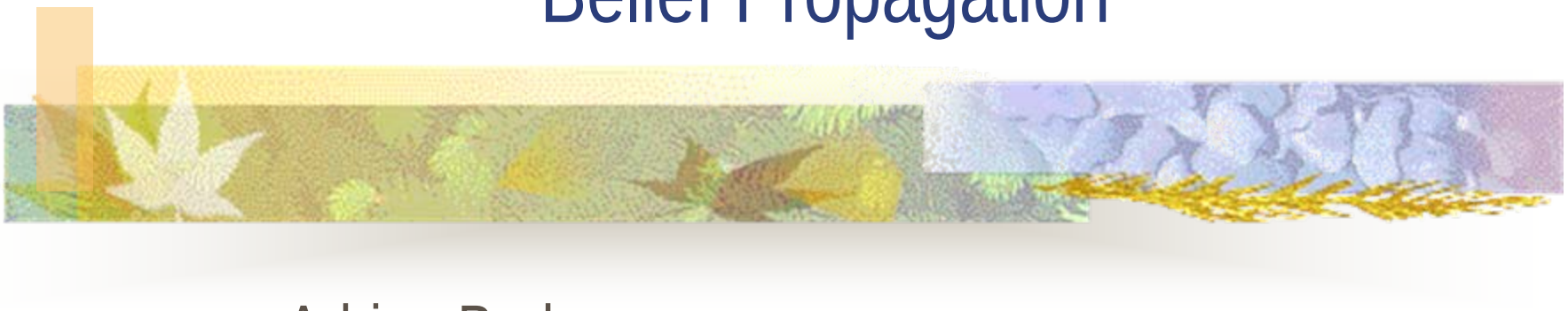
Belief Propagation



Adrian Barbu

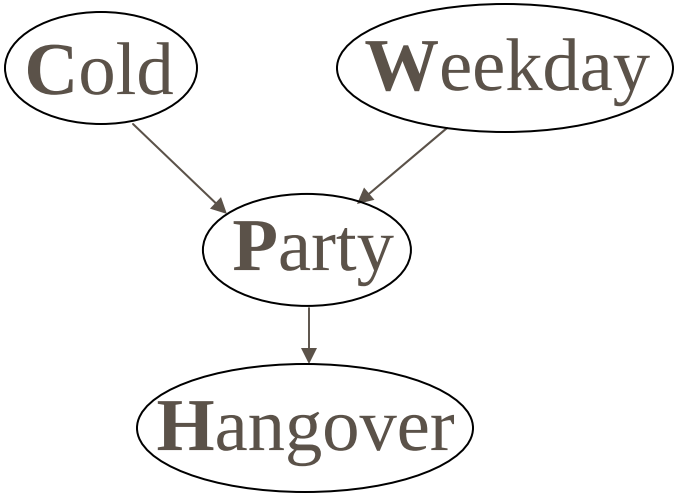
Belief Propagation

- Approximate inference in MRF or Directed Graphical Models
- General form of the probability function

$$P(\mathbf{x}) = P(x_1, ..., x_N) = \frac{1}{Z} \prod_{c \in C} \psi_c(\mathbf{x}_c)$$

- Message Passing Algorithm
- Perform Message Passing
  - Run until convergence
  - Not always converges

Example



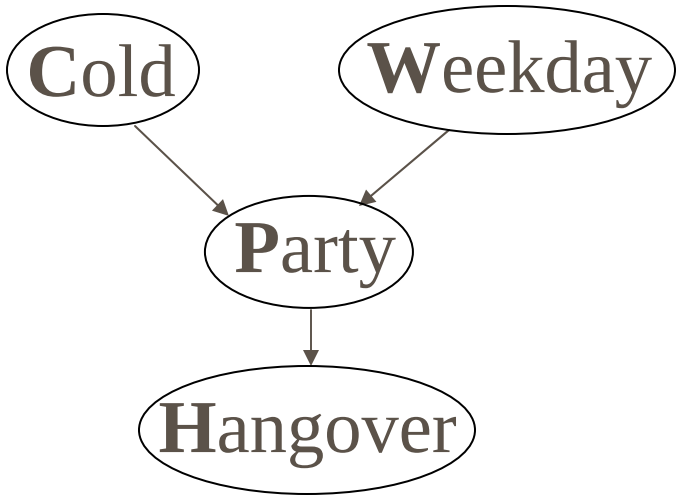
Joint Probability: $p(x_H, x_P, x_C, x_W) =$

$$p(x_H \mid x_P)\, p(x_P \mid x_C, x_W)\, p(x_C)\, p(x_W)$$

Marginal Probability: $p(x_H) = \sum_{x_P, x_C, x_W} p(x_H, x_P, x_C, x_W)$

8-sum

Example (cont.)



Marginal Probability:

$$p(x_H) = \sum_{x_P, x_C, x_W} p(x_H, x_P, x_C, x_W)$$

8-sum

Localize probabilities:

$$p(x_P) = \sum_{x_C, x_W} p(x_P \mid x_C, x_W) p(x_C) p(x_W) \leftarrow 4\text{-sum}$$

$$p(x_H) = \sum_{x_P} p(x_H \mid x_P) p(x_P) \leftarrow 2\text{-sum}$$

Approach

- GM: Define variables and connections

$$x_C, x_W, x_P, x_H; p(x_C), p(x_W), p(x_P \mid x_C, x_W), p(x_H \mid x_P)$$

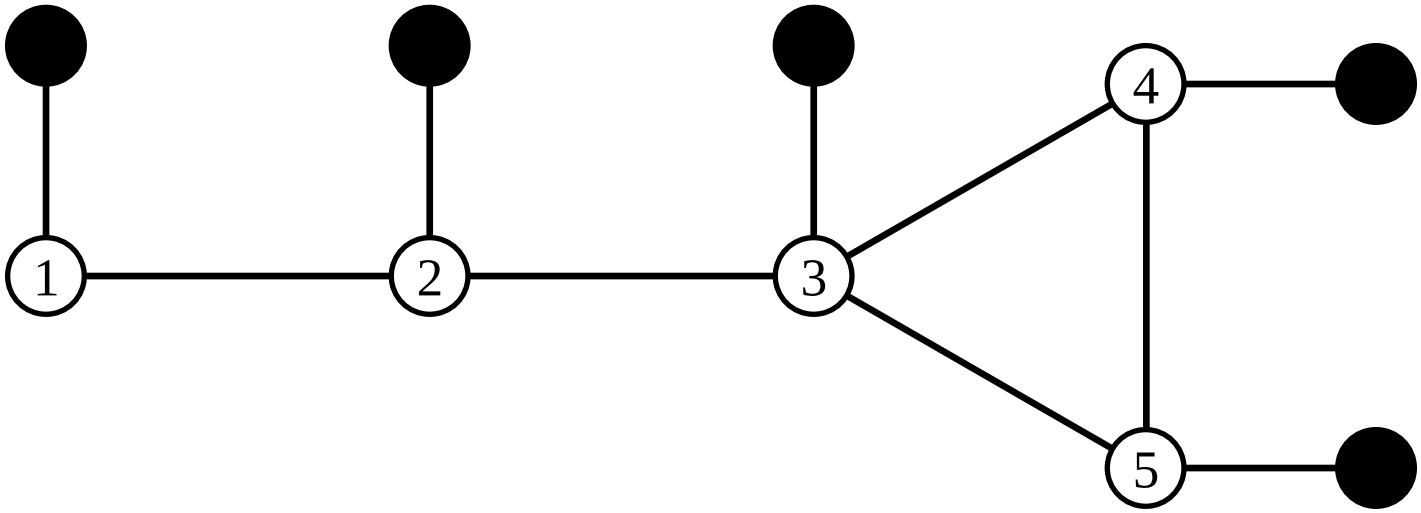
- Calculate marginal probabilities efficiently

$$p(x_H) = \sum_{x_C, x_W, x_P} p(x_C, x_W, x_P, x_H)$$

- Find most likely configuration

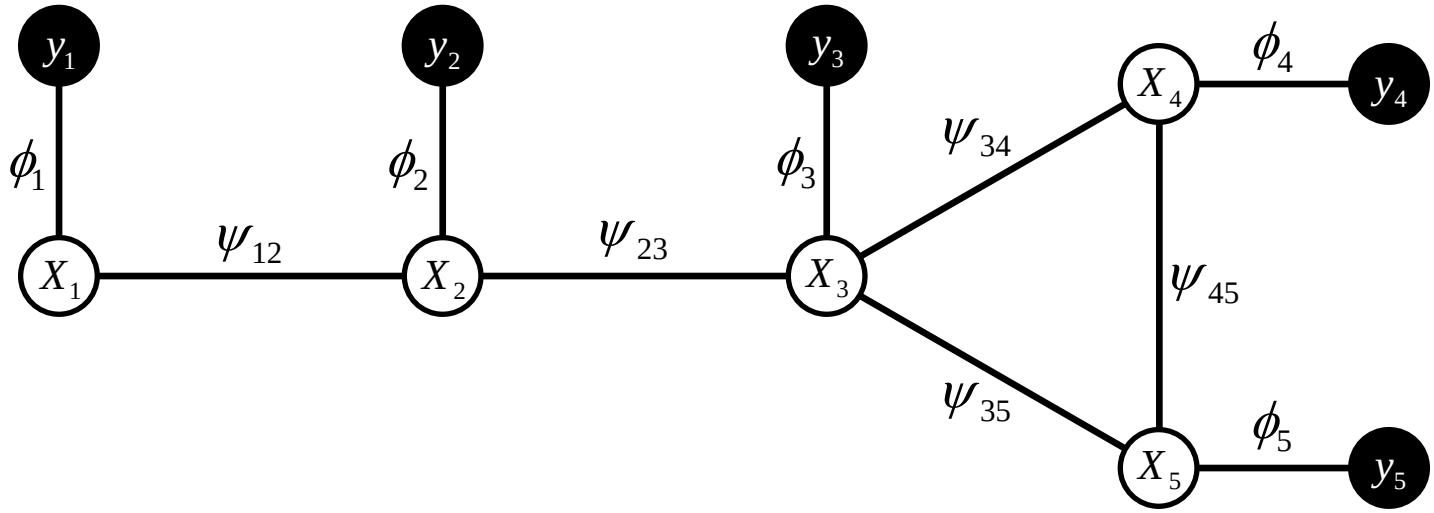
$$\arg\max_{x_H} p(x_H)$$

Pairwise Markov Random Field



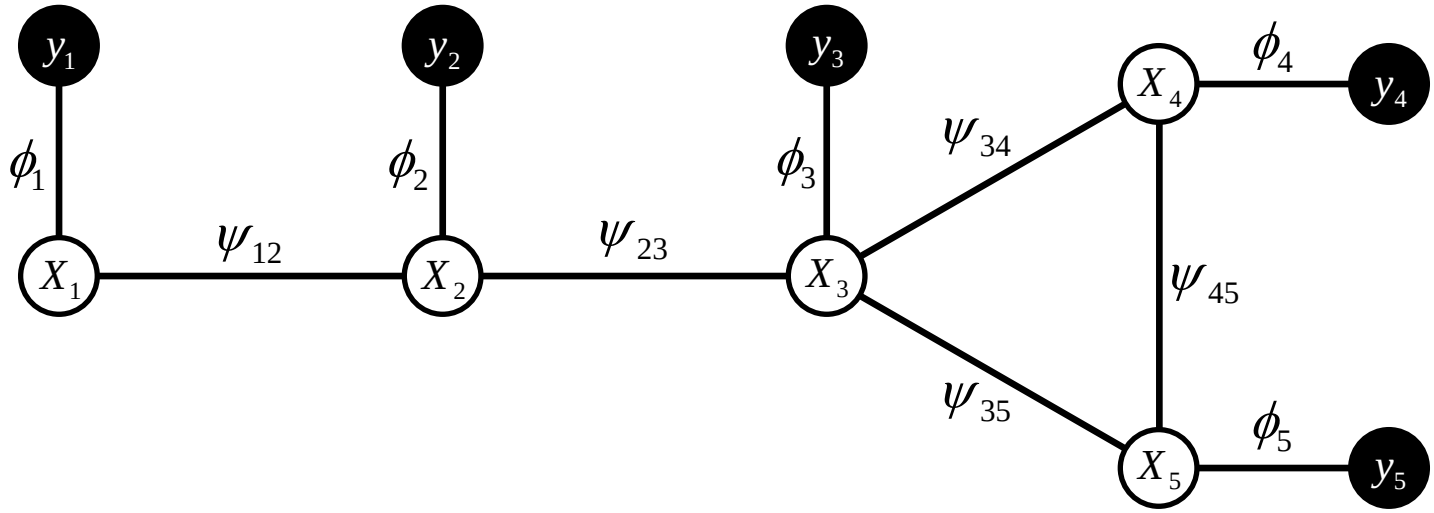
- Basic structure: vertices, edges

Pairwise Markov Random Field



- Basic structure: vertices, edges
- Vertex *i* has set of possible states X_i and observed value y_i
- Compatibility between states and observed values, $\phi_i(x_i, y_i)$
- Compatibility between neighboring vertices *i* and *j*, $\psi_{ij}(x_i, x_j)$

