Probabilistic Graphical Models

MAP inference

Siamak Ravanbakhsh

Fall 2019

Learning objectives

- MAP inference and its complexity
- exact & approximate MAP inference
  - max-product and max-sum message passing
  - relationship to LP relaxation
  - graph-cuts for MAP inference

Optimization

$$x^* = \arg\max_x f(x)$$

Optimization

$$x^* = \arg\max_x f(x)$$

$$\begin{aligned} g_c(x) &\geq 0 \quad \forall c \\ h_d(x) &= 0 \quad \forall d \end{aligned}$$

may or may not have constraints

continuous or **discrete** *(combinatorial)*...

Optimization

$$x^* = \arg\max_x f(x)$$

$$\begin{aligned} g_c(x) &\geq 0 \quad \forall c \\ h_d(x) &= 0 \quad \forall d \end{aligned}$$

may or may not have constraints

continuous or **discrete** *(combinatorial)*...

- local search heuristics
  - hill-climbing
  - beam search
  - tabu search
  - ..
- simulated annealing
- integer program
- genetic algorithm
- branch and bound: when you can efficiently upper-bound partial assignments

Optimization

$$x^* = \arg\max_x f(x)$$

$$g_c(x) \geq 0 \quad \forall c$$
$$h_d(x) = 0 \quad \forall d$$

may or may not have constraints

continuous or **discrete** *(combinatorial)*...

- local search heuristics
  - hill-climbing
  - beam search
  - tabu search
  - ..
- simulated annealing
- integer program
- genetic algorithm
- branch and bound: when you can efficiently upper-bound partial assignments

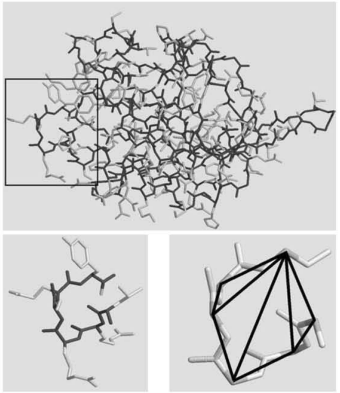
what if f(x) is structured? $f(x) = \sum_I f_I(x_I)$

MAP inference in a graphical model

Definition & complexity

MAP $\arg\max_x p(x)$

decision problem given Bayes-net, deciding whether $p(x) > c$ for some x is NP-complete!



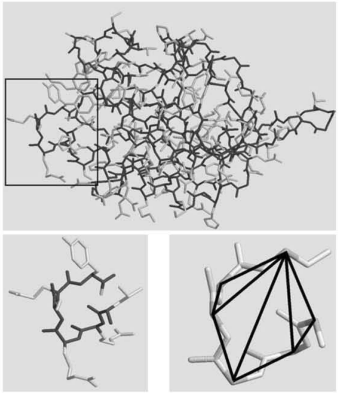
side-chain prediction as MAP inference

(Yanover & Weiss)

Definition & complexity

MAP $\arg\max_x p(x)$

decision problem given Bayes-net, deciding whether $p(x) > c$ for some x is NP-complete!



side-chain prediction as MAP inference

(Yanover & Weiss)

Marginal MAP $\arg\max_x \sum_y p(x, y)$

decision problem given Bayes-net for $p(x, y)$, deciding whether $p(x) > c$ for some x is complete for NP^{PP}

- is NP-hard even for trees

PP: a non-deterministic Turing machine that accepts if the majority of paths accept

NP: a non-deterministic Turing machine that accepts if a single path accepts (with access to a PP oracle)

Problem & terminology

MAP inference:

$$\arg\max_x p(x) = \arg\max_x \frac{1}{Z} \prod_I \phi_I(x_I)$$

$$\equiv \arg\max_x \tilde{p}(x) = \arg\max_x \prod_I \phi_I(x_I)$$

ignore the normalization constant

aka max-product inference

Problem & terminology

MAP inference:

$$\arg\max_x p(x) = \arg\max_x \frac{1}{Z} \prod_I \phi_I(x_I)$$

$$\equiv \arg\max_x \tilde{p}(x) = \arg\max_x \prod_I \phi_I(x_I)$$

ignore the normalization constant

aka max-product inference

with evidence:

$$\arg\max_x p(x \mid e) = \arg\max_x \frac{p(x,e)}{p(e)} \equiv \arg\max_x p(x,e)$$

Problem & terminology

MAP inference:

$$\arg\max_x p(x) = \arg\max_x \frac{1}{Z} \prod_I \phi_I(x_I)$$

$$\equiv \arg\max_x \tilde{p}(x) = \arg\max_x \prod_I \phi_I(x_I)$$

ignore the normalization constant

aka max-product inference

with evidence:

$$\arg\max_x p(x \mid e) = \arg\max_x \frac{p(x,e)}{p(e)} \equiv \arg\max_x p(x,e)$$

log domain:

$$\arg\max_x p(x) \equiv \arg\max_x \sum_I \ln \phi_I(x_I) \equiv \arg\min_x -\ln \tilde{p}(x)$$

aka max-sum inference

aka min-sum inference (energy minimization)

Max-marginals

marginal $\sum_{x \in Val(x)} \phi(x, y)$ used in sum-product inference

is replaced with *max-marginal* $\max_{x \in Val(x)} \phi(x, y)$

$$\phi(a, b, c)$$

a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	c^1	0.25
a^1	c^2	0.35
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.15
a^3	c^2	0.21

$$\phi'(a, c) = \max_b \phi(a, b, c)$$

distributive law for MAP inference

$\max(ab, ac) = a\max(b, c)$	max-product inference
$\max(a+b, a+c) = a + \max(b, c)$	max-sum inference
$\max(\min(a, b), \min(a, c)) = \max(a, \min(b, c))$	min-max inference
$ab + ac = a(b + c)$	sum-product inference

3 operations ⟷ 2 operations

distributive law for MAP inference

$$\max(ab, ac) = a\max(b, c)$$ max-product inference

$$\max(a+b, a+c) = a + \max(b, c)$$ max-sum inference

$$\max(\min(a,b), \min(a,c)) = \max(a, \min(b,c))$$ min-max inference

$$ab + ac = a(b+c)$$ sum-product inference

3 operations — 2 operations

save computation by factoring the operations

in disguise $\max_{x,y} f(x,y)g(y,z) = \max_y g(y,z) \max_x f(x,y)$

- assuming $|Val(X)| = |Val(Y)| = |Val(Z)| = d$
- complexity: from $\mathcal{O}(d^3)$ to $\mathcal{O}(d^2)$

Max-product variable elimination

- the procedure is similar to VE for sum-product inference
- eliminate **all** the variables

