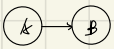
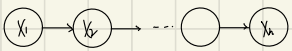
Variable Elimination for Marginal MAP

eg. 

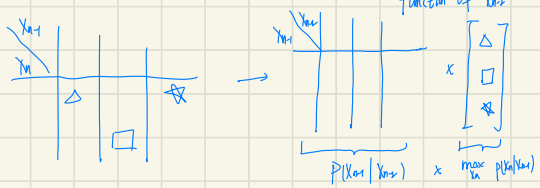
$\max_{a,b} P(a,b) = \max_a \max_b p(a)\cdot p(b|a) = \max_a \max_b p(a)\,p(b|a)$

for any particular value of a, $\max_b p(a)\,p(b|a) = p(a)\cdot \max_b p(b|a)$

the MAP inference turns into a dynamic programming



$$\max_{x_1\cdots x_n} P(x_1,\cdots x_n) = \max_{x_2\cdots x_n} P(x_1)\cdot P(x_2|x_1)\cdot P(x_3|x_2)\cdots P(x_n|x_{n-1})$$
$$= \max_{x_1} P(x_1) \max_{x_2} P(x_2|x_1) \cdots \max_{x_{n-1}} P(x_{n-1}|x_{n-2}) \cdot \underbrace{\max_{x_n} P(x_n|x_{n-1})}_{\text{function of } x_{n-1}}$$
$$\underbrace{\qquad\qquad}_{\text{function of } x_{n-2}}$$



Let X be a set of variables, and $Y \notin X$ a variable, $\phi(X,Y)$ be a factor

define the maximization of Y in ϕ to be a factor over X

$$\psi(X) = \max_Y \phi(X,Y)$$

if $X \notin \text{Scope}[\phi_1]$ then $\begin{cases} \max_X (\phi_1\cdot\phi_2) = \phi_1 \cdot \max_X \phi_2 \\ \max_X (\phi_1+\phi_2) = \phi_1 + \max_X \phi_2 \end{cases}$

```
def max_product_eliminate_variable(Φ: set of factors,
                                   Z: variable to be eliminated)
    Φ' = {φ | φ∈Φ, Z ∈ Scope[φ]}
    Φ'' = {Φ − Φ'}
    ψ = ∏_{φ∈Φ'} φ        // ψ = Σ_{φ∈Φ'} φ in case of max-sum
    τ = max_Z ψ
    return Φ'' ∪ {τ}, ψ
```

```
def max_product_variable_elimination(Φ):
    Let X_1 ... X_n be an ordering of X:
        for i = 1...k:
            Φ, φ_{X_i} = max_product_eliminate_var(Φ, X_i)
    {x_i^*} = traceback_map({φ_{X_i} : i=1...k})
    return {x_i^*}, Φ        // Φ contains the probability of MAP
```

```
def traceback_map({φ_{X_i} : i=1...k}):
    for i = k...1:
        u_i = (x_{i+1}^* ... x_k^*) ⟨ Scope[φ_{X_i}] − {X_i} ⟩
        x_i^* = argmax_{X_i} φ_{X_i}(X_i, u_i)
    return {x_i^*}
```

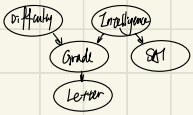
· Variable Elimination for Marginal MAP

$\max_y \sum_w \tilde{P}_\Phi(y,w)$ where $yUW = \mathcal{X}$.

$\max_y \left(\sum_w \prod_{\phi \in \Phi} \phi\right)(y)$

eliminate the variables W by summing them out, and max out Y

eg. $\max_{S,L} \sum_{G,I,D} P(I,D,G,S,L)$



$\max_{S,L} \sum_{G,I,D} \phi_D(D) \cdot \phi_I(I) \cdot \phi_G(G,I,D) \cdot \phi_S(S,I) \cdot \phi_L(L,G)$

$= \max_{S,L} \sum_G \phi_L(L,G) \sum_I \phi_I(I) \cdot \phi_S(S,I) \sum_D \phi_D(D) \cdot \phi_G(G,I,D)$

$\tau_1(I,G)$

$\tau_2(S,G)$

$\tau_3(S,L)$

$\tau_1(I,G) = \sum_D \phi_D(D) \cdot \phi_G(G,I,D)$ // eliminate D by summing out

$\tau_2(S,G) = \sum_I \phi_I(I) \cdot \phi_S(S,I) \cdot \tau_1(I,G)$ // eliminate I by summing out