Pairwise MRF: Probabilities



- Joint probability: $$p(x_1,\ldots,x_5)=\frac{1}{Z}\prod_{i=1}^{5}\phi_i(x_i,y_i)\prod_{(ij)}\psi_{ij}(x_i,x_j)$$
- Marginal probability: $$p_i(x_i)=\sum_{x_j\in X_j,1\le j\le 5,j\neq i}p(x_1,\ldots,x_5)$$
  - Advantage: allows average over ambiguous states
  - Disadvantage: complexity exponential in number of vertices

BP on a Chain

$$p(\mathbf{x}) = \frac{1}{Z}\exp(-E(\mathbf{x})), E(\mathbf{x}) = \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{ij}(x_i, x_j)$$

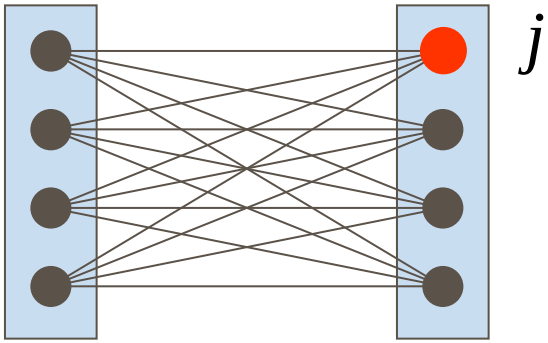
- Dynamic programming: global minimum of H in linear time
- BP:
  - Inward pass (dynamic programming)
  - Outward pass
  - Gives min-marginals

[Pearl'88]

$$\min_{\mathbf{x}} \{E(\mathbf{x}) | x_q = j\}$$

Inward Pass (Dynamic Programming)

$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$

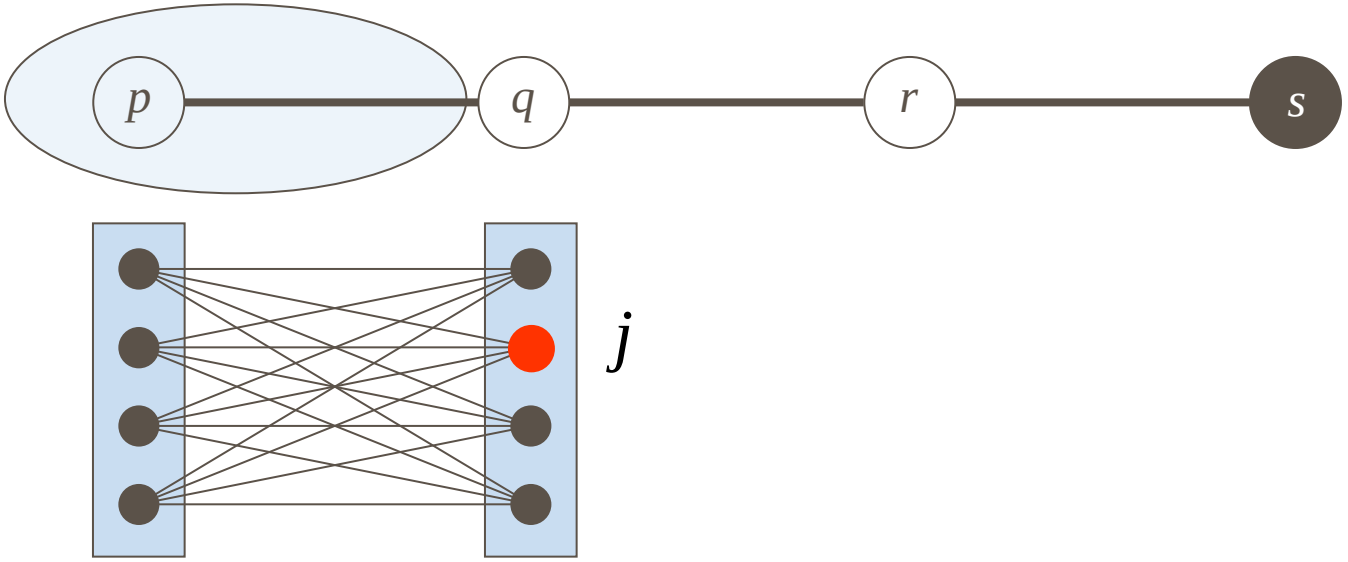


$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)

$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$

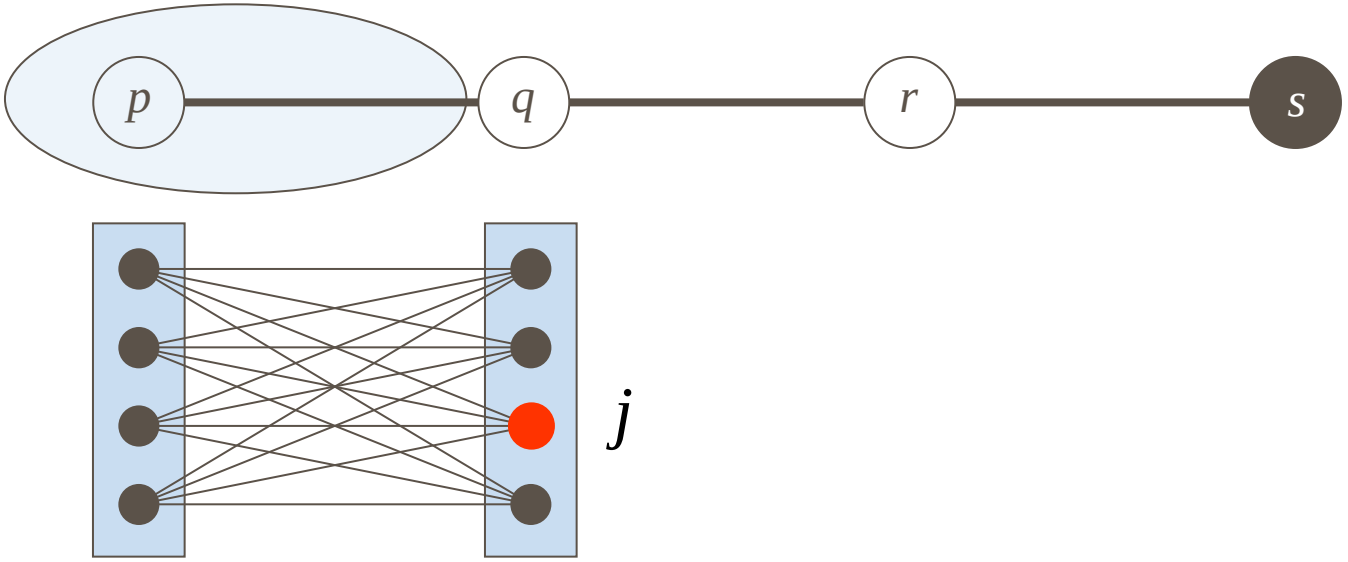


$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)

$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$

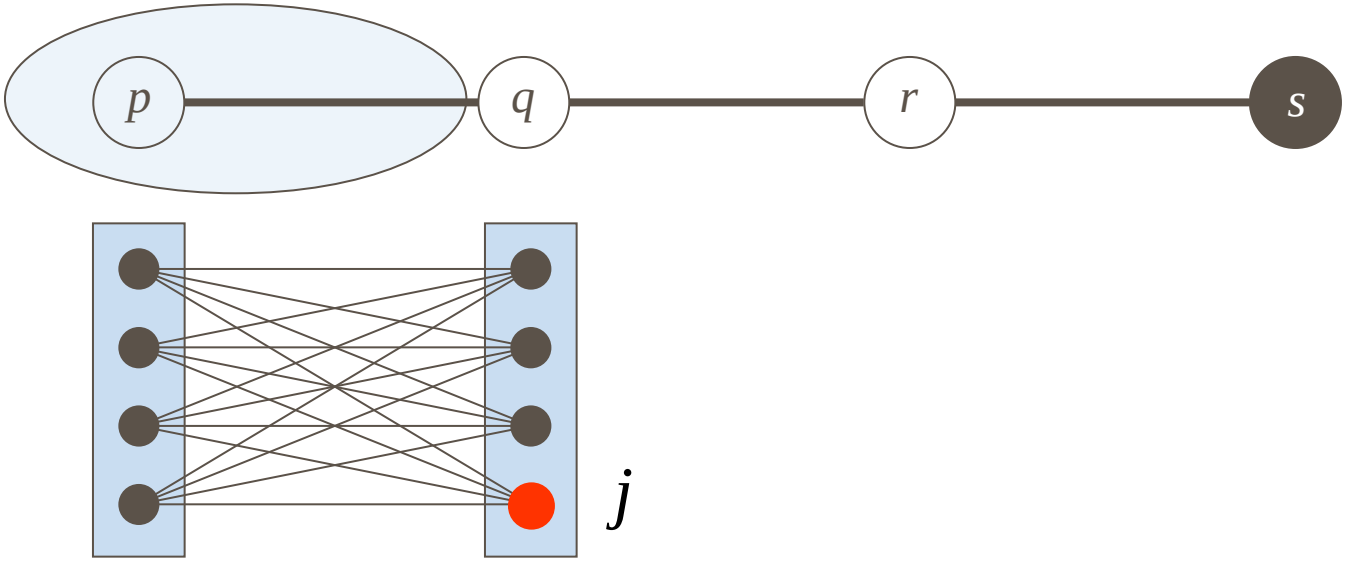


$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)

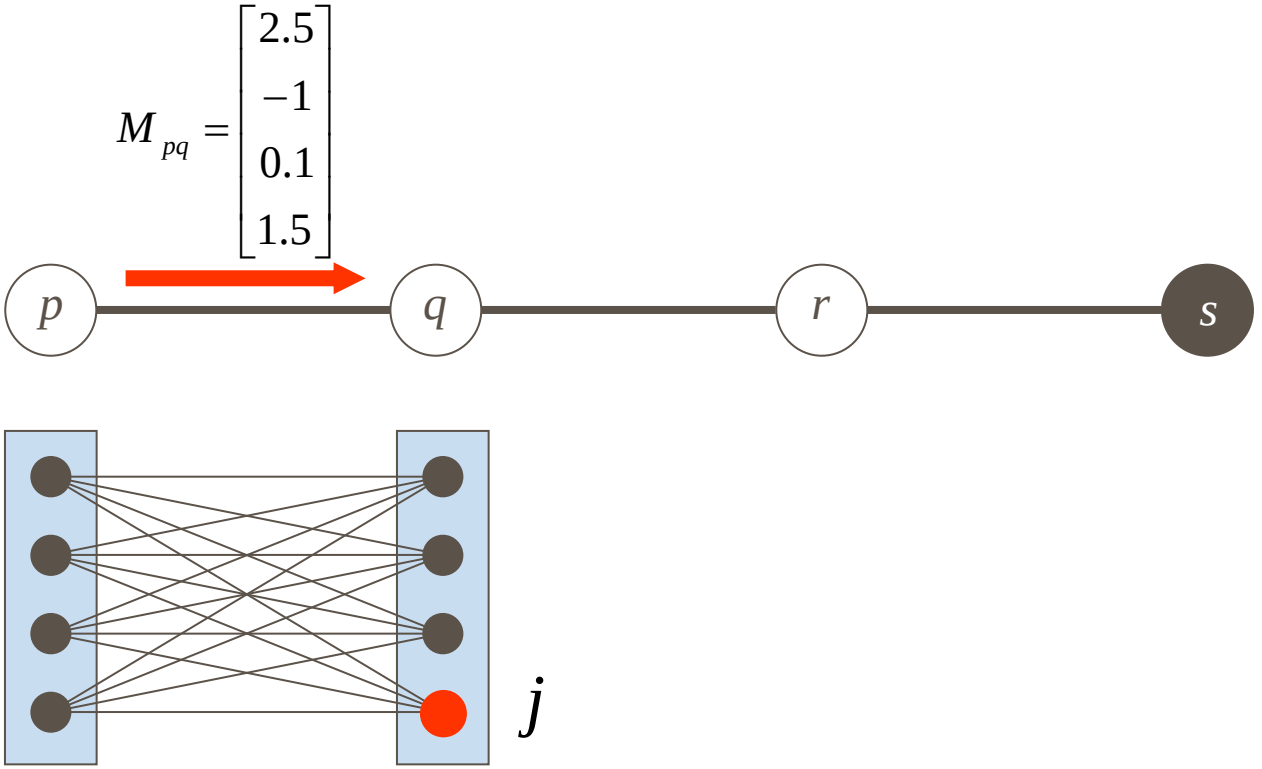
$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$