- input: $\Phi^{t=0} = \{\phi_1, \ldots, \phi_K\}$ a set of factors *(e.g. CPDs)*
- output: $\max_x \tilde{p}(x) = \max_x \prod_I \phi_I(x_I)$
- go over $x_{i_1}, \ldots, x_{i_n}$ in some order:
  - collect all the relevant factors: $\Psi^t = \{\phi \in \Phi^t \mid x_{i_t} \in Scope[\phi]\}$
  - calculate their product: $\psi_t = \prod_{\phi \in \Psi^t} \phi$
  - max-marginalize out x_{i_t}: $\psi'_t = \max_{x_{i_t}} \psi_t$
  - update the set of factors: $\Phi^t = \Phi^{t-1} - \Psi^t + \{\psi'_t\}$
- return the scalar in $\Phi^{t=m}$ as $\max_x \tilde{p}(x)$

maximizing value

similar to the partition function: $Z = \sum_x \tilde{p}(x)$

Decoding the max-value

we need to recover the maximizing assignment x^*

keep $\{\psi_{t=1}, \ldots, \psi_{t=n}\}$, produced during inference

- input: $\Phi^{t=0} = \{\phi_1, \ldots, \phi_K\}$ a set of factors *(e.g. CPDs)*
- output: $\max_x \tilde{p}(x) = \max_x \prod_I \phi_I(x_I)$
- go over $x_{i_1}, \ldots, x_{i_n}$ in some order:
  - collect all the relevant factors: $\Psi^t = \{\phi \in \Phi^t \mid x_{i_t} \in Scope[\phi]\}$
  - calculate their product: $\psi_t = \prod_{\phi \in \Psi^t} \phi$
  - max-marginalize out x_{i_t}: $\psi'_t = \max_{x_{i_t}} \psi_t$
  - update the set of factors: $\Phi^t = \Phi^{t-1} - \Psi^t + \{\psi'_t\}$
- return the scalar in $\Phi^{t=m}$ as $\max_x \tilde{p}(x)$

Decoding the max-value

start from the last eliminated variable

$\psi_{t=n}$ should have been a function of x_{i_n} alone: ${x_{i_n}}^* \leftarrow \arg\max \psi_n$

- input: $\Phi^{t=0} = \{\phi_1, \ldots, \phi_K\}$ a set of factors *(e.g. CPDs)*
- output: $\max_x \tilde{p}(x) = \max_x \prod_I \phi_I(x_I)$
- go over $x_{i_1}, \ldots, x_{i_n}$ in some order:
  - collect all the relevant factors: $\Psi^t = \{\phi \in \Phi^t \mid x_{i_t} \in Scope[\phi]\}$
  - calculate their product: $\psi_t = \prod_{\phi \in \Psi^t} \phi$
  - max-marginalize out x_{i_t}: $\psi'_t = \max_{x_{i_t}} \psi_t$
  - update the set of factors: $\Phi^t = \Phi^{t-1} - \Psi^t + \{\psi'_t\}$
- return the scalar in $\Phi^{t=m}$ as $\max_x \tilde{p}(x)$

Decoding the max-value

start from the last eliminated variable

at this point we have $x^*_{i_n}$

$\psi_{t=n-1}$ can only have $x_{i_{n-1}}, x_{i_n}$ in its domain $x_{i_{n-1}}{}^* \leftarrow \arg\max_{x_{i_{n-1}}} \psi_{n-1}(x_{i_{n-1}}, x_{i^*_n})$

- input: $\Phi^{t=0} = \{\phi_1, \ldots, \phi_K\}$ a set of factors *(e.g. CPDs)*
- output: $\max_x \tilde{p}(x) = \max_x \prod_I \phi_I(x_I)$
- go over $x_{i_1}, \ldots, x_{i_n}$ in some order:
  - collect all the relevant factors: $\Psi^t = \{\phi \in \Phi^t \mid x_{i_t} \in Scope[\phi]\}$
  - calculate their product: $\psi_t = \prod_{\phi \in \Psi^t} \phi$
  - max-marginalize out x_{i_t}: $\psi'_t = \max_{x_{i_t}} \psi_t$
  - update the set of factors: $\Phi^t = \Phi^{t-1} - \Psi^t + \{\psi'_t\}$
- return the product of scalars in $\Phi^{t=m}$ as $\max_x \tilde{p}(x)$

and so on...

Marginal-MAP variable elimination

- the procedure remains similar for $\max_{y_1,...,y_m} \sum_{x_1,...,x_n} \prod_I \phi_I(x_I)$
- max and sum do not commute

$$\max_x \sum_y \phi(x, y) \neq \sum_y \max_x \phi(x, y)$$

Marginal-MAP variable elimination

- the procedure remains similar for $\max_{y_1,...,y_m} \sum_{x_1,...,x_n} \prod_I \phi_I(x_I)$
- max and sum do not commute

  $\max_x \sum_y \phi(x,y) \neq \sum_y \max_x \phi(x,y)$

- *cannot* use arbitrary elimination order

Marginal-MAP variable elimination

- the procedure remains similar for $\max_{y_1,\ldots,y_m} \sum_{x_1,\ldots,x_n} \prod_I \phi_I(x_I)$
- max and sum do not commute

$\max_x \sum_y \phi(x,y) \neq \sum_y \max_x \phi(x,y)$

- *cannot* use arbitrary elimination order
- first, eliminate $\{x_1, \ldots, x_n\}$ (sum-prod VE)

Marginal-MAP variable elimination

- the procedure remains similar for $\max_{y_1,...,y_m} \sum_{x_1,...,x_n} \prod_I \phi_I(x_I)$
- max and sum do not commute

  $\max_x \sum_y \phi(x,y) \neq \sum_y \max_x \phi(x,y)$

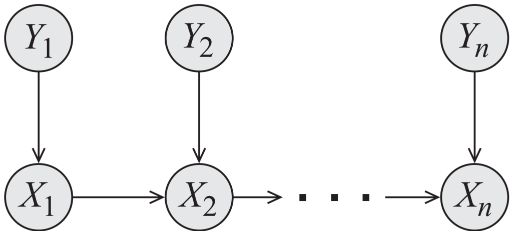
- *cannot* use arbitrary elimination order
- first, eliminate $\{x_1, \ldots, x_n\}$ (sum-prod VE)
- then eliminate $\{y_1, \ldots, y_m\}$ (max-prod VE)
  - decode the maximizing value

Marginal-MAP variable elimination

- the procedure remains similar for $\max_{y_1,...,y_m} \sum_{x_1,...,x_n} \prod_I \phi_I(x_I)$
- max and sum do not commute

$\max_x \sum_y \phi(x,y) \neq \sum_y \max_x \phi(x,y)$

- *cannot* use arbitrary elimination order
- first, eliminate $\{x_1, \ldots, x_n\}$ (sum-prod VE)
- then eliminate $\{y_1, \ldots, y_m\}$ (max-prod VE)
  - decode the maximizing value