$\tau_3(S,L) = \sum_G \phi_L(L,G) \cdot \tau_2(S,G)$ // eliminate G by summing out

$\arg\max_{S,L} \tau_3(S,L)$

must perform variable summations before variable maximization

· Max-Product in Clique Trees

For sum-product, we use clique trees to compute the sum-marginals over each cliques in the tree

Idea, we compute a set of max-marginals over each clique

```
def max_message(i: sending clique, j: receiving clique):
    ψ(C_i) = ψ_i · ∏_{k∈Nb(i)−{j}} δ_{k→i}
    τ(S_ij) = max_{C_i − S_ij} ψ(C_i)
    return τ(S_ij)
```

```
def initialize_cliques(Φ: set of factors, α: assignments of factors to cliques):
    for each C_i ∈ α
        ψ_i(C_i) = ∏{φ | φ ∈ α(i)}
    return {ψ_i(C_i)}
```

```
def clique_tree_max_product_calibrate(Φ: set of factors, T: clique tree):
    {ψ_i} = initialize_cliques(Φ, T)
    while ∃(i,j) s.t. i is ready to transmit to j
        δ_{i→j}(S_ij) = max_message(i,j)
    for each clique i
        β_i = ψ_i · ∏_{k∈Nb(i)} δ_{k→i}
    return {β_i}
```

// converge after a upward pass and a downward pass

// the resulting clique tree with beliefs (unnormalized) max-marginal $\tilde{P}_\Phi(\xi)$ of the most likely assignment ξ consistent with each $c_i \in Val(C_i)$

because the max-product message passing process does not compute the partition function (unlike sum-product)
we cannot derive the actual probability of any assignment from marginals.
However, since the partition function is a constant, we can still compare the values of each assignment

$$\max_{C_i - S_{ij}} \beta_i = \max_{C_j - S_{ij}} \beta_j = U_{ij}(S_{ij})$$

β_1	A	B
	a^0	b^0

$= \max_{\Sigma} \hat{P}_\Phi(a^0, b^0, \Sigma)$ $\quad \Sigma = \mathcal{X} - \{A, B\}$

$\max_B \beta_1(B, b) = \max_{\Sigma - \{B\}} \hat{P}_\Phi(b)$

the max-marginals must agree, in this case, the two clusters are max-calibrated

$$\begin{aligned} U_{ij}(S_{ij}) &= \max_{C_i - S_{ij}} \beta_i \\ &= \max_{C_i - S_{ij}} \psi_i \cdot \prod_{k \in (Nb_i - Nb_j)} \delta_{k \to i} \\ &= \max_{C_i - S_{ij}} \psi_i \cdot \delta_{j \to i} \cdot \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i} \qquad \text{// analogous to sum-product} \\ &= \delta_{j \to i} \cdot \max_{C_i - S_{ij}} \psi_i \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i} \\ &= \delta_{j \to i} \cdot \delta_{i \to j} \end{aligned}$$

$U_{ij}(S_{ij}) = \delta_{i \to j} \cdot \delta_{j \to i}$ in a max-calibrated tree (a clique's max-marginal over a sepset can be computed from max-product messages)
can also define a max-product belief update message passing algorithm that's analogous to belief update for sum-product

$$Q_T = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{ij \in E_T} U_{ij}(S_{ij})}$$

$$\left.\begin{aligned} \prod_{i \in V_T} \beta_i(C_i) &= \prod_{i \in V_T} \psi_i \prod_{j \in Nb(i)} \delta_{j \to i} \\ \prod_{ij \in E_T} U_{ij}(S_{ij}) &= \prod_{ij \in E_T} \delta_{j \to i} \delta_{i \to j} \end{aligned}\right\} \frac{\prod_{i \in V_T} \psi_i \prod_{j \in Nb(i)} \delta_{j \to i}}{\prod_{ij \in E_T} \delta_{i \to j} \delta_{j \to i}} = \prod_{i \in V_T} \psi_i = \hat{P}_\Phi$$

```
def initialize_clique_tree(T):
    for each clique C_i ∈ T:
        β_i = ∏{φ : φ ∈ C_i}
    for each edge (i-j) ∈ E_T
        U_ij = 1
```

```
def belief_update_message(i: sending clique, j: receiving clique):
    δ_{i→j} = max_{C_i - S_ij} β_i      // δ_{i→j} = Σ_{C_i - S_ij} β_i in case of sum-product
    β_j = β_j · δ_{i→j} / U_ij
    U_ij = δ_{i→j}
```

```
def clique_tree_max_product_calibrate(E, T):
    initialize_clique_tree(T)
    while exists an uninformed clique in T:
        select (i-j) ∈ E_T
        belief_update_message(i,j)
    return {β_i}
```