$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

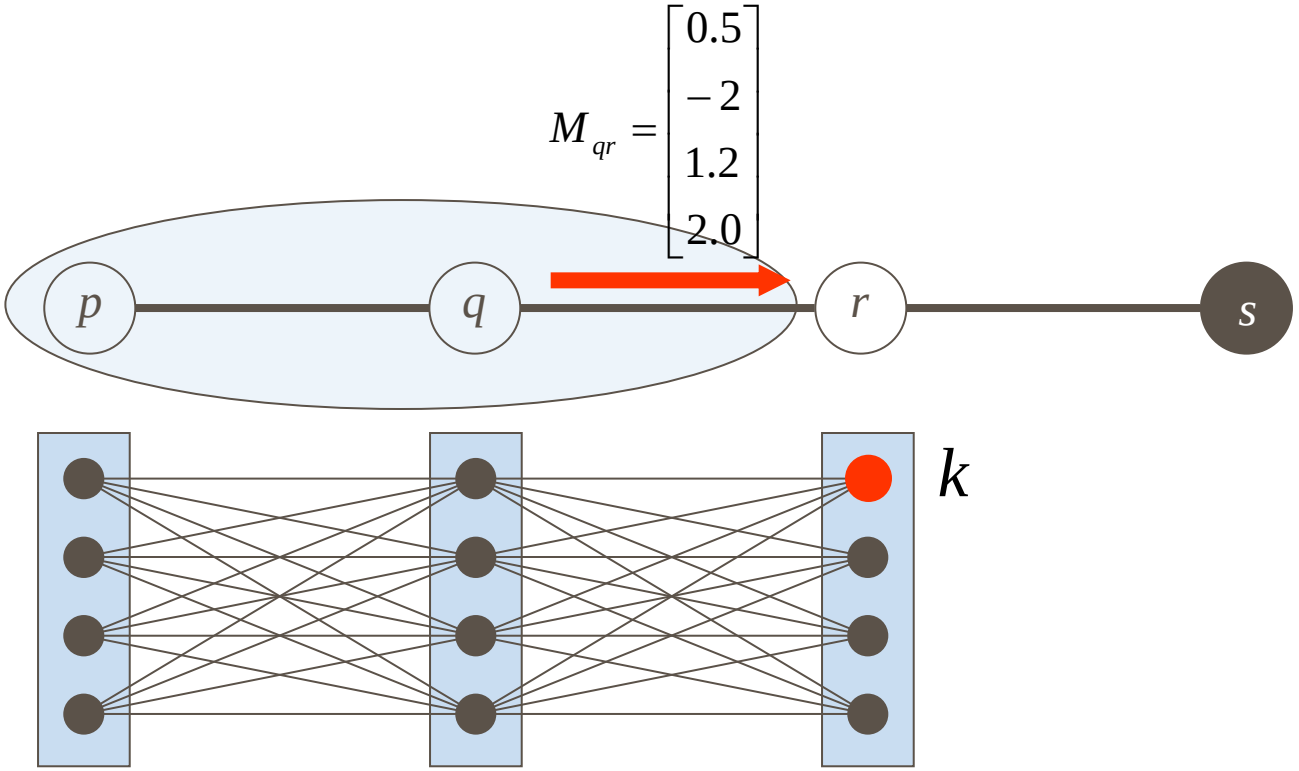
Inward Pass (Dynamic Programming)



$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

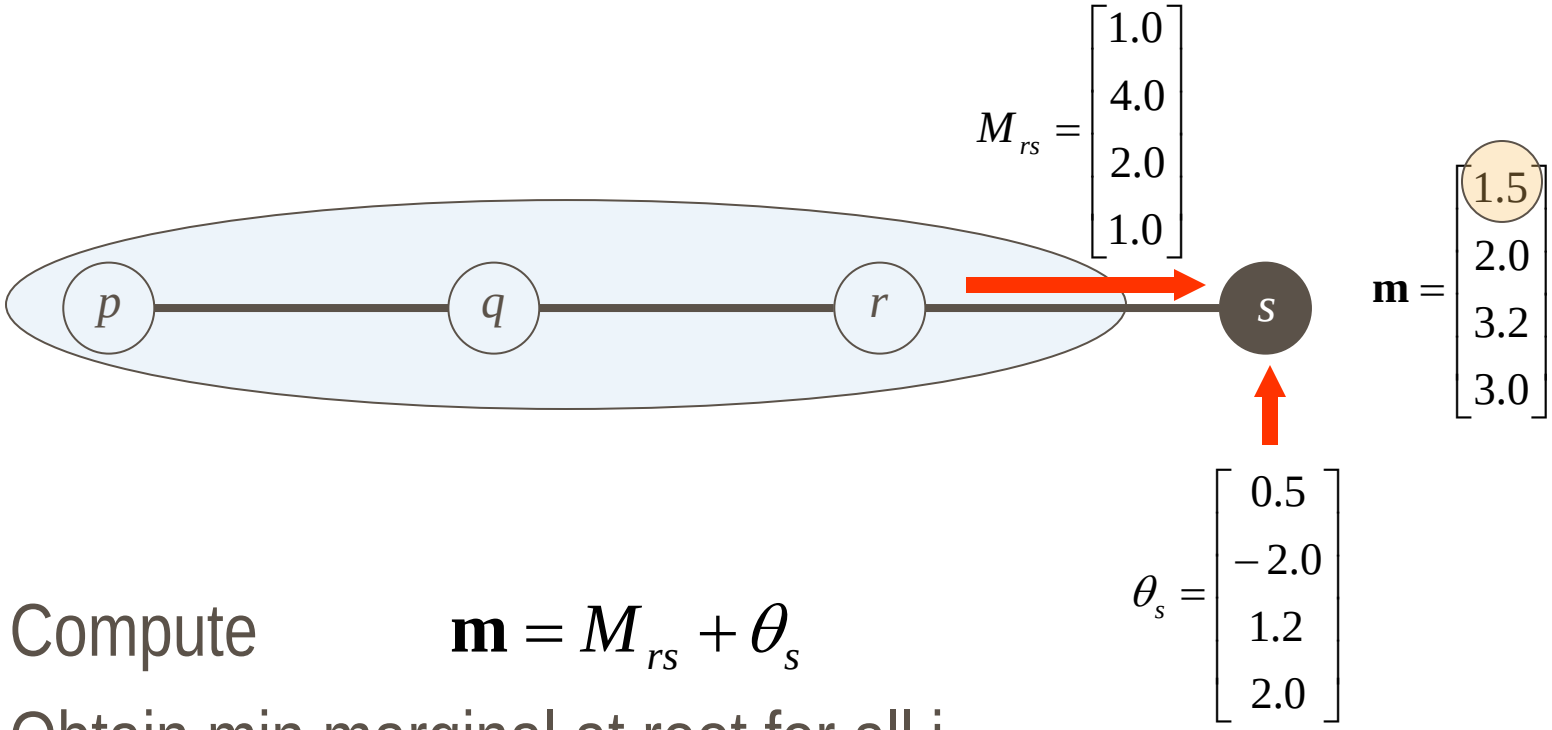
Inward Pass (Dynamic Programming)



$$M_{qr}(k) = \min_j \left\{ \left(\theta_q(j) + M_{pq}(j) \right) + \theta_{qr}(j,k) \right\}$$

Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)



- Compute $\mathbf{m} = M_{rs} + \theta_s$
- Obtain min marginal at root for all j

$$m_j = \min_{\mathbf{x}} \{ \mathrm{E}(\mathbf{x}) \mid x_s = j \}$$

- Find j with minimum E: s*=1

Outward Pass

$$r^* = \arg\min_j [\theta_r(j) + M_{qr}(j) + \theta_{rs}(j, s^*)]$$

Slide credit: V. Kolmogorov

Outward Pass

$$q^* = \arg\min_j [\theta_q(j) + M_{pq}(j) + \theta_{qr}(j, r^*)]$$

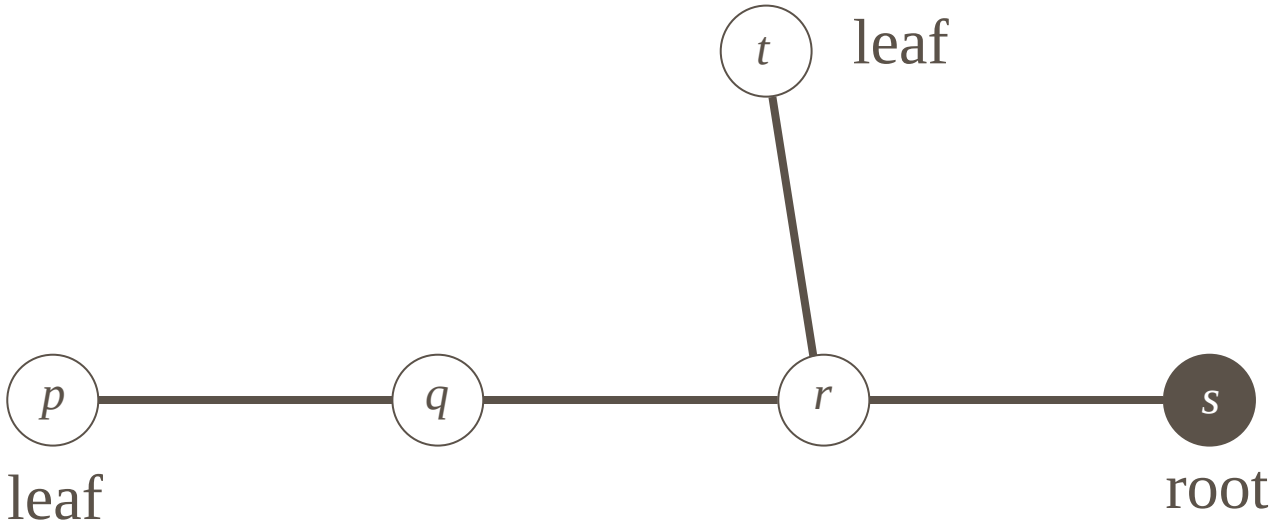
Slide credit: V. Kolmogorov

Outward Pass

$$p^* = \arg\min_j [\theta_p(j) + \theta_{pq}(j, q^*)]$$

Slide credit: V. Kolmogorov

BP on a Tree



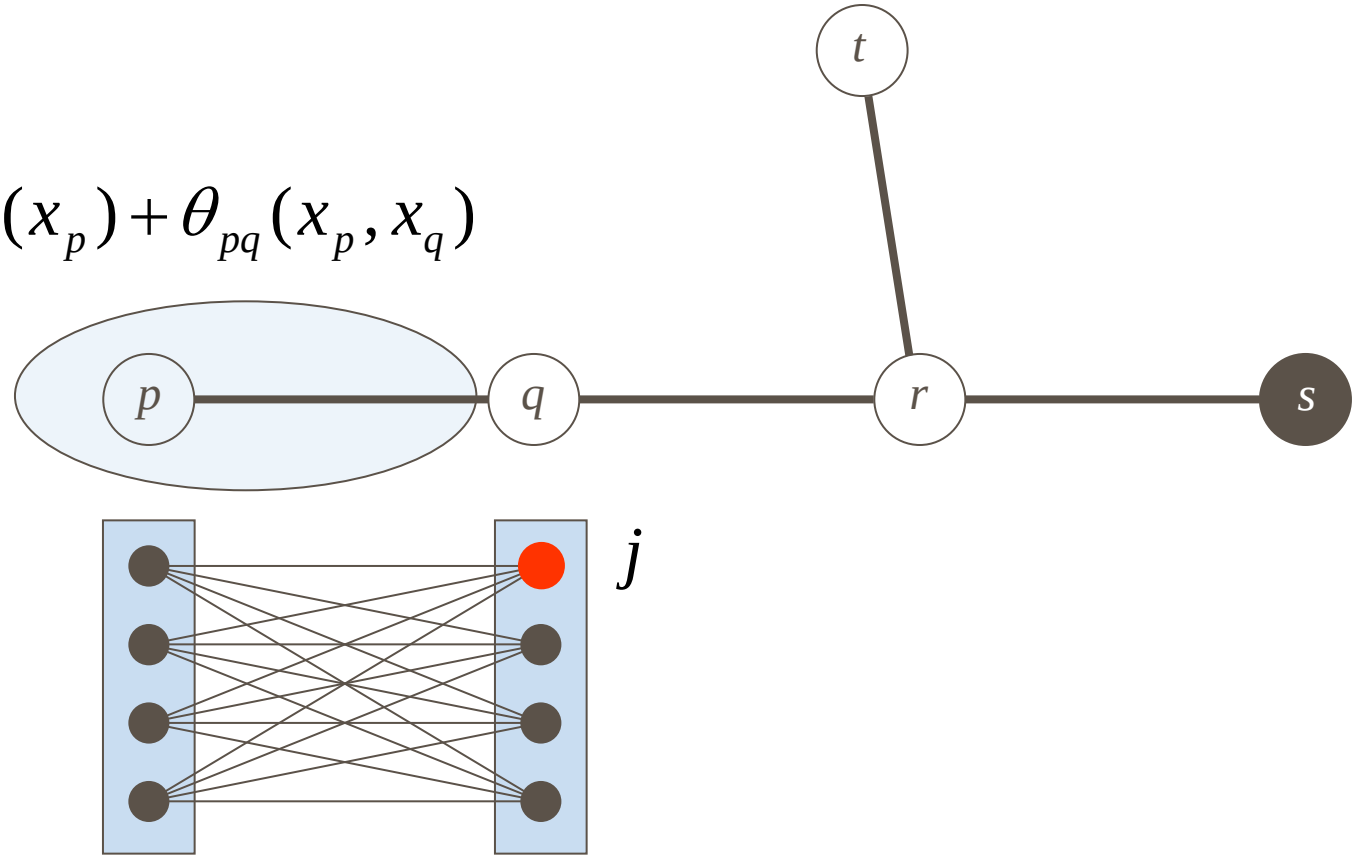
- Dynamic programming: global minimum in linear time
- BP:
  - Inward pass (dynamic programming)
  - Outward pass
  - Gives min-marginals

$$\min_{\mathbf{x}} \{E(\mathbf{x}) | x_q = j\}$$

[Pearl'88]