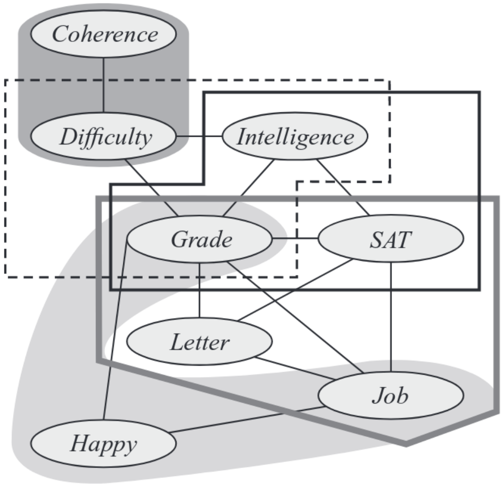
example: exponential complexity despite low tree-width

Max-product BP

In clique-trees, cluster-graphs, factor-graph

building the chordal graph
building the clique-tree
tree-width (complexity of inference)
...

remains the **same**!



Max-product BP

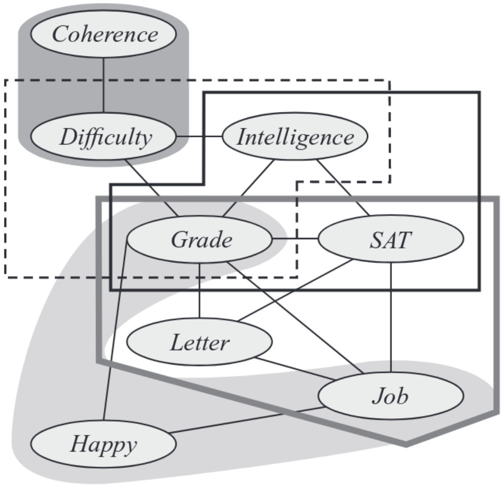
In clique-trees, cluster-graphs, factor-graph

- building the chordal graph
- building the clique-tree
- tree-width (complexity of inference)
- ...

remains the **same**!

main **differences**:

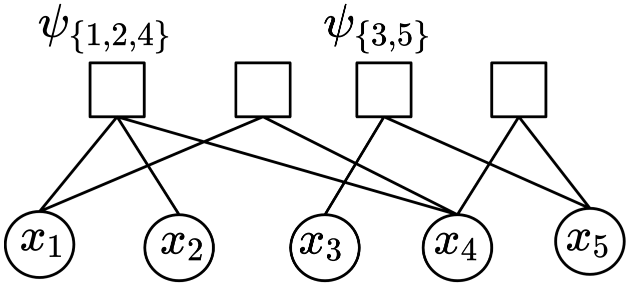
- replacing sum with max
- decoding the maximizing assignment
- variational interpretation



Max-product BP

Example **factor-graph**

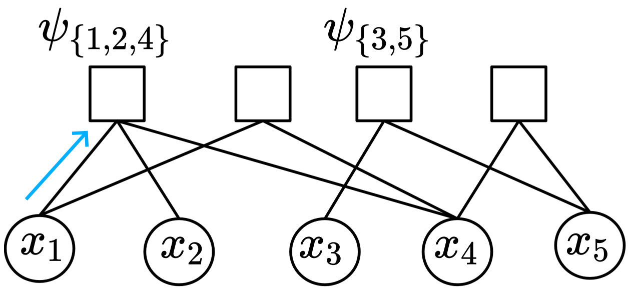
$$p(\mathbf{x}) = \frac{1}{Z} \prod_I \psi_I(x_I)$$



Max-product BP

Example **factor-graph**

$$p(\mathbf{x}) = \frac{1}{Z}\prod_I \psi_I(x_I)$$

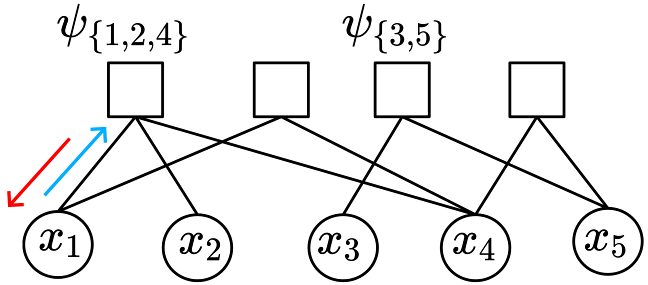


variable-to-factor message: $\delta_{i\to I}(x_i) \propto \prod_{J|i\in J, J\neq I} \delta_{J\to i}(x_i)$

Max-product BP

Example **factor-graph**

$$p(\mathbf{x}) = \frac{1}{Z} \prod_I \psi_I(x_I)$$



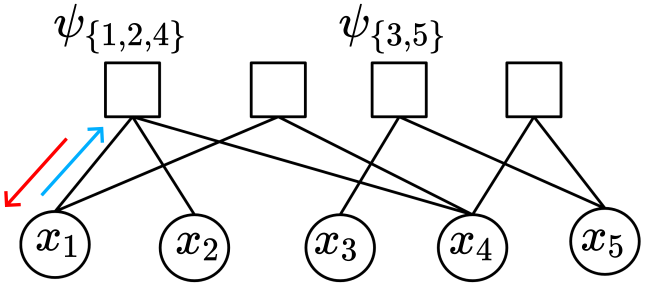
variable-to-factor message: $\delta_{i \to I}(x_i) \propto \prod_{J | i \in J, J \neq I} \delta_{J \to i}(x_i)$

factor-to-variable message: $\delta_{I \to i}(x_i) \propto \max_{x_{I-i}} \psi_I(x_I) \prod_{j \in I-i} \delta_{j \to I}(x_i)$

Max-product BP

Example **factor-graph**

$$p(\mathbf{x}) = \frac{1}{Z} \prod_I \psi_I(x_I)$$



variable-to-factor message: $\delta_{i \to I}(x_i) \propto \prod_{J|i \in J, J \neq I} \delta_{J \to i}(x_i)$

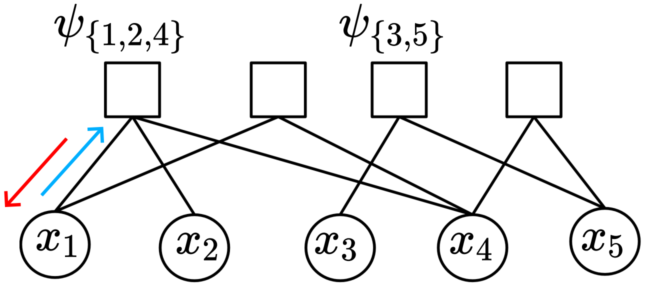
factor-to-variable message: $\delta_{I \to i}(x_i) \propto \max_{x_{I-i}} \psi_I(x_I) \prod_{j \in I-i} \delta_{j \to I}(x_i)$

approx. max-marginals: $\beta(x_i) \propto \prod_{J|i \in J} \delta_{J \to i}(x_i)$