// U_{ij} is set to be 1 initially
$\therefore Q_T = \frac{\prod \beta_i}{\prod U_{ij}}$ holds at the beginning

At each iteration, suppose update $i \to j$, $\frac{\beta_j'}{U_{ij}'} = \frac{\beta_j \cdot \frac{\delta_{i \to j}}{U_{ij}}}{\delta_{i \to j}} = \beta_j / U_{ij}$
other beliefs unchange
$\therefore Q_T = \cdots$ holds all the time (deduction)

In an execution of max-product message passing (belief propagation or belief update)
in a clique tree, $Q_T = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{ij \in E_T} U_{ij}(S_{ij})}$ holds throughout the algorithm (as in sum-product)

· Decoding Max-Marginals

these two conditions are equivalent

1° the set of node beliefs $\{\text{MaxMarg}_{\tilde{P}_\Phi}(X_i): X_i \in \mathcal{X}\}$ unambiguous (each belief has a unique max value)

with $x_i^* = \arg\max_{x_i} \text{MaxMarg}_{\tilde{P}_\Phi}(x_i)$ being the unique optimum

2° $\tilde{P}_\Phi$ has a unique MAP assignment $(x_1^*, \dots x_n^*)$ // To prove

---

Let ϕ be a factor over Y, ψ be a factor over Scope $\mathcal{Z} \subseteq \mathcal{Y}$ s.t. $\psi(z) = \max_{y \sim z} \phi(y)$

if $y^* = \arg\max_y \phi(y)$, then y^* is also an optimal assignment for ϕ/ψ

$\phi(A,B)$	A	B	$\psi = \max_B \phi(A,B)$	A		ϕ/ψ	A	B	
	a^1	b^1	0.5	a^1	0.5		a^1	b^1	1
	a^1	b^2	0.2				a^1	b^2	2/5
	a^2	b^1	0	a^2	0		a^2	b^1	1
	a^2	b^2	0				a^2	b^2	1
	a^3	b^1	0.3	a^3	0.45		a^3	b^1	3/5
	a^3	b^2	0.45				a^3	b^2	1

Let $\beta_i(C_i)$ be a belief in a max-calibrated clique tree. An assignment ξ^* has the local optimality property if

$\forall C_i$, $\xi^*\langle C_i\rangle \in \arg\max_{C_i} \beta_i(C_i)$

that is, the assignment to C_i in ξ^* optimizes the C_i belief (decoding: find a local-optimal ξ^*)

Let $\beta_i(C_i)$ be a set of max-calibrated beliefs in a clique tree T, with its associated sepset beliefs.

Let $Q_T = \frac{\prod_i \beta_i(C_i)}{\prod_{ij} \mu_{ij}(S_{ij})}$ be the clique measure. An assignment ξ^* satisfies the local optimality property relative to $\{\beta_i(C_i)\}$ iff ξ^* is the global MAP for Q_T

1° global → local:

$\forall\ c_i \in \text{Val}(C_i)$, $\beta_i(C_i) = \max_{x - C_i} \tilde{P}_\Phi(C_i, x - C_i)$

$\because \xi^* = \arg\max_x \tilde{P}_\Phi(x)$

$\therefore \xi^*\langle C_i\rangle = \arg\max_{C_i} \beta_i(C_i)$

C_i	$x - C_i$
ξ^*	

2° local → global

$$Q_T(\xi^*) = \frac{\prod_i \beta_i(\xi^*\langle C_i\rangle)}{\prod_{ij} \mu_{ij}(\xi^*\langle S_{ij}\rangle)} = \beta_r(\xi^*\langle C_r\rangle) \cdot \frac{\prod_{i \neq r} \beta_i(\xi^*\langle C_i\rangle)}{\prod_{ij} \mu_{ij}(\xi^*\langle S_{ij}\rangle)}$$

$$= \beta_r(\xi^*\langle C_r\rangle) \cdot \prod_{i \neq r} \frac{\beta_i(\xi^*\langle C_i\rangle)}{\mu_{i,\pi(i)}(\xi^*\langle S_{i,\pi(i)}\rangle)}$$

$$= \beta_r(\xi^*\langle C_r\rangle) \cdot \prod_{i \neq r} \frac{\beta_i(\xi^*\langle C_i\rangle)}{\max_{C_i - S_{ij}} \beta_i(C_i)}(\xi^*\langle C_i\rangle)$$