Inward Pass (Dynamic Programming)

$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$

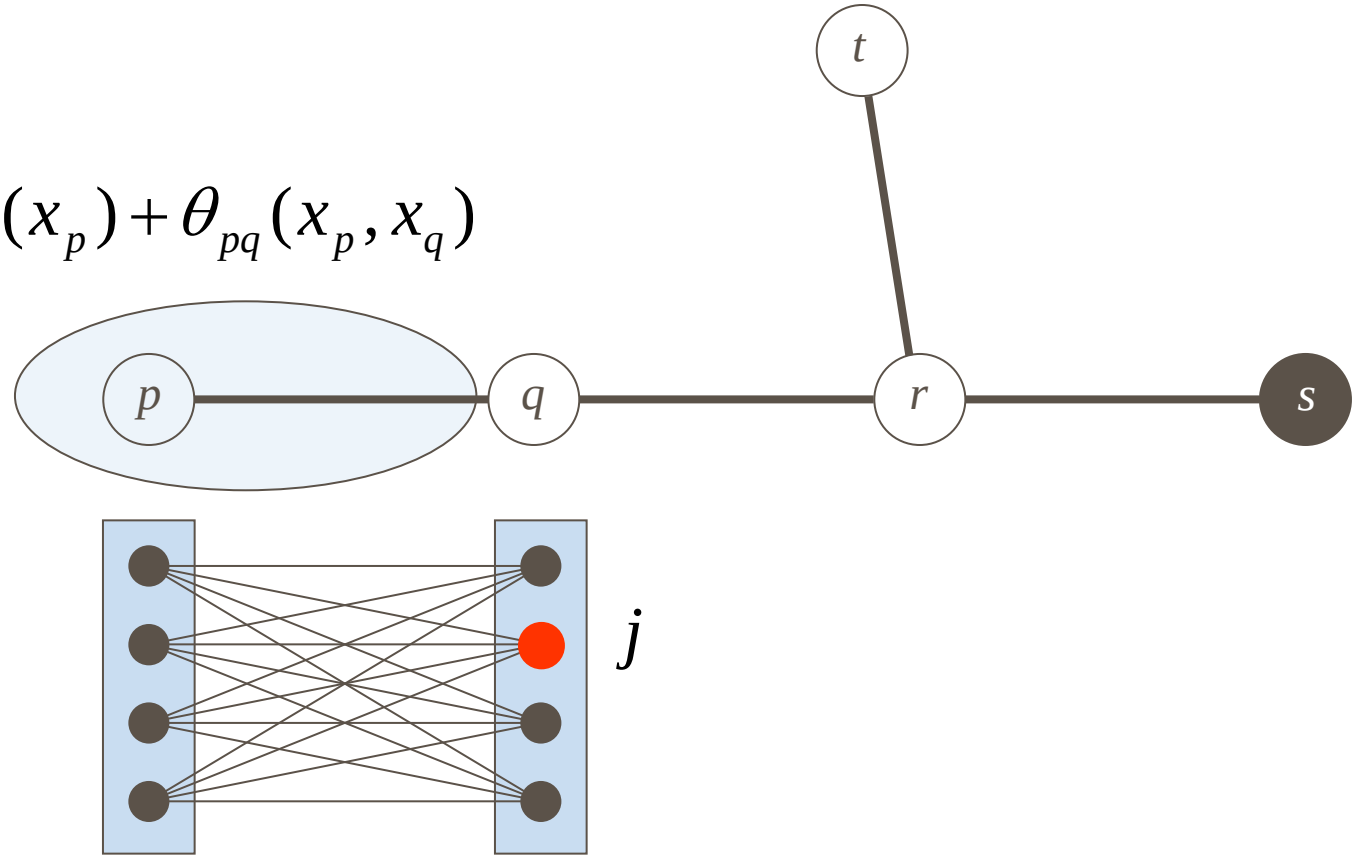


$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)

$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$

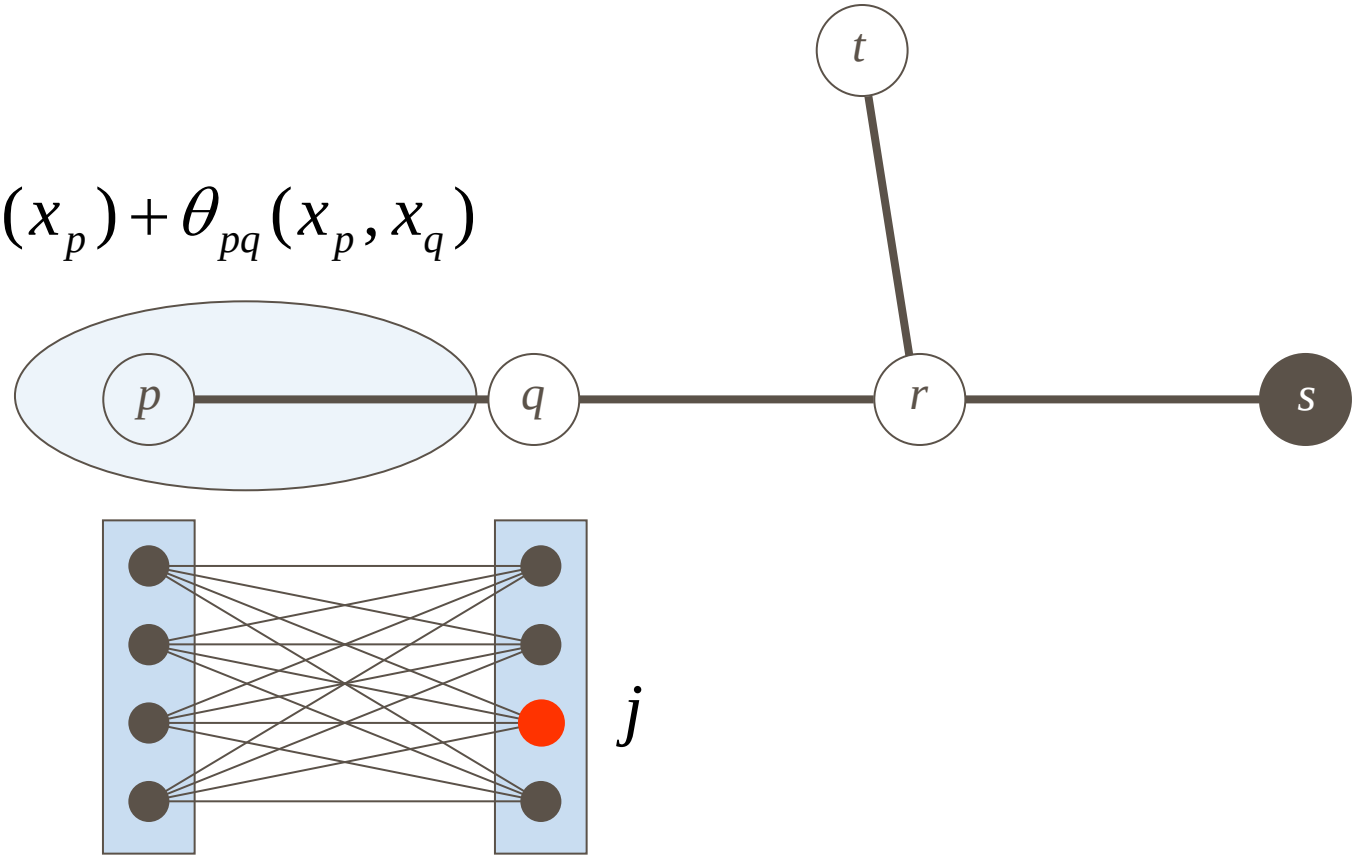


$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)

$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$

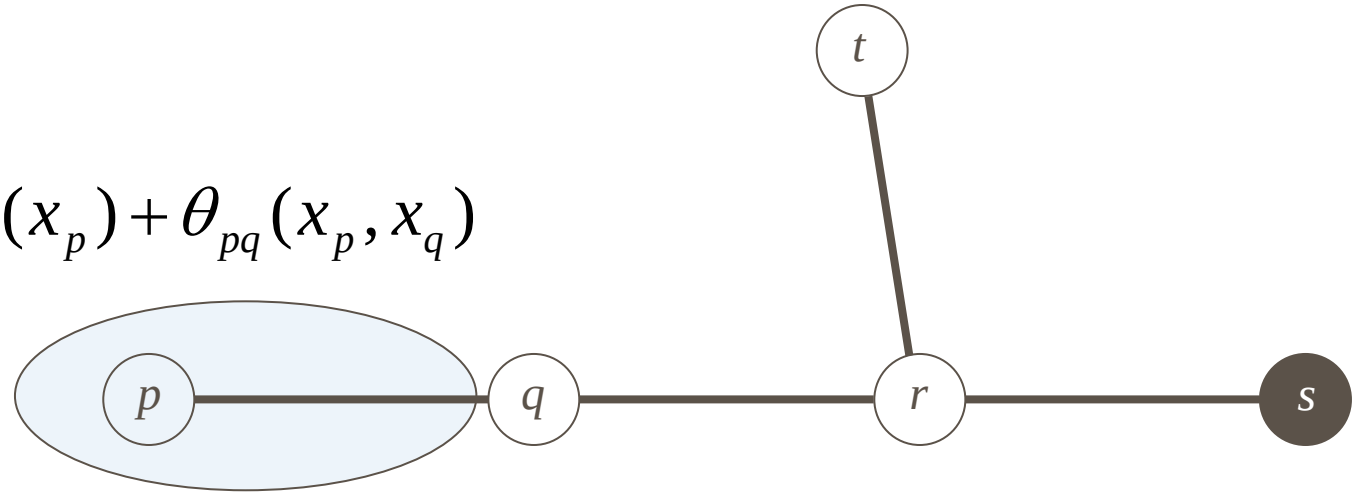


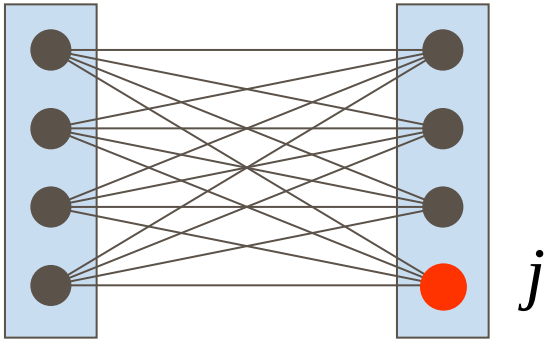
$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)