Max-product BP

Example **factor-graph**

$$p(\mathbf{x}) = \frac{1}{Z}\prod_I \psi_I(x_I)$$



variable-to-factor message: $\delta_{i\to I}(x_i) \propto \prod_{J|i\in J, J\neq I} \delta_{J\to i}(x_i)$

factor-to-variable message: $\delta_{I\to i}(x_i) \propto \max_{x_{I-i}} \psi_I(x_I)\prod_{j\in I-i} \delta_{j\to I}(x_i)$

approx. max-marginals: $\beta(x_i) \propto \prod_{J|i\in J} \delta_{J\to i}(x_i)$

use damping for convergence in loopy graphs

Decoding exact max-marginals

clique-trees &factor-graphs without any loops

Single MAP assignment

MAP assignment is unique $\Leftrightarrow$ max-marginals are unambiguous

$$x^* = \arg\max_x p(x) \qquad \Leftrightarrow \qquad x_i^* = \arg\max_{x_i} \beta(x_i)$$

Decoding exact max-marginals

clique-trees &factor-graphs without any loops

Single MAP assignment

MAP assignment is unique $\Leftrightarrow$ max-marginals are unambiguous

$x^* = \arg\max_x p(x)$

$x_i^* = \arg\max_{x_i} \beta(x_i)$

Multiple *MAP assignments* $\Rightarrow$

example

$p(x_1, x_2) = \frac{1}{2}\mathbb{I}(x_1 = x_2)$

$\beta(x_1 = 0) = \beta(x_1 = 1)$

$\beta(x_2 = 0) = \beta(x_2 = 1)$

Decoding exact max-marginals

clique-trees &factor-graphs without any loops

Single MAP assignment

MAP assignment is unique $\iff$ max-marginals are unambiguous

$$x^* = \arg\max_x p(x) \qquad x_i^* = \arg\max_{x_i} \beta(x_i)$$

Multiple *MAP assignments* $\implies$ a join assignment x^* exists that is **locally optimal**

example

$$p(x_1, x_2) = \tfrac{1}{2}\mathbb{I}(x_1 = x_2)$$

$$\beta(x_1 = 0) = \beta(x_1 = 1)$$

$$\beta(x_2 = 0) = \beta(x_2 = 1)$$

$$\beta(x_i^*) = \max_{x_i} \beta(x_i) \forall i$$

$$\beta(x_I^*) = \max_{x_I} \beta(x_I) \forall I$$

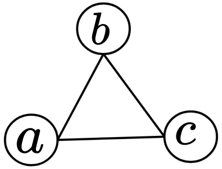
easy to find (how?)

Decoding pseudo max-marginals

cluster-graphs, loopy factor-graphs

best local assignments may be incompatible

example



	b=0	b=1
a=0	1	2
a=1	2	1

$\beta(a, b)$

	b=0	b=1
c=0	1	2
c=1	2	1

$\beta(b, c)$

	a=0	a=1
c=0	1	2
c=1	2	1

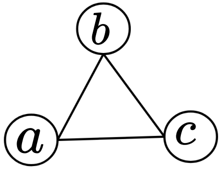
$\beta(a, c)$

Decoding **pseudo** max-marginals

cluster-graphs, loopy factor-graphs

best local assignments may be incompatible

example



	b=0	b=1
a=0	1	2
a=1	2	1

$\beta(a,b)$

	b=0	b=1
c=0	1	2
c=1	2	1

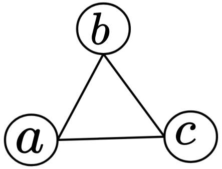
$\beta(b,c)$

	a=0	a=1
c=0	1	2
c=1	2	1

$\beta(a,c)$

... or compatible

example



	b=0	b=1
a=0	3	2
a=1	2	3

$\beta(a,b)$

	b=0	b=1
c=0	3	2
c=1	2	3

$\beta(b,c)$

	a=0	a=1
c=0	3	2
c=1	2	3

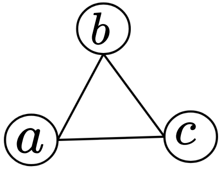
$\beta(a,c)$

Decoding **pseudo** max-marginals

cluster-graphs, loopy factor-graphs

best local assignments may be incompatible

example



	b=0	b=1
a=0	1	2
a=1	2	1

$\beta(a, b)$

	b=0	b=1
c=0	1	2
c=1	2	1

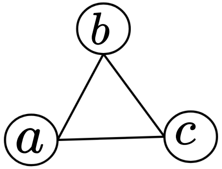
$\beta(b, c)$

	a=0	a=1
c=0	1	2
c=1	2	1

$\beta(a, c)$

... or compatible

example



	b=0	b=1
a=0	3	2
a=1	2	3

$\beta(a, b)$

	b=0	b=1
c=0	3	2
c=1	2	3

$\beta(b, c)$

	a=0	a=1
c=0	3	2
c=1	2	3

$\beta(a, c)$

If $m(a), m(b), m(c)$ have unique max., a unique locally optimal belief exists

Decoding **pseudo** max-marginals

cluster-graphs, loopy factor-graphs

given a set of cluster max-marginals $\{m_I(x_I)\}_I$ how to find **locally optimal** $\hat{x}^*$ (optimal in all m_I) if it exists

- reduce to a constraint satisfaction problem
- use **decimation**:
  - run inference
  - fix a subset of variables $\hat{x}^*_I = \arg\max_{x_I} m_I(x_I)$
  - repeat until all vars are fixed

Optimality of max-product **loopy** BP

a **locally optimal** assignment $\hat{x}^*$ is a **strong local maxima** of $p(x)$

$m(\hat{x}_i^*) = \max_{x_i} m(x_i) \forall i$

$m(\hat{x}_I^*) = \max_{x_I} m(x_I) \forall I$

Optimality of max-product **loopy** BP

a **locally optimal** assignment $\hat{x}^*$ is a **strong local maxima** of $p(x)$

$m(\hat{x}_i^*) = \max_{x_i} m(x_i) \forall i$

$m(\hat{x}_I^*) = \max_{x_I} m(x_I) \forall I$

↓

no better assignment exists in a **large neighborhood** of $\hat{x}^*$

Optimality of max-product loopy BP

a **locally optimal** assignment $\hat{x}^*$ is a **strong local maxima** of $p(x)$

