prod_{w \in N(u)/v} m_{wu}^*(x_u) \prod_{w \in N(v)/u} m_{wv}^*(x_v).$$

Belief Parameterization

The beliefs B^* define another parameterization of the distribution:



Belief Parameterization

The beliefs \mathbf{B}^* define **another parameterization** of the distribution:

$$\begin{aligned} p(\mathbf{x}; \mathbf{B}^*) &= \alpha \prod_{u \in V} B_u^*(x_u) \prod_{(u,v) \in E} \frac{B_{uv}^*(x_u, x_v)}{B_u^*(x_u) B_v^*(x_v)} \\ &= \alpha \prod_{u \in V} \psi_u(x_u) \prod_{(u,v) \in E} \psi_{uv}(x_u, x_v) \\ &= p(\mathbf{x}; \Psi) \end{aligned}$$

as can be shown by substituting in the definition of \mathbf{B}^* .

Consistency

Definition: Let $H = (V_H, E_H)$ be a subgraph with distribution

$$p_H(\mathbf{x}_{V_H}; \mathbf{B}_H^*) = \alpha \prod_{u \in V_H} B_u^*(x_u) \prod_{(u,v) \in E_H} \frac{B_{uv}^*(x_u)}{B_u^*(x_u) B_v^*(x_v)}.$$

The beliefs \mathbf{B}^* are **consistent** w.r.t H if the corresponding beliefs \mathbf{B}_H^* are the true max-marginals of $p_H(\mathbf{x}_{V_H}; \mathbf{B}_H^*)$.

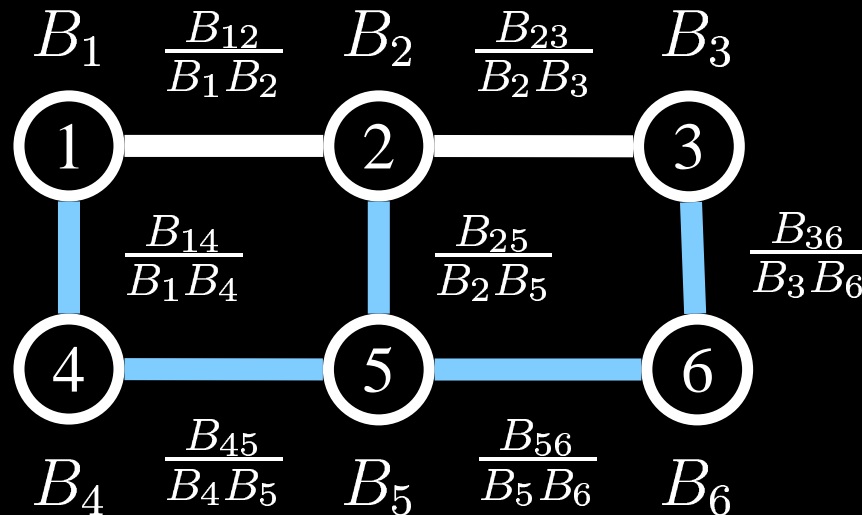
Edge Consistency

The edge beliefs are **consistent**:

$$\max_{x_v} B_{uv}^*(x_u, x_v) = B_u^*(x_u),$$

as can be seen by substituting in the message definitions of B_{uv}^* and B_u^* .

Tree Consistency

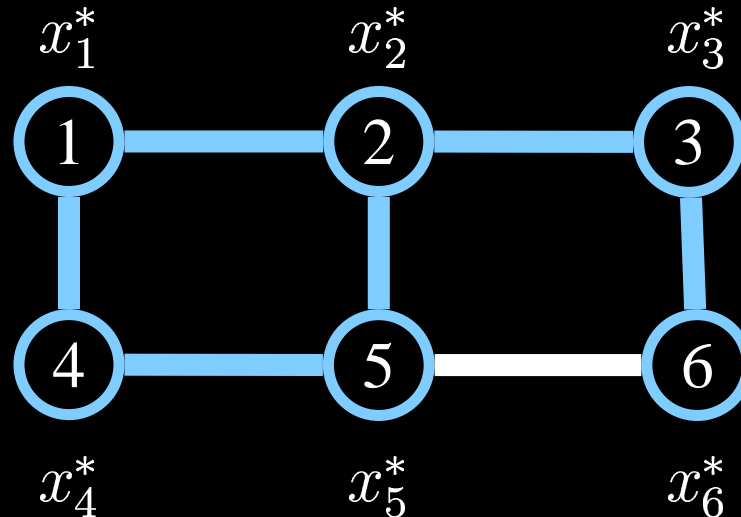


The steady-state beliefs \mathbf{B}^* are consistent w.r.t **every subtree** T of G .