$$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$$

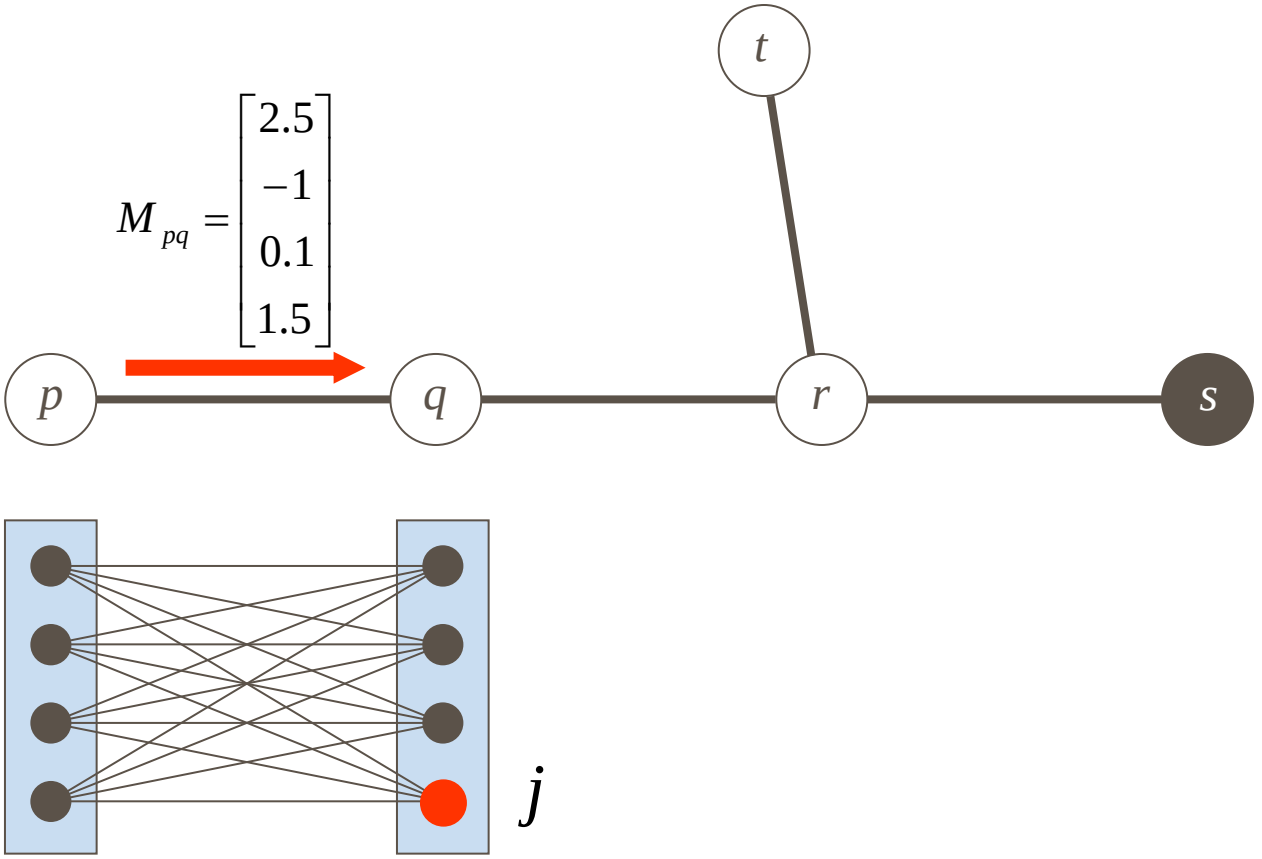




$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

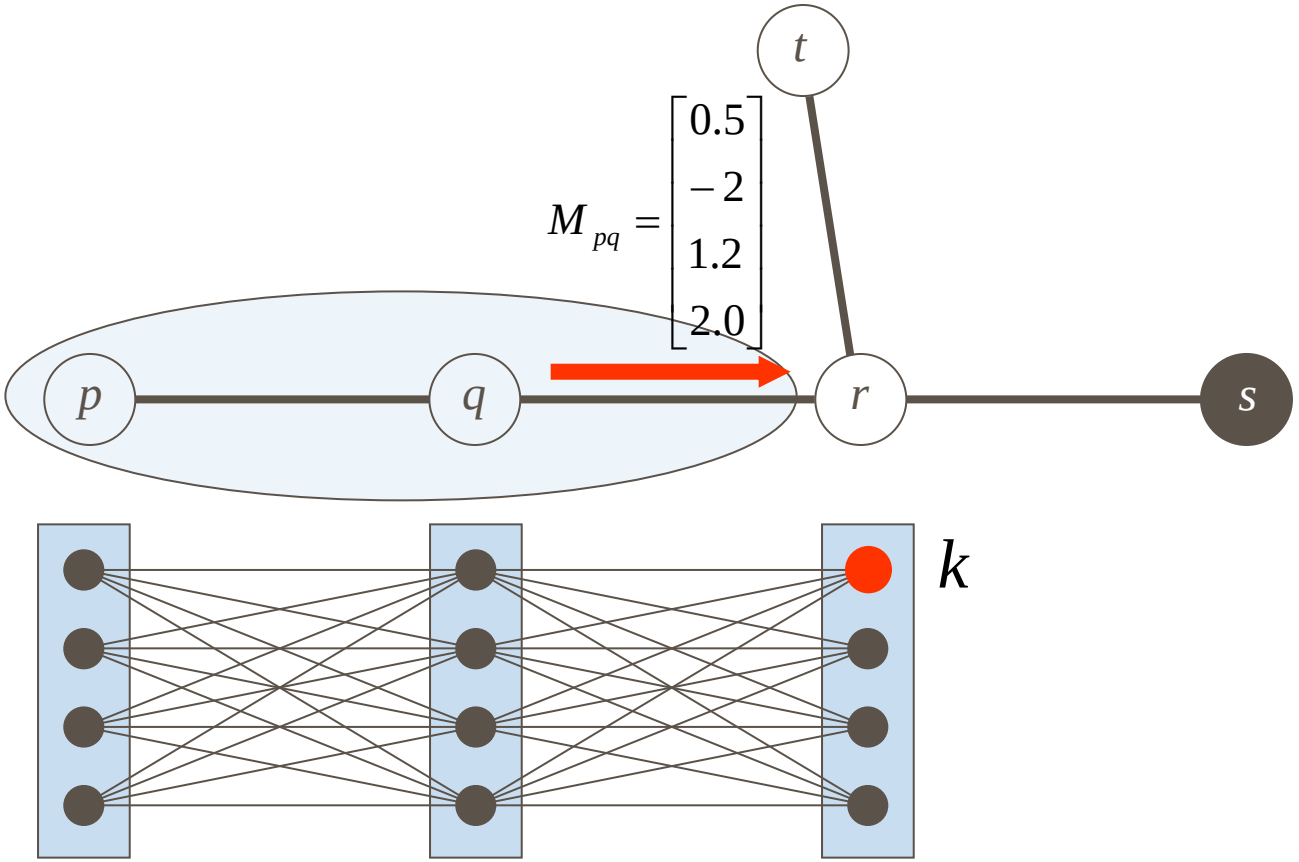
Inward Pass (Dynamic Programming)



$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i, j) \right\}$$

Slide credit: V. Kolmogorov

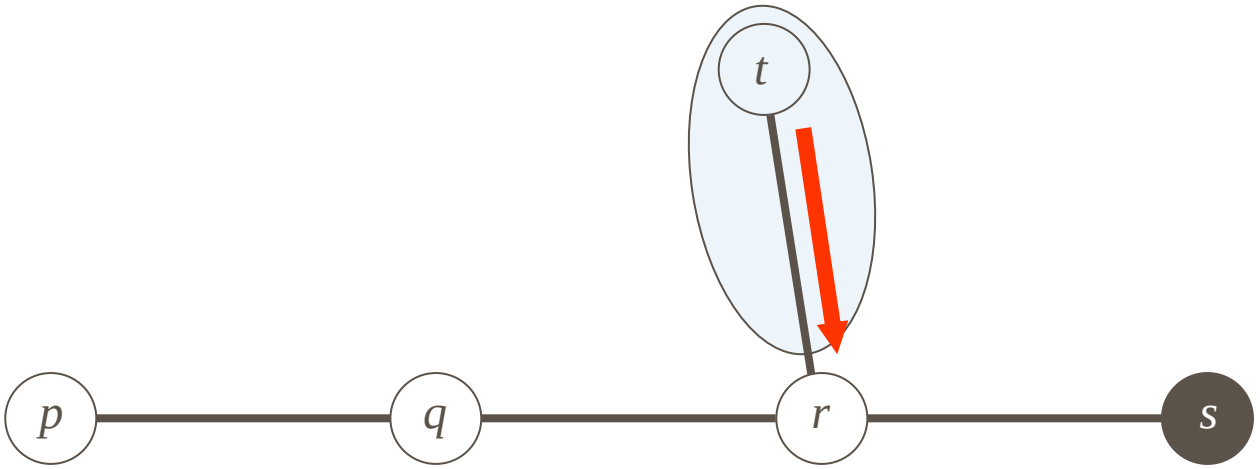
Inward Pass (Dynamic Programming)



$$M_{qr}(k) = \min_j \left\{ \left(\theta_q(j) + M_{pq}(j) \right) + \theta_{qr}(j,k) \right\}$$

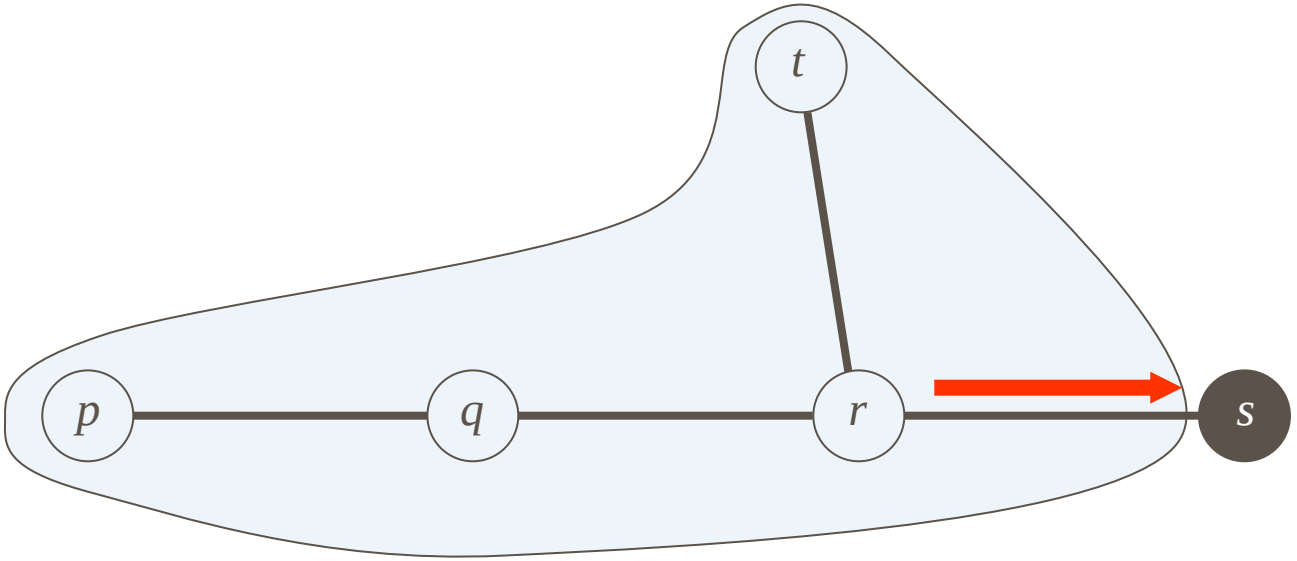
Slide credit: V. Kolmogorov

Inward Pass (Dynamic Programming)



Slide credit: V. Kolmogorov

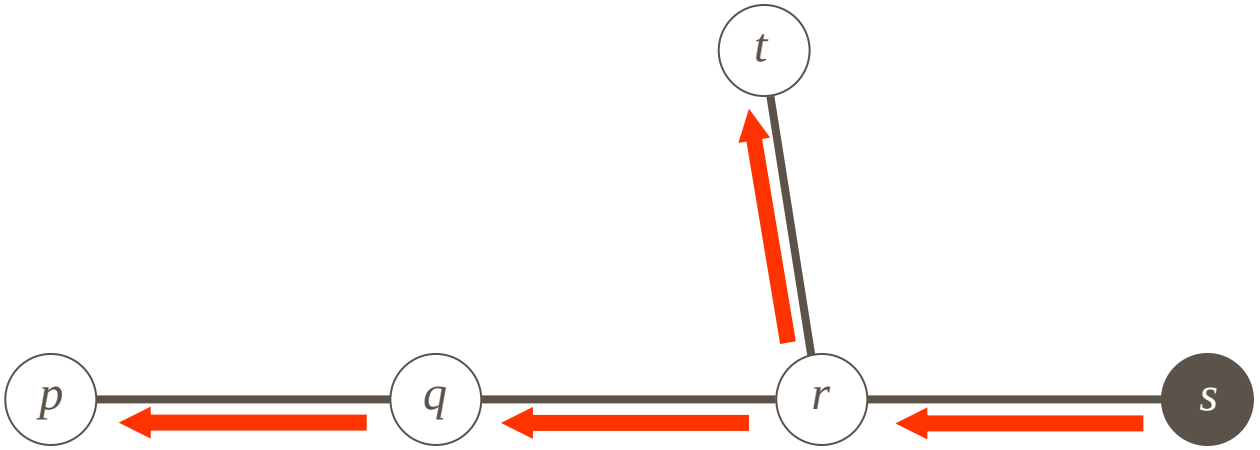
Inward Pass (Dynamic Programming)



$$M_{rs}(k) = \min_j [\theta_r(j) + M_{qr}(j) + M_{tr}(j) + \theta_{rs}(j,k)]$$

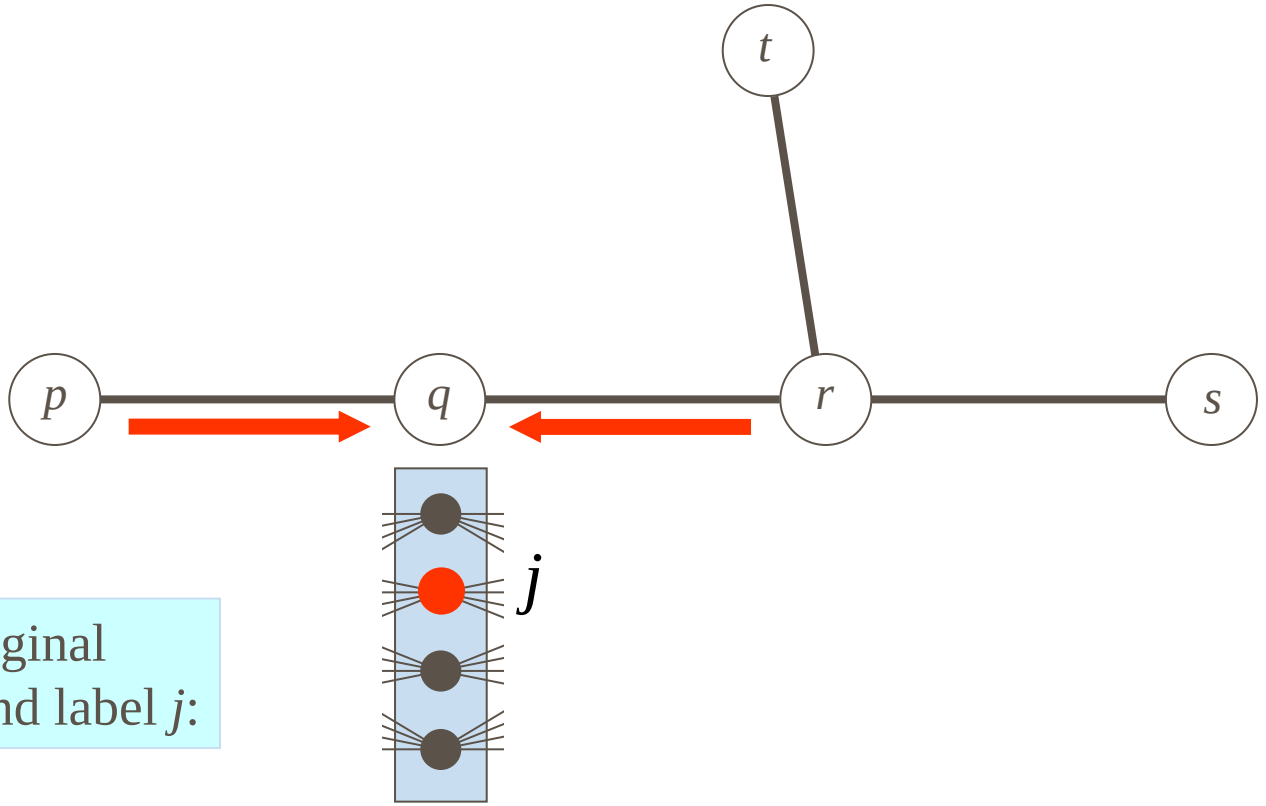
Slide credit: V. Kolmogorov

Outward Pass



Slide credit: V. Kolmogorov

BP on a Tree: Min-marginals



Min-marginal
for node q and label j:

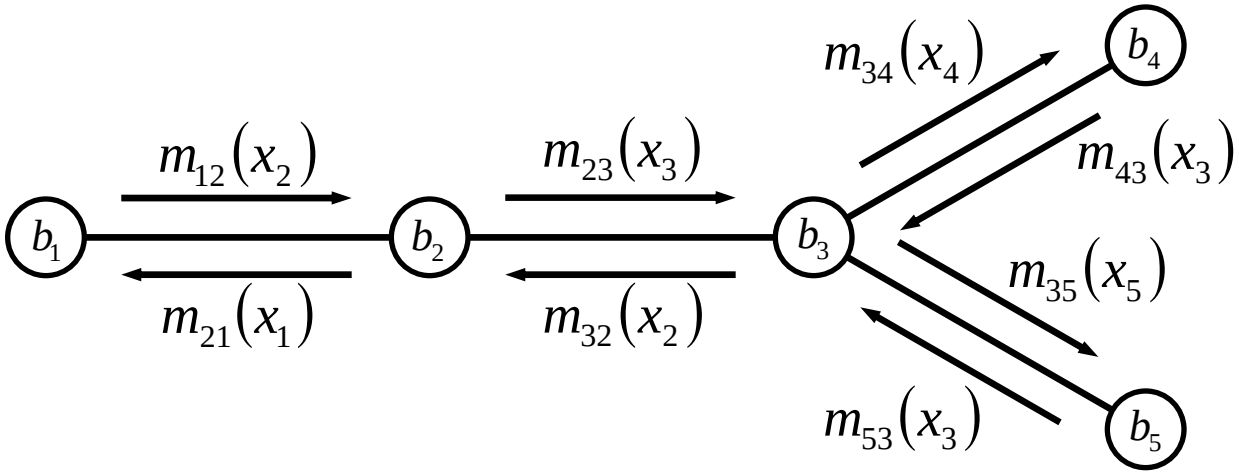
$$\min_{\mathbf{x}} \left\{ E(\mathbf{x}) \;\middle|\; x_q = j \right\} = \theta_q(j) + M_{pq}(j) + M_{rq}(j)$$

Slide credit: V. Kolmogorov

Other BP Versions

- Key property: min(a+b,a+c) = a+min(b,c)
- BP can be generalized to any operators satisfying the above property
- E.g., instead of (min,+), we could have:
  - (max,*)
    Resulting algorithm is called max-product.
    What does it compute?
    - The state with max posterior probability
  - (+,*)
    Resulting algorithm is called sum-product.
    What does it compute?
    - Marginal probabilities

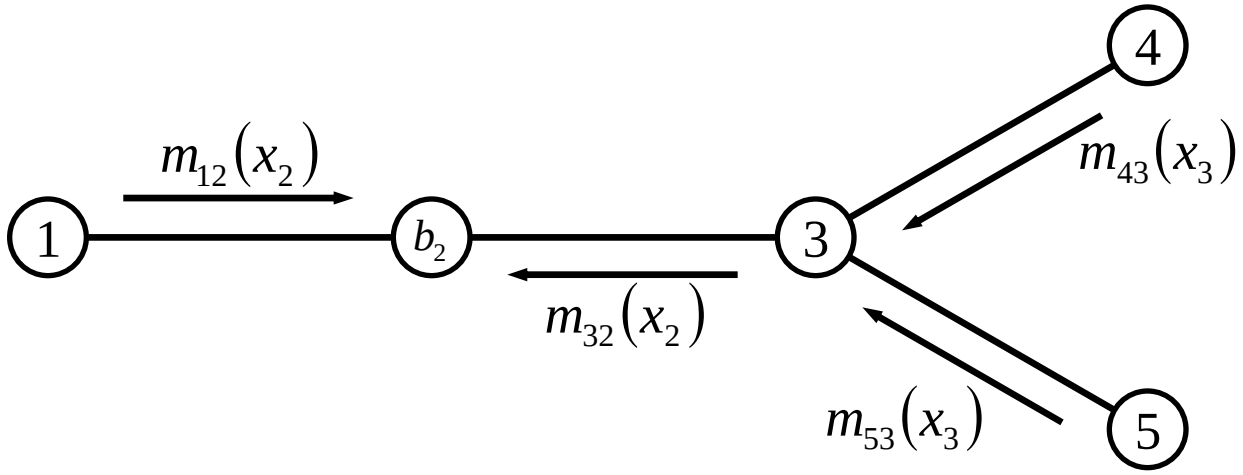
Belief Propagation



- Beliefs replace probabilities: $b_i(x_i) = \frac{1}{z_i}\phi_i(x_i, y_i)\prod_{j\in N(i)} m_{ji}(x_i)$
- Messages propagate information:

$$m_{ji}(x_i) = \sum_{x_j \in X_j} \phi_j(x_j, y_j)\psi_{ji}(x_j, x_i)\prod_{k\in N(j)\setminus i} m_{kj}(x_j)$$

Belief Propagation Example



$$b_2(x_2) = \frac{1}{z_2}\phi_2(x_2, y_2)\prod_{j\in N(2)} m_{j2}(x_2) = \frac{1}{z_2}\phi_2(x_2, y_2)m_{12}(x_2)m_{32}(x_2)$$

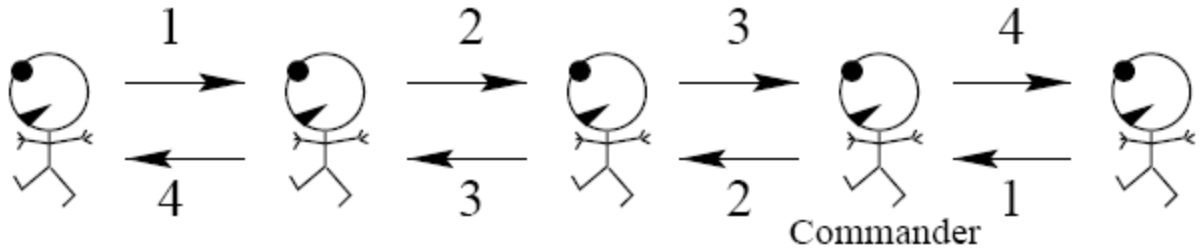
$$m_{12}(x_2) = \sum_{x_1\in X_1}\phi_1(x_1, y_1)\psi_{12}(x_1, x_2)$$

$$m_{32}(x_2) = \sum_{x_3\in X_3}\phi_3(x_3, y_3)\psi_{32}(x_3, x_2)m_{43}(x_3)m_{53}(x_3)$$

$$m_{43}(x_3) = \sum_{x_4\in X_4}\phi_4(x_4, y_4)\psi_{43}(x_4, x_3); m_{53}(x_3) = \sum_{x_5\in X_5}\phi_5(x_5, y_5)\psi_{53}(x_5, x_3)$$

BP = Distributive Algorithm

- BP works in a distributive manner
  - (as a result, it can be parallelized)
- Essentially BP is a decentralized algorithm
- Global results through local exchange of information
- Simple example: counting soldiers

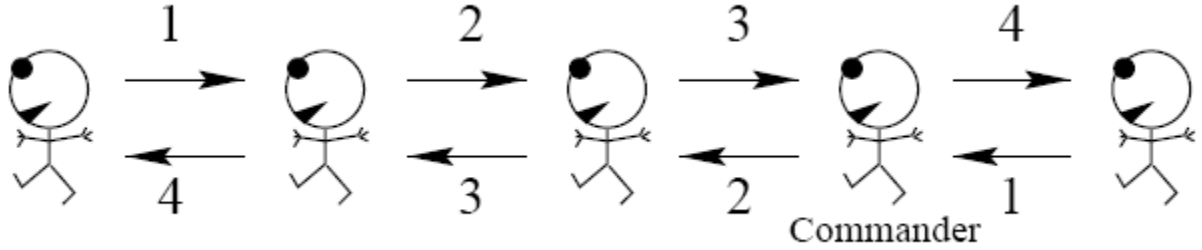


From David MacKay's book "Information Theory, Inference, and Learning"

Counting Soldiers

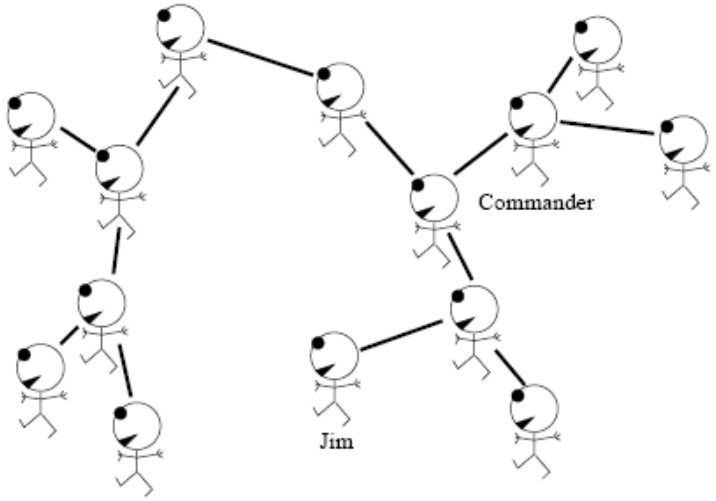
Each soldier follows these rules:

1. If you are the front soldier in the line, say the number `one' to the soldier behind you.
2. If you are the rearmost soldier in the line, say the number `one' to the soldier in front of you.
3. If a soldier ahead of or behind you says a number to you, add one to it, and say the new number to the soldier on the other side.



From David MacKay's book "Information Theory, Inference, and Learning"

Counting Soldiers



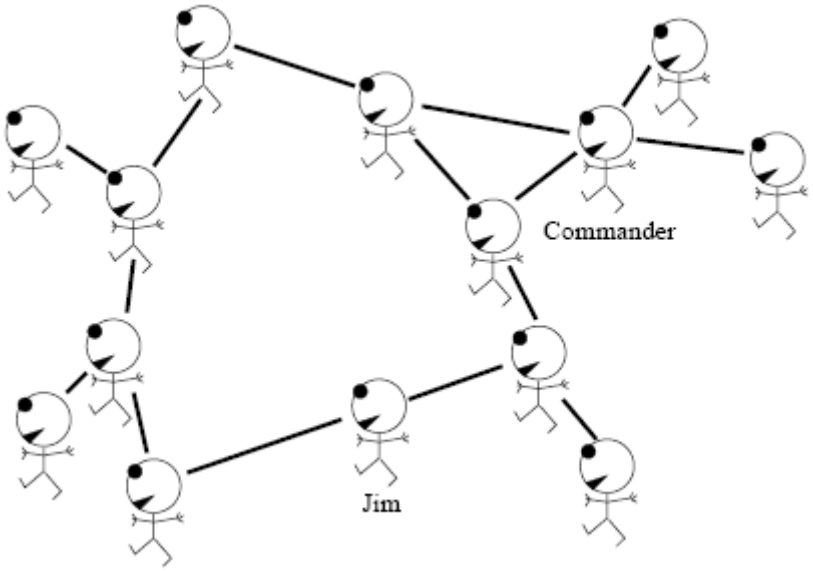
Can be extended to a tree:

1. Count your number of neighbors, N.
2. Keep count of the number of messages you have received from your neighbors, m, and of the values v_1, v_2, …, v_N of each of those messages. Let V be the running total of the messages you have received.
3. If the number of messages you have received, m, is N–1, identify the neighbor who has not sent you a message and tell him the number V+1.
4. If the number of messages you have received is equal to N, then:
   (a) the number V + 1 is the required total.
   (b) to each neighbor n say the number V+1–v_n.

From David MacKay's book "Information Theory, Inference, and Learning"

Counting Soldiers

- Localized approach does not work in graphs with loops



BP: Questions

- When can we calculate beliefs exactly?
- When do beliefs equal probabilities?
- When is belief propagation efficient?