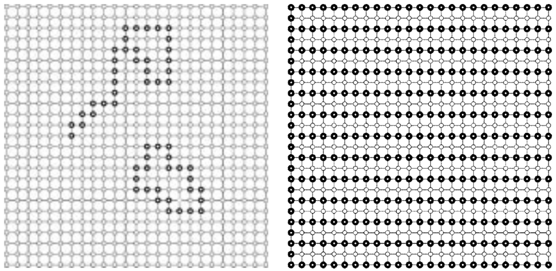
$m(\hat{x}_i^*) = \max_{x_i} m(x_i) \forall i$

$m(\hat{x}_I^*) = \max_{x_I} m(x_I) \forall I$

no better assignment exists in a **large neighborhood** of $\hat{x}^*$

- pick any subset of variables $T \subseteq \{1, \dots, n\}$
- build a subgraph $\mathcal{G}_T$ with all factors that have a variable in T
- if this subgraph does not have more than one loop then
- $p(\hat{x}^*)$ cannot be improved by changing the vars in T

example



from: Weiss & Freeman

Using integer and linear programming

pairwise case

$$\ln \tilde{p}(x) = \sum_{i,j} \ln \phi_{i,j}(x_i, x_j)$$

looking for an assignment x^* to maximize this sum

Using integer and linear programming

pairwise case

$$\ln \tilde{p}(x) = \sum_{i,j} \ln \phi_{i,j}(x_i, x_j)$$

looking for an assignment x^* to maximize this sum

integer-programming formulation:

$$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

$$q_{i,j}(x_i, x_j) \in \{0, 1\} \quad \forall i, j \in \mathcal{E}, x_i, x_j$$

picks a single assignment for vars in each factor

Using integer and linear programming

pairwise case

$$\ln \tilde{p}(x) = \sum_{i,j} \ln \phi_{i,j}(x_i, x_j)$$

looking for an assignment x^* to maximize this sum

integer-programming formulation:

$$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

$$q_{i,j}(x_i, x_j) \in \{0, 1\} \quad \forall i, j \in \mathcal{E}, x_i, x_j$$

picks a single assignment for vars in each factor

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$$

$$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$$

ensure that assignments to different factors are consistent

Using integer and linear programming

pairwise case

$$\ln \tilde{p}(x) = \sum_{i,j} \ln \phi_{i,j}(x_i, x_j)$$

looking for an assignment x^* to maximize this sum

integer-programming formulation:

$$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

$$q_{i,j}(x_i, x_j) \in \{0, 1\} \quad \forall i, j \in \mathcal{E}, x_i, x_j$$

picks a single assignment for vars in each factor

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$$

$$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$$

ensure that assignments to different factors are consistent

solution to this NP-hard program is the MAP assignment

Using integer and linear programming

pairwise case

linear programming has a polynomial-time solution

$$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

$$q_{i,j}(x_i, x_j) \in \{0, 1\}$$

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$$

$$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$$

ensure that assignments to different factors are consistent

Using integer and linear programming

pairwise case

linear programming has a polynomial-time solution

$$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

$q_{i,j}(x_i, x_j) \in \{0, 1\}$ relax this constraint to $q_{i,j}(x_i, x_j) \geq 0 \quad \forall i, j \in \mathcal{E}, x_i, x_j$

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$$

$$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$$

ensure that assignments to different factors are consistent

Using integer and linear programming

pairwise case

linear programming has a polynomial-time solution

$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$

$q_{i,j}(x_i, x_j) \in \{0, 1\}$ relax this constraint to $q_{i,j}(x_i, x_j) \geq 0 \quad \forall i, j \in \mathcal{E}, x_i, x_j$

$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$

$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$

ensure that assignments to different factors are consistent

local consistency constraints that we saw earlier

- outer-bound to marginal polytope for globally consistent $\{q_{i,j}\}$

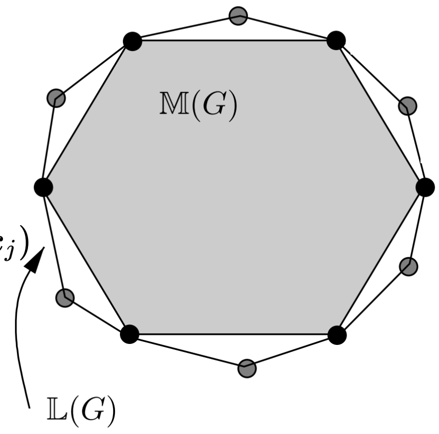
Using integer and linear programming

pairwise case

$\mathbb{M}$arginal polytope

$[q_{i,j}(x_i, x_j)]_{i,j \in \mathcal{E}, x_i, x_j}$

$\exists q(x) s.t. \max_{x_{-i,j}} q(x) = q_{i,j}(x_i, x_j)$



alternative form

the convex hull of sufficient statistics for all assignments to x

$$conv\{[\mathbb{I}[X_i = x_i, X_j = x_j]]_{i,j \in \mathcal{E}, x_i, x_j} \mid X\}$$

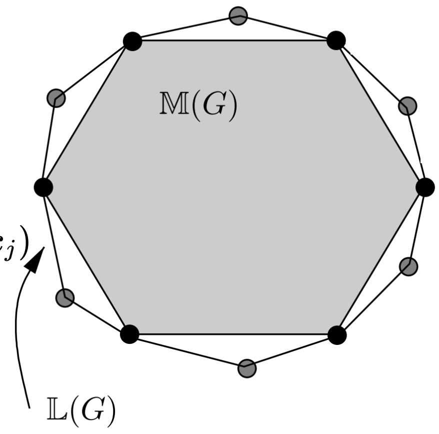
Using integer and linear programming

pairwise case

$\mathbb{M}$arginal polytope

$[q_{i,j}(x_i, x_j)]_{i,j \in \mathcal{E}, x_i, x_j}$

$\exists q(x) s.t. \max_{x_{-i,j}} q(x) = q_{i,j}(x_i, x_j)$



$\mathbb{L}$ocal consistency polytope

$[q_{i,j}(x_i, x_j)]_{i,j \in \mathcal{E}, x_i, x_j}$

$q_{i,j}(x_i, x_j) \geq 0 \quad \forall i, j \in \mathcal{E}, x_i, x_j$

$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$

$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$

alternative form

the convex hull of sufficient statistics for all assignments to x

$conv\{[\mathbb{I}[X_i = x_i, X_j = x_j]]_{i,j \in \mathcal{E}, x_i, x_j} \mid X\}$

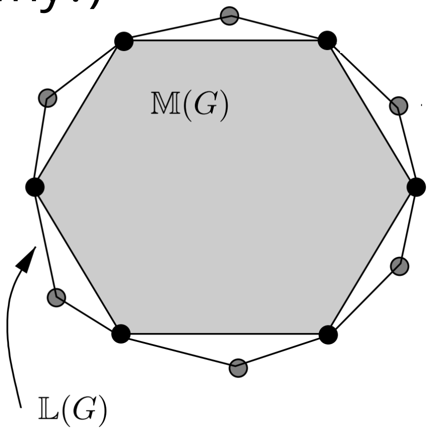
Using integer and linear programming

why is this important?