$\because \xi^*\langle C_i\rangle$ maximizes $\beta_i(C_i)$ locally

$\therefore \xi^*\langle C_i\rangle$ is the maximum assignment of $\frac{\beta_i(C_i)}{\max_{C_i - S_{ij}} \beta_i(C_i)}$

$\therefore \xi^*$ is the maximum assignment of Q_T (the global MAP)

Max-Product Belief Propagation in Loopy Cluster Graph

```
def initialize_cluster_graph(U):
    for each cluster C_i
        β_i = ∏{φ ∈ C_i}
    for each edge (i-j) ∈ E_U:
        δ_{i→j} = 1
        δ_{j→i} = 1

def max_message(i: sending clique, j: receiving clique):
    ψ(C_i) = ψ_i ∏_{k ∈ (Nb_i - {j})} δ_{k→i}
    τ(S_ij) = max_{C_i - S_ij} ψ(C_i)   // Σ_{C_i - S_ij} ψ(C_i) in sum-product
    return τ(S_ij)

def cluster_graph_max_product_calibrate(Φ, U):
    initialize_cluster_graph(U)
    while graph is not calibrated
        select (i-j) ∈ E_U
        δ_{i→j}(S_ij) = max_message(i, j)
    for each clique i
        β_i = ψ_i · ∏_{k ∈ Nb_i} δ_{k→i}
    return {β_i}
```

no guarantee that the algorithm will converge, it tends to converge less often than sum-product

at convergences, the result will be a set of calibrated clusters. But the resulting beliefs will not generally be the exact max-marginals. instead, the resulting beliefs are called "pseudo-max-marginals"

In an execution of max-product message passing (whether belief propagation or belief update) in a cluster graph,

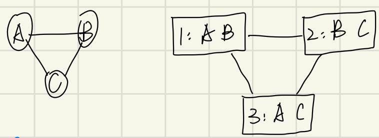
$$\tilde{P}_\Phi = \frac{\prod_i \beta_i(C_i)}{\prod_{(i-j)} \mu_{ij}(S_{ij})}$$

holds initially and after every message-passing step

Decoding the Pseudo-Max-Marginals

In loopy cluster graphs, there may not exsits a joint assignment that satisfies the local optimality property

eg



β_1	A	B	
	a^0	b^0	1
	a^0	b^1	2
	a^1	b^0	2
	a^1	b^1	1

↔

β_2	B	C	
	b^0	c^0	1
	b^0	c^1	2
	b^1	c^0	2
	b^1	c^1	1

β_3	A	C	
	a^0	c^0	1
	a^0	c^1	2
	a^1	c^0	2
	a^1	c^1	1

$\mu_{i,j}(C) = \max_B \beta_2 = $ (C: c_1 2, c_0 2) $= \max_A \beta_3$

(a calibrated cluster graph)

does not exist an assignment ξ^* such that

$\xi^*\langle A,B\rangle$ maxs β_1

⋮

$\xi^*\langle A,C\rangle$ maxs β_3

β_1	A	B	
	a^0	b^0	2
	a^0	b^1	1
	a^1	b^0	1
	a^1	b^1	2

↔

β_2	B	C	
	b^0	c^0	2
	b^0	c^1	1
	b^1	c^0	1
	b^1	c^1	2

β_3	A	C	
	a^0	c^0	2
	a^0	c^1	1
	a^1	c^0	1
	a^1	c^1	2

$\exists\ \xi^* = (a^0, b^0, c^0)\ \ (a^1, b^1, c^1)$

that satisfies the local optimality property

MAP as Relaxed Boolean LP

$\arg\max_{\xi} \log \tilde{P}_{\Phi}(\xi) = \arg\max_{\xi} \sum_{r\in R} \log(\phi_r(C_r))$ where $\Phi = \{\phi_r : r \in R\}$, C_r is the scope of ϕ_r.