Answer: Singly-Connected Graphs (SCG's)

- Graphs without loops
- Messages terminate at leaf nodes
- Beliefs equal probabilities
- Complexity in previous example reduced from $13S^5$ to $24S^2$

BP on Loopy Graphs

- Messages do not terminate
- Possible approximate solutions
  - Standard belief propagation
  - Generalized belief propagation

BP with Two Graphs: Goals

- Utilize advantages of SCG's
- Be accurate and efficient on loopy graphs

BP with Two Graphs: SCG's

- Consider loopy graph with *n* vertices
- Select two sets of SCG's that approximate the graph

$$\{G_1, \ldots, G_n\}$$

$$\{H_1, \ldots, H_n\}$$

- Calculate beliefs on each set of SCG's:
  - $b_i^G(x_i)$ and $b_i^H(x_i)$
- Select maximum beliefs from both sets
  - $b_i(x_i) = \max\left(b_i^G(x_i), b_i^H(x_i)\right)$

BP-TwoGraphs: Vision SCG's

- Rectangular grid of pixel vertices
- H_i: horizontal graphs
- G_i: vertical graphs

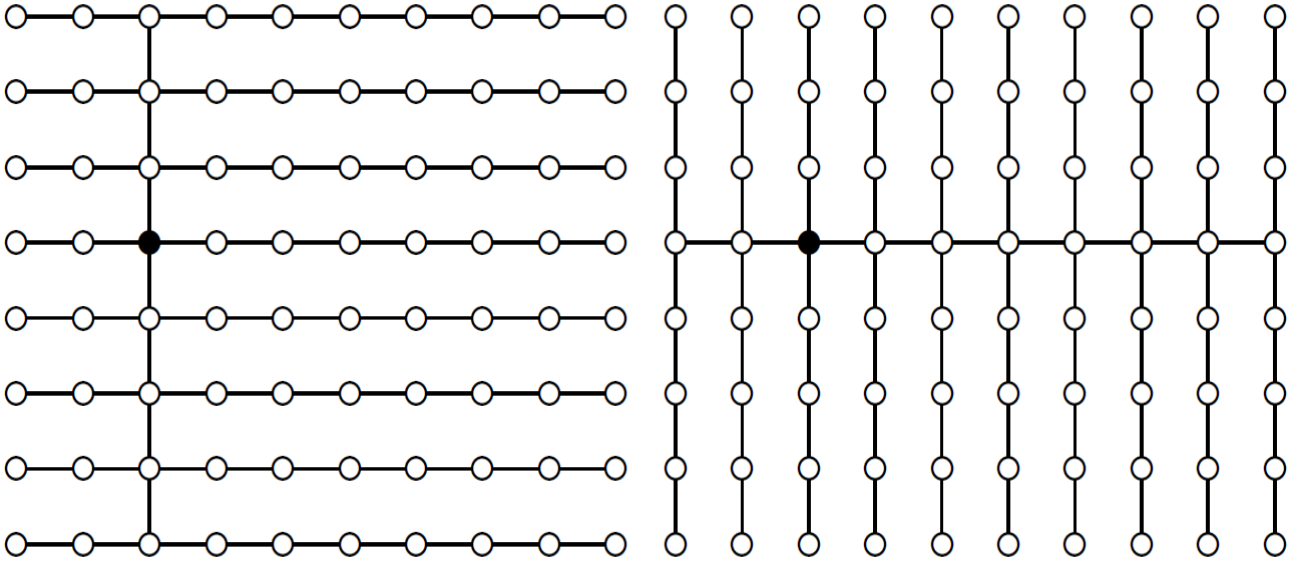


Image Segmentation

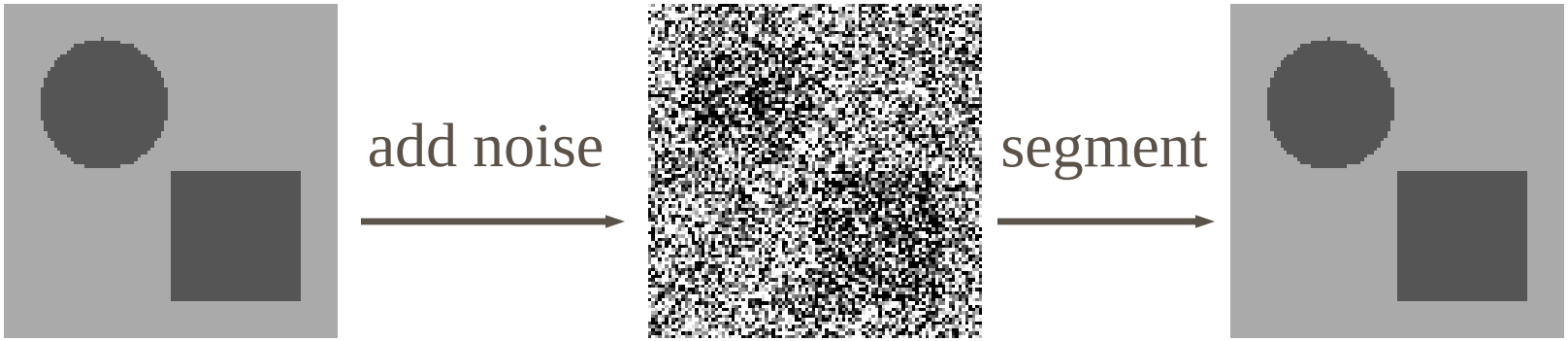
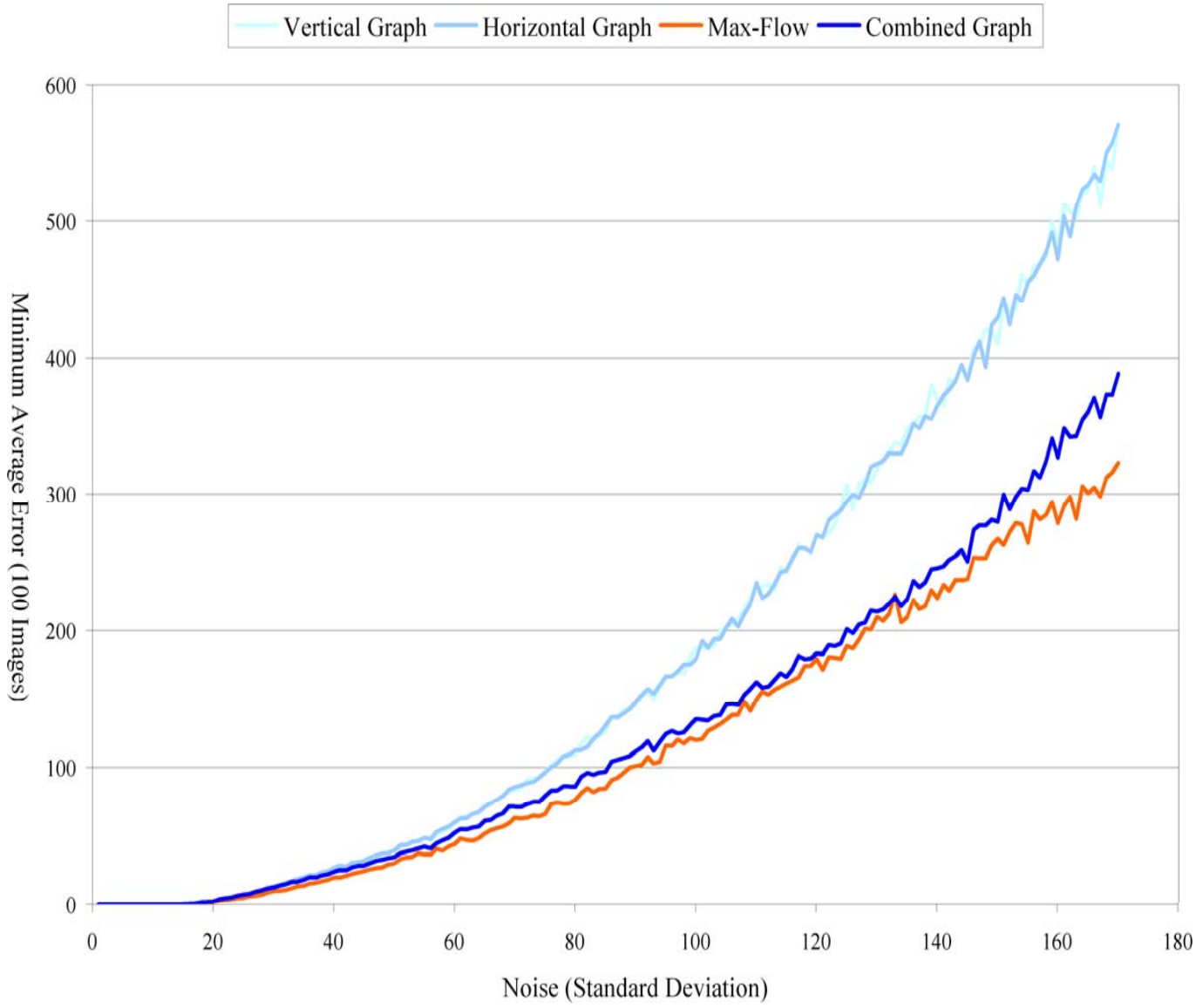
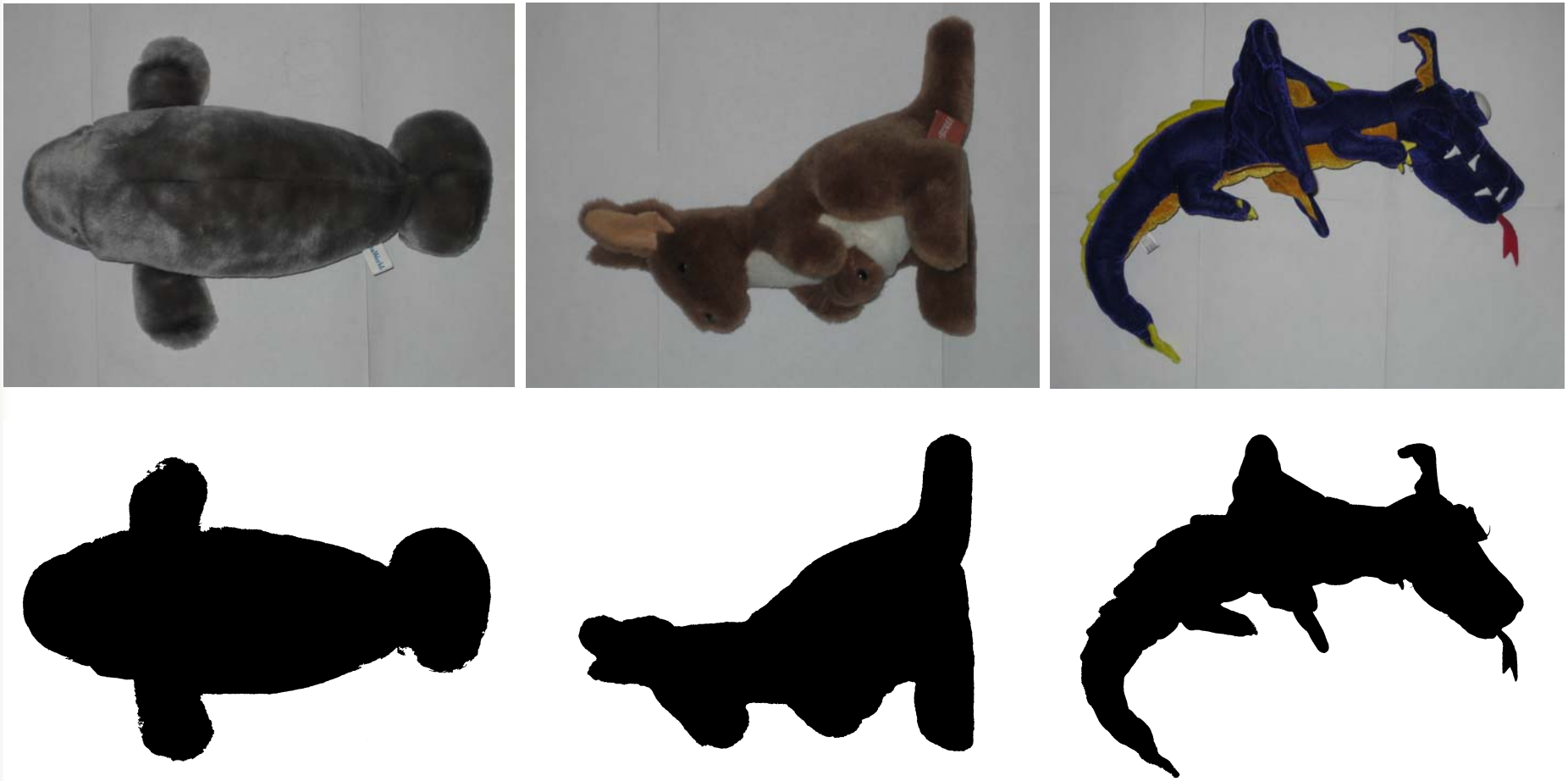