LP solutions are at corners of the polytope (why?)

LP using $\mathbb{L}$ is an upper-bound

to the MAP value using $\mathbb{M}$



Using integer and linear programming

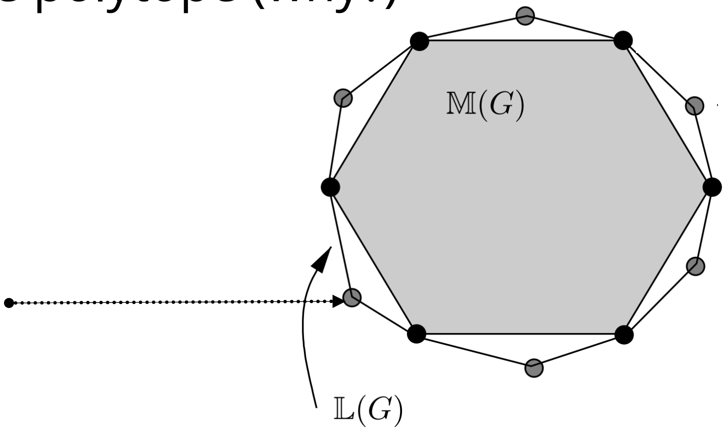
why is this important?

LP solutions are at corners of the polytope (why?)

LP using $\mathbb{L}$ is an upper-bound

to the MAP value using $\mathbb{M}$

LP solution found using $\mathbb{L}$



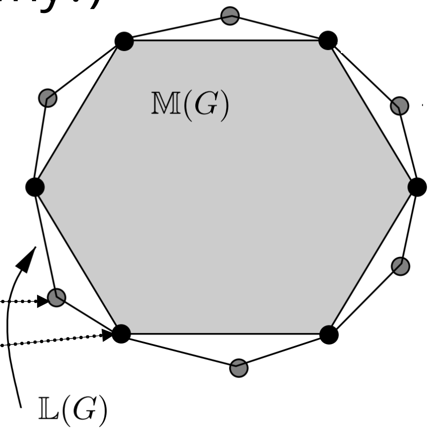
Using integer and linear programming

why is this important?

LP solutions are at corners of the polytope (why?)

LP using $\mathbb{L}$ is an upper-bound

to the MAP value using $\mathbb{M}$



LP solution found using $\mathbb{L}$

LP solution found using $\mathbb{M}$

- is integral (by definition)
- gives the correct MAP assignment
- $\mathbb{M}$ is difficult to specify

Recall: variational derivation of BP

$$\arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} H(q_{i,j}) - \sum_i (|Nb_i| - 1) H(q_i) + \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

Recall: variational derivation of BP

$$\arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} H(q_{i,j}) - \sum_i (|Nb_i| - 1) H(q_i) + \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

$$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$$

$$q_{i,j}(x_i, x_j) \geq 0 \quad \forall i, j \in \mathcal{E}, x_i, x_j$$

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$$

locally consistent
marginal distributions

Recall: variational derivation of BP

$$\arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} H(q_{i,j}) - \sum_i (|Nb_i| - 1) H(q_i) + \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j)$$

$$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i,j \in \mathcal{E}, x_j$$

$$q_{i,j}(x_i, x_j) \geq 0 \quad \forall i,j \in \mathcal{E}, x_i, x_j$$

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$$

locally consistent marginal distributions

BP update is derived as "fixed-points" of the Lagrangian

- BP **messages** are the (exponential form of the) **Lagrange multipliers**

Relationship between LP & BP

sum-product BP objective

pairwise case

$$\arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) \quad + \quad H(q)$$

$$q_{i,j}(x_i, x_j) \geq 0 \quad \forall i, j \in \mathcal{E}, x_i, x_j$$

$$\sum_{x_i} q_i(x_i) = 1 \quad \forall i$$

$$\sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j) \quad \forall i, j \in \mathcal{E}, x_j$$

Relationship between LP & BP

pairwise case

sum-product BP objective

LP objective

$$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) \quad + \quad H(q)$$

Relationship between LP & BP

pairwise case

sum-product BP objective

LP objective

$$\arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) \; + \; H(q)$$

replace $p(x)^{\frac{1}{T}} \propto \prod_{i,j\in\mathcal{E}} \phi_{i,j}(x_i, x_j)^{\frac{1}{T}}$ in the equation above

Relationship between LP & BP

pairwise case

sum-product BP objective

LP objective

$$\arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) \; + \; H(q)$$

replace $p(x)^{\frac{1}{T}} \propto \prod_{i,j\in\mathcal{E}} \phi_{i,j}(x_i, x_j)^{\frac{1}{T}}$ in the equation above

$$\arg\max_{\{q\}} \frac{1}{T} \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) + H(q)$$

Relationship between LP & BP

pairwise case

sum-product BP objective

LP objective

$$\arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) \; + \; H(q)$$

replace $p(x)^{\frac{1}{T}} \propto \prod_{i,j \in \mathcal{E}} \phi_{i,j}(x_i, x_j)^{\frac{1}{T}}$ in the equation above

$$\arg\max_{\{q\}} \frac{1}{T} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) + H(q)$$

$$= \arg\max_{\{q\}} \sum_{i,j \in \mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) + TH(q)$$

Relationship between LP & BP

pairwise case

sum-product BP objective

LP objective

$$\arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) \;+\; H(q)$$

replace $p(x)^{\frac{1}{T}} \propto \prod_{i,j\in\mathcal{E}} \phi_{i,j}(x_i, x_j)^{\frac{1}{T}}$ in the equation above

$$\arg\max_{\{q\}} \frac{1}{T} \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) + H(q)$$

$$= \arg\max_{\{q\}} \sum_{i,j\in\mathcal{E}} \sum_{x_{i,j}} q_{i,j}(x_i, x_j) \ln \phi_{i,j}(x_i, x_j) + TH(q)$$