This is shown by exploiting the edge consistency described to remove leaf nodes one at a time.

In the end, we are left with only two nodes, a trivial base case.

Tree Plus Cycle Optimality



Let $H = (V_H, E_H)$ be a subgraph of G with at most one cycle per connected component and distribution

$$p_H(\mathbf{x}_{V_H}; \mathbf{B}_H^*) = \alpha \prod_{u \in V_H} B_u^*(x_u) \prod_{(u,v) \in E_H} \frac{B_{uv}^*(x_u)}{B_u^*(x_u) B_v^*(x_v)}.$$

The max-product assignment $\mathbf{x}_{V_H}^*$ maximizes p_H .

Tree Plus Cycle Optimality

Using the edge consistency described, we can show that

$$\log \frac{B_{uv}^*(x_u, x_v)}{B_v^*(x_v)} \leq \log \frac{B_{uv}^*(x_u^*, x_v^*)}{B_v^*(x_v^*)}$$

for any other assignment x_u, x_v .

Tree Plus Cycle Optimality

If H is a connected subgraph containing one cycle, then the edges of H can be directed so that **each node has exactly one parent**:

$$\begin{aligned}\log p(\mathbf{x}_H; \mathbf{B}_H^*) &= \sum_{u \in V_H} \log \frac{B_{uv}^*(x_u, x_v)}{B_v^*(x_v)} \\ &\leq \sum_{u \in V_H} \log \frac{B_{uv}^*(x_u^*, x_v^*)}{B_v^*(x_v^*)} \\ &= \log p(\mathbf{x}_H^*; \mathbf{B}_H^*)\end{aligned}$$

where v is the parent of u .

Corollaries of TPS Optimality

The two results proved using the **unwrapped tree** are corollaries of the Tree-Plus-Cycle optimality.

The Tree-Plus-Cycle optimality can also be used to show an **error bound** on the max-product assignment for an arbitrary graph.

Future Work

I have presented three techniques for analyzing loopy BP. Experimental results are better than the results proved. Future work includes extending each technique to be more general and prove stronger results.

- Prove convergence properties of the max-product algorithm on a single-cycle graph using algebraic analysis.
- Prove the optimality of the max-product algorithm for specific multiple-loop graphs using the unwrapped tree technique.
- Show more powerful optimality results for arbitrary graph structures with specific potential properties.

References

- Aji, S., Horn, G., and McEliece, R. (1998). On the convergence of iterative decoding on graphs with a single cycle. In *IEEE International Symposium on Information Theory*.
- McEliece, R., Rodemich, E., and Cheng, J. (1995). The Turbo decision algorithm. In *33rd Allerton Conference on Communications, Control and Computing*, Monticello, IL.
- Murphy, K., Weiss, Y., and Jordan, M. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*, pages 467–475.
- Pearl, J. (1998). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA.
- Wainwright, M. (January, 2002). *Stochastic Processes on Graphs with Cycles: Geometric and Variational Approaches*. PhD thesis, MIT, Cambridge, MA.
- Wainwright, M., Jaakola, T., and Willsky, A. (2003). Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5).
- Wainwright, M., Jaakola, T., and Willsky, A. (October 28, 2002). Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. Technical Report P-2554, Laboratory for Information and Decision Systems, MIT.
- Weiss, Y. (November, 1997). Belief propagation and revision in networks with loops. Technical Report AI Memo No. 1616, C.B.C.L. Paper No. 155, AI Lab, MIT.
- Weiss, Y. and Freeman, W. (2001a). Correctness of belief propagation in Gaussian graphical models or arbitrary topology. *Neural Computation*, 13:2173–2200.
- Weiss, Y. and Freeman, W. (2001b). On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):723–735.