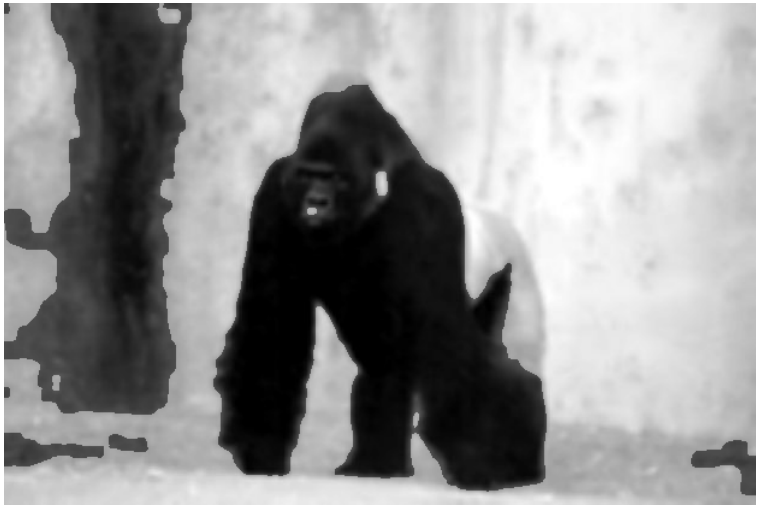
Image Segmentation: Results



Real Image Segmentation: Training



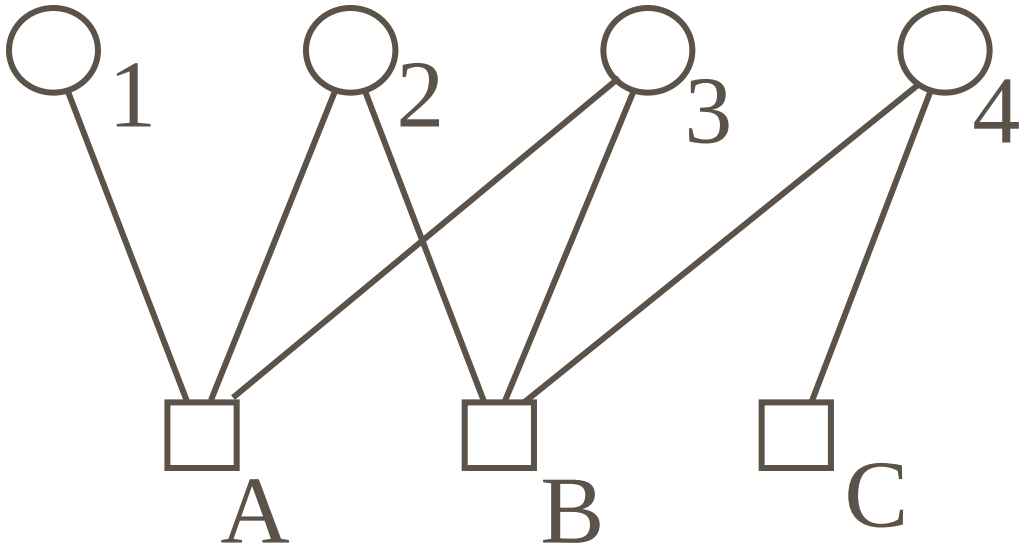
Real Image Segmentation: Results



Generalized BP: Factor Graphs

- Explicitly add nodes for each MRF clique c

$$P(\mathbf{x}) = P(x_1, ..., x_N) = \frac{1}{Z} \prod_c \psi_c(\mathbf{x}_c)$$



$$P(x_1, x_2, x_3, x_4) = \frac{1}{Z} \psi_A(x_1, x_2, x_3) \psi_B(x_2, x_3, x_4) \psi_C(x_4)$$

Computing Marginal Probabilities

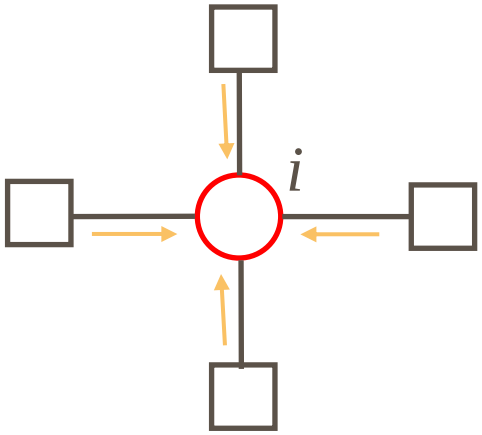
$$p_S(\mathbf{x}_S) = \sum_{\mathbf{x}_{-s}} p(\mathbf{x})$$

Fundamental for

- Decoding error-correcting codes
- Inference in Bayesian networks
- Computer Vision or Medical Imaging
- Statistical physics of magnets

Non-trivial because of the exponential number of terms in the sum.

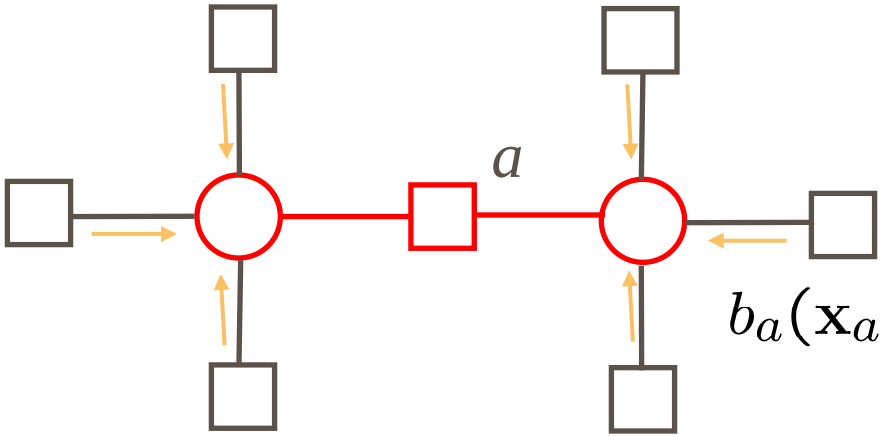
Standard Belief Propagation



$$b_i(x_i) \propto \prod_{a \in N(i)} m_{a \to i}(x_i)$$

"beliefs" "messages"

The "belief" is the BP approximation of the marginal probability.



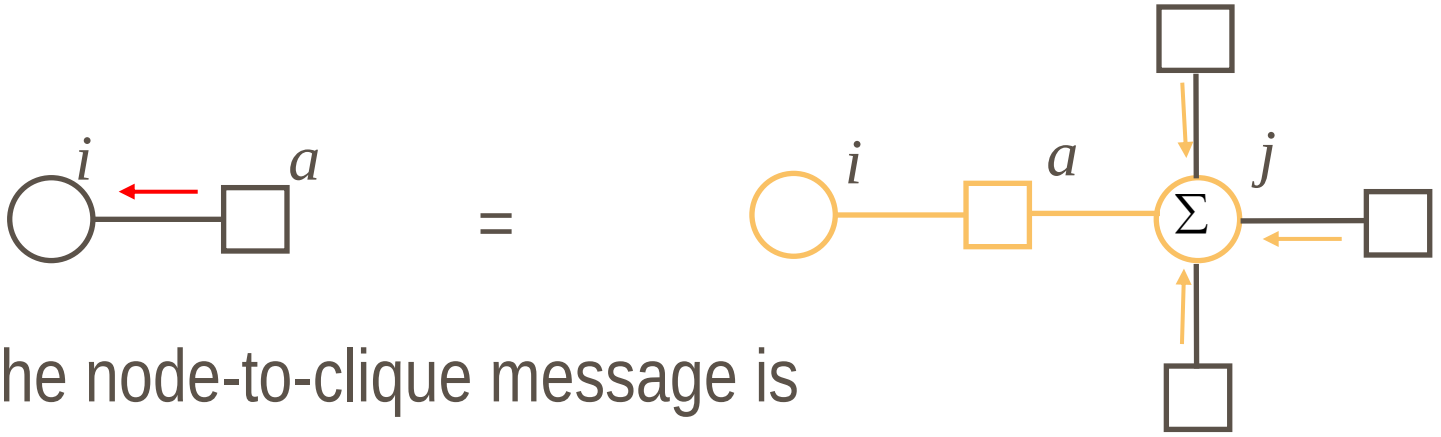
$$b_a(\mathbf{x}_a) \propto \psi_a(\mathbf{x}_a) \prod_{i \in N(a)} \prod_{b \in N(i) \setminus a} m_{b \to i}(x_i)$$

BP Message Update Rules

- Using $b_i(x_i) = \sum_{\mathbf{x}_{a-i}} b_a(\mathbf{x}_a)$

we get the clique to node message

$$m_{a\to i}(x_i) = \sum_{\mathbf{x}_{a-i}} \psi_a(\mathbf{x}_a) \prod_{j\in N(a)\setminus i} \prod_{b\in N(j)\setminus a} m_{b\to j}(x_j)$$



- The node-to-clique message is

$$m_{i\to a}(x_i) = \prod_{b\in N(i)\setminus a} m_{b\to i}(x_i)$$

Message Update

- Denoting the set of messages between all nodes as

$$\vec{M} = (m_{a \to i})_{a,i}$$

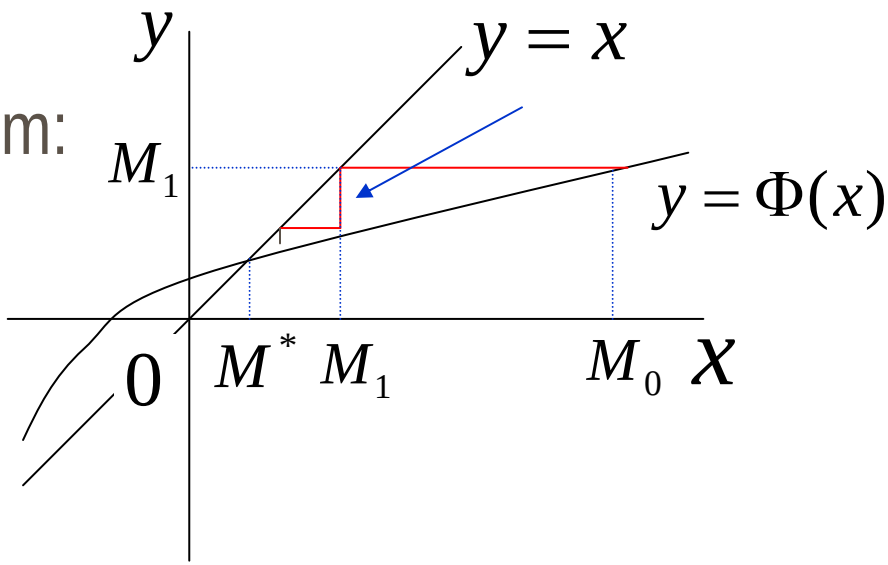
- The message update rules can be summarized as

$$\vec{M} = \Phi(\vec{M})$$

- This yields a fixed-point algorithm:
  - Initialize $\vec{M}_0$
  - Repeat

$$\vec{M}_{t+1} = \Phi(\vec{M}_t)$$

  - Until convergence



- Not guaranteed to converge, but it does in practice

Challenges in BP

- Message update:

$$m_{a\to i}(x_i) = \sum_{\mathbf{x}_{a-i}} \psi_a(\mathbf{x}_a) \prod_{j\in N(a)\setminus i} \prod_{b\in N(j)\setminus a} m_{b\to j}(x_j)$$

  - Need to sum over all other variables from x_a except x_i
  - Unfeasible for large cliques
  - BP usually used for pairwise cliques
- Memory demands
  - Need to memorize all messages
  - Each message is a probability
  - Can be huge for large graphs and many entries in the probability
- Convergence
  - Not always converges

Max-Product BP

- Finds (approximate) most probable configuration

$$\mathbf{x} = \arg\max P(\mathbf{x})$$

- Use the exponential form of the MRF

$$P(\mathbf{x}) = P(x_1, ..., x_N) = \frac{1}{Z} \exp(-\sum_c \phi_c(\mathbf{x}_c))$$

- Min-sum (aka Max-Product) BP:
  - Replace the sum by a min and the product by a sum :

$$m_{a \to i}(x_i) = \min_{\mathbf{x}_{a-i}} [\phi_a(\mathbf{x}_a) + \sum_{j \in N(a) \setminus i} m_{j \to a}(x_j)]$$

$$m_{i \to a}(x_i) = \sum_{b \in N(i) \setminus a} m_{b \to i}(x_i)$$

Some Facts About BP

- BP is exact on trees.
- If BP converges it reaches a local minimum of an objective function = the Bethe free energy (Yedidia et.al '00 , Heskes '02)
  - often good approximation of the global minimum
- When it converges, convergence is fast near the fixed point.
- Many exciting applications:
  - Error correcting decoding (MacKay, Yedidia, McEliece, Frey)
  - Computer Vision (Freeman, Weiss)
  - Bioinformatics (Weiss)
  - Constraint satisfaction problems (Dechter)
  - Game theory (Kearns)

References

- JS Yedidia, WT Freeman, Y Weiss. Understanding belief propagation and its generalizations.
- David MacKay. Information Theory, Inference, and Learning
  - http://www.inference.phy.cam.ac.uk/mackay/itila/
- http://www.eurasip.org/Proceedings/Eusipco/Eusipco2008/tutorials/tutorial_2%20komodakis_zabih_part_2.pdf
- http://www.csd.uoc.gr/~komod/ICCV07_tutorial/ICCV07_tutorial_vnk.ppt