$T \to 0$

Relationship between LP & BP

sum-product BP for marginalization

at the zero-temperature limit $\lim_{T \to 0} p(x)^{\frac{1}{T}}$

is similar to LP relaxation of MAP inference

they are equivalent for concave entropy approximations

Relationship between LP & BP

sum-product BP for marginalization

at the zero-temperature limit $\lim_{T \to 0} p(x)^{\frac{1}{T}}$

is similar to LP relaxation of MAP inference

they are equivalent for concave entropy approximations

sum-product BP

at the zero-temperature limit $\lim_{T \to 0} p(x)^{\frac{1}{T}}$

is similar to **max-product** BP

they are equivalent for concave entropy approximations

Relationship between LP & BP

sum-product BP for marginalization

at the zero-temperature limit $\lim_{T \to 0} p(x)^{\frac{1}{T}}$

is similar to LP relaxation of MAP inference

they are equivalent for concave entropy approximations

sum-product BP

at the zero-temperature limit $\lim_{T \to 0} p(x)^{\frac{1}{T}}$

is similar to **max-product** BP

they are equivalent for concave entropy approximations

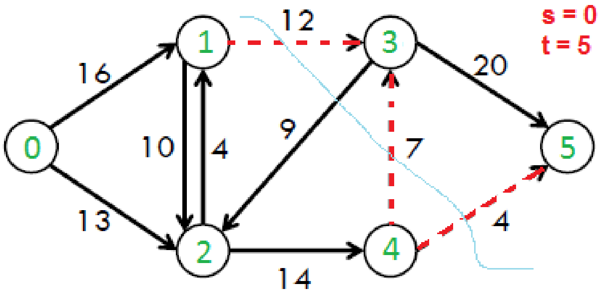
In practice, max-product BP can be much more efficient than LP

- it uses the graph structure

using graph cuts

reduce MAP inference to min-cut problem

use **efficient** & **optimal** min-cut solvers



graph-cut problem: partition the nodes into two sets that include source and target at min cost

$\mathcal{O}(VE)$ algorithms exist

only for a family of factors

arbitrary graph (i.e., large tree width poses no problem)

image: https://www.geeksforgeeks.org

using graph cuts

reduce MAP inference to min-cut problem

use **efficient** & **optimal** min-cut solvers

setting:

- binary pairwise MRF

  $p(x) \propto \exp(-E(x))$

  $E(x) = \sum_i \epsilon_i(x_i) + \sum_{i,j \in \mathcal{E}} \epsilon_{i,j}(x_i, x_j)$

- *sub-modular* $\epsilon_{i,j}$

  $\epsilon_{i,j}(1,1) + \epsilon_{i,j}(0,0) \leq \epsilon_{i,j}(1,0) + \epsilon_{i,j}(0,1)$

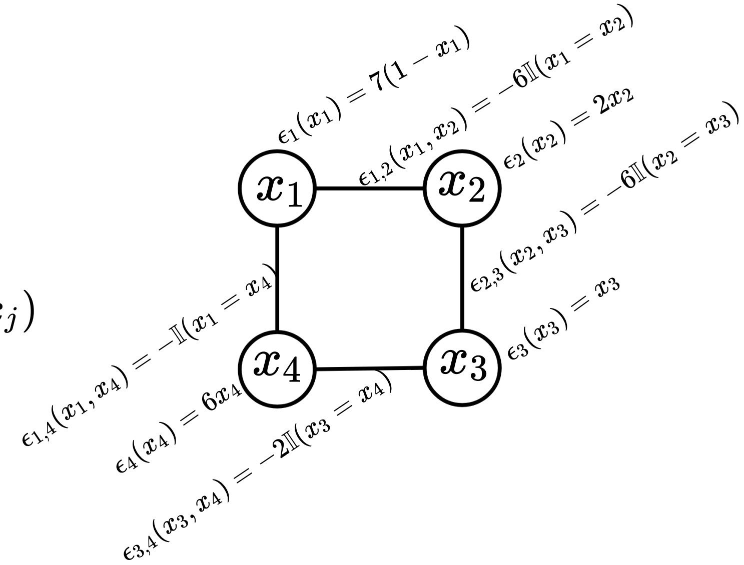
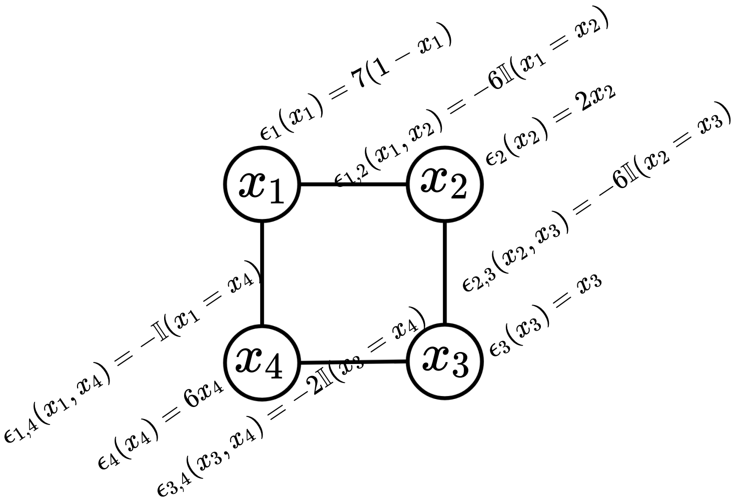


image: https://www.geeksforgeeks.org

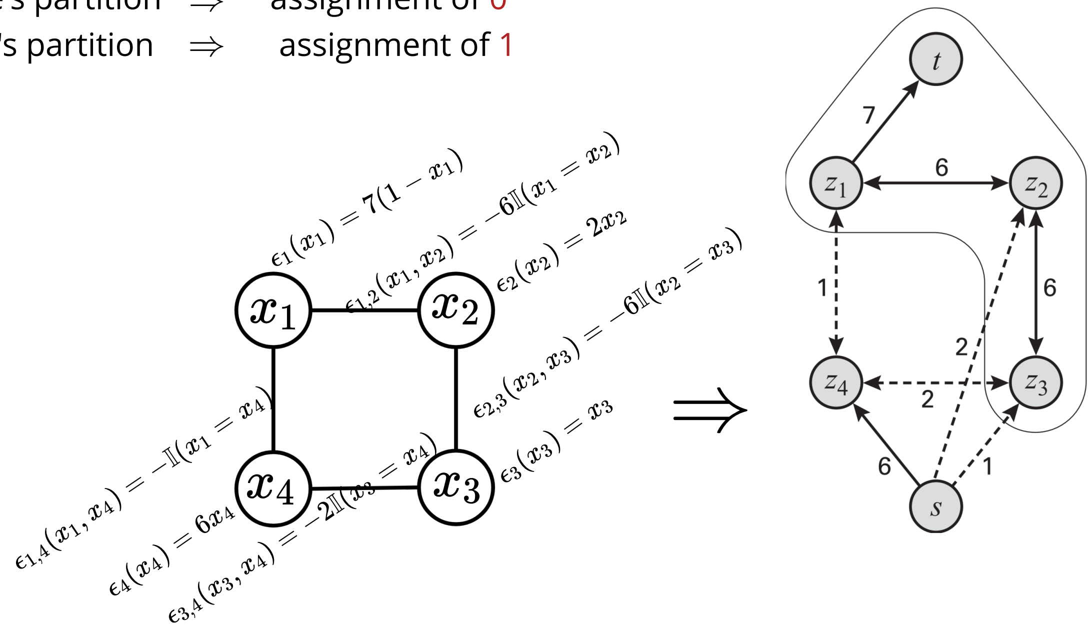
reduction to graph-cuts: example



reduction to graph-cuts: example

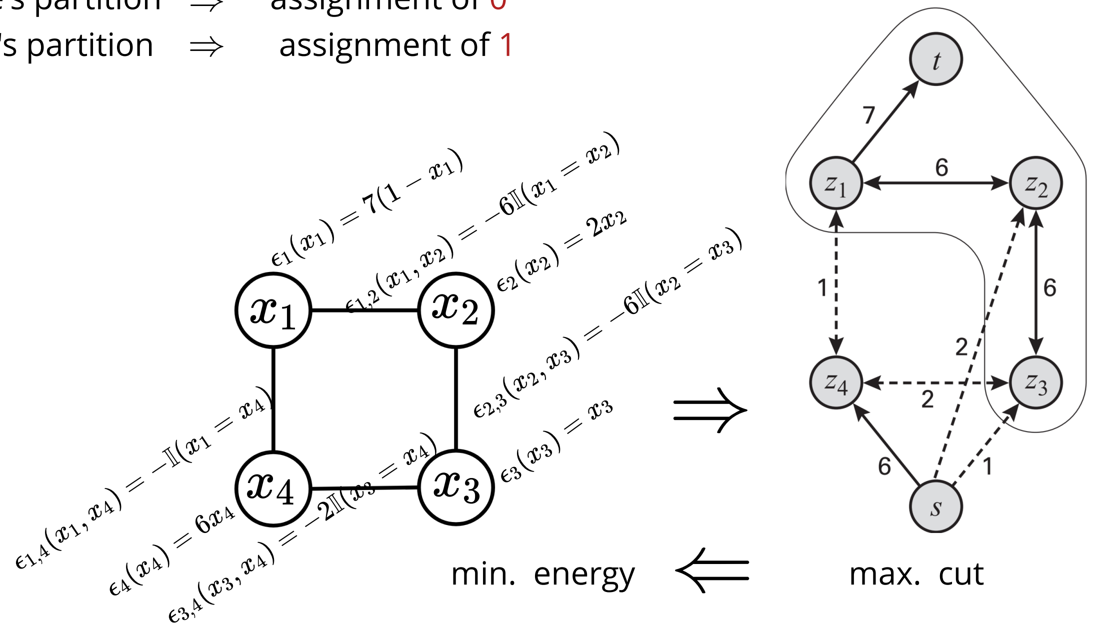
source node's partition ⇒ assignment of 0

target node's partition ⇒ assignment of 1



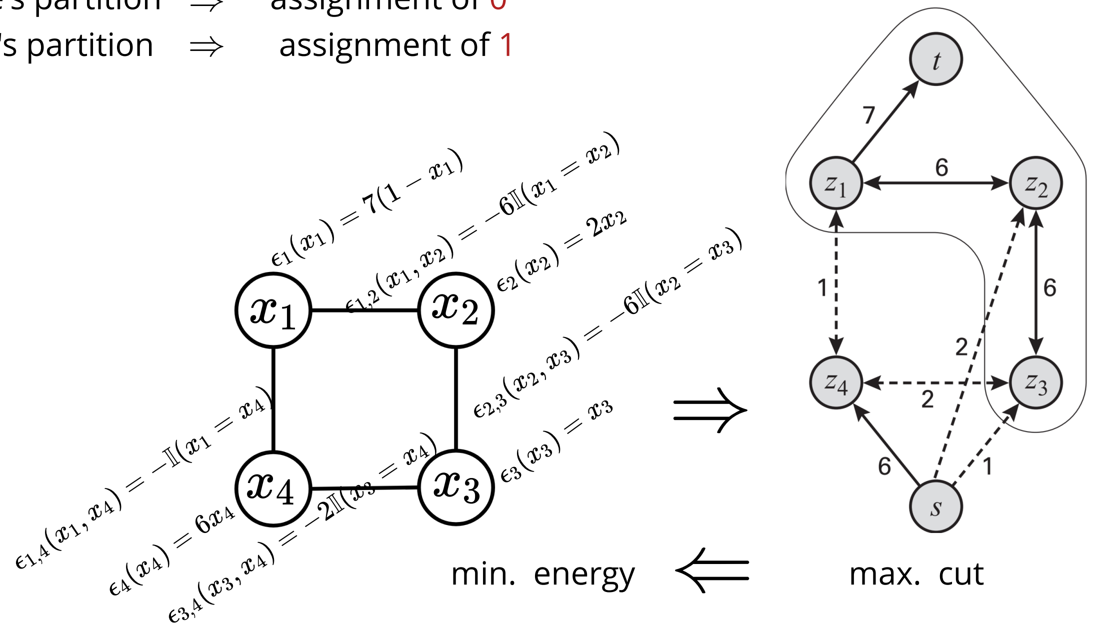
reduction to graph-cuts: example

source node's partition ⇒ assignment of 0
target node's partition ⇒ assignment of 1



reduction to graph-cuts: example

source node's partition $\Rightarrow$ assignment of 0
target node's partition $\Rightarrow$ assignment of 1



non-optimal extensions to variables with higher cardinality

Other methods for MAP inference

- variable elimination
- max-product belief propagation
- IP and LP relaxation
- graph-cuts
- dual decomposition
- branch and bound methods
- local search

Summary

- MAP and marginal MAP are NP-hard
- distributive law extends to MAP inference
  - variable elimination
  - clique-tree
  - loopy BP

an additional challenge of decoding

Summary

- MAP and marginal MAP are NP-hard
- distributive law extends to MAP inference
  - variable elimination
  - clique-tree
  - loopy BP

  an additional challenge of decoding
- variational perspective, connects three approaches:
  - max-product LBP (can find strong local optima!)
  - sum-product LBP (theoretical zero temperature limit)
  - LP relaxations

Summary

- MAP and marginal MAP are NP-hard
- distributive law extends to MAP inference
  - variable elimination
  - clique-tree
  - loopy BP

  an additional challenge of decoding
- variational perspective, connects three approaches:
  - max-product LBP (can find strong local optima!)
  - sum-product LBP (theoretical zero temperature limit)
  - LP relaxations
- for some family of loopy graphs, exact polynomial-time inference is possible (graph-cuts)