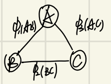
define $n_r = |Val(C_r)|$, $\eta_r^j = \log[\phi_r(c_r^j)]$ when $\xi\langle C_r\rangle = c_r^j$

introduce optimization variables $q(x_r^j)$ where $\begin{cases} r\in R \text{ enumerates different factors} \\ j=1,\dots,n_r \text{ enumerates assignments to } \phi_r \end{cases}$

$$q(x_r^j) = \begin{cases} 1 & \text{if } C_r = c_r^j \\ 0 & \text{otherwise} \end{cases}$$

MAP: $\max_q \sum_{r\in R}\sum_{j=1}^{n_r} \eta_r^j \cdot q(x_r^j) = \eta^T q$

eg.



$|Val(A)| = |Val(B)| = 2$ $n_1 = 4$

$|Val(C)| = 3$ $n_2 = 6$, $n_3 = 6$

$$\max_q \sum_{r=1,2,3}\sum_{j=1}^{n_r} \eta_r^j\, q(x_r^j)$$

s.t. $q(x_r^j) \in \{0,1\}$ $\forall r\in R, j\in\{1,\dots,n_r\}$

$\sum_{j=1}^{n_r} q(x_r^j) = 1$ $\forall r \in R$ // mutual exclusive

$\sum_{\dots} q(x_r^j) = \sum_{\dots} q(x_{r'}^{k})$ i.e.

ϕ_1	A	B
1	a^1	b^1
2	a^1	b^2
3	a^2	b^1
4	a^2	b^2

ϕ_2	B	C
1	b^1	
2		
3		
4	b^2	
5		
6		

$q(x_1^1) + q(x_1^3) = q(x_2^1) + q(x_2^2) + q(x_2^3)$ $B = b^1$

$q(x_1^2) + q(x_1^4) = q(x_2^4) + q(x_2^5) + q(x_2^6)$ $B = b^2$

eg.

θ_1	X_1	
	0	$\theta_1(0)$
	1	$\theta_1(1)$

θ_2	X_2	
	0	$\theta_2(0)$
	1	$\theta_2(1)$

θ_{12}	X_1	X_2	
	0	0	$\theta_{12}(0,0)$
	0	1	$\theta_{12}(0,1)$
	1	0	$\theta_{12}(1,0)$
	1	1	$\theta_{12}(1,1)$

$$u = \begin{bmatrix} \begin{bmatrix}1\\0\end{bmatrix} \\ \begin{bmatrix}0\\1\end{bmatrix} \\ \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \end{bmatrix} \quad \begin{matrix} \text{node potential } X_1=0 \\ \text{node potential } X_2=1 \\ \text{edge potential } (X_1=0, X_2=1) \end{matrix} \quad \begin{bmatrix} \begin{bmatrix}\theta_1(0)\\\theta_1(1)\end{bmatrix} \\ \begin{bmatrix}\theta_2(0)\\\theta_2(1)\end{bmatrix} \\ \begin{bmatrix}\theta_{12}(0,0)\\\theta_{12}(0,1)\\\theta_{12}(1,0)\\\theta_{12}(1,1)\end{bmatrix} \end{bmatrix} = \theta$$

$\max_u \theta^T u$

s.t. $u_i \in \{0,1\}$

$Au = 1$

$Cu = Du$

$A = \begin{bmatrix} 1\ 1 & & \\ & 1\ 1 & \\ & & 1\ 1\ 1\ 1 \end{bmatrix}$ ① θ_1 ② θ_2 ③ θ_{12}

$$\underbrace{\begin{bmatrix} 1 & & & 0&0&0&0 \\ & 1 & & 0&0&0&0 \\ & & 1 & 0&0&0&0 \\ 1&0&0&0&0&0 \end{bmatrix}}_{C} u = \underbrace{\begin{bmatrix} 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&1&0&1 \end{bmatrix}}_{D} u$$

① when $X_1=0$, θ_{12} takes value $(0,0)$ or $(0,1)$

② when $X_2=0$, θ_{12} takes value $(0,0)$ or $(1,0)$

replace the constraint $q(x_r^j) \in \{0,1\}$ with $q(x_r^j) \geq 0$

$\begin{cases} q(x_r^j) \\ \sum_j q(x_r^j) = 1 \end{cases} \Rightarrow q(x_r^j) \in [0,1]$

Find $\{q(x_r^j) : r\in R; j=1,\dots,n_r\}$

max. $\eta^T q$ // relaxation of the original problem.

s.t. $\sum_{j=1}^{n_r} q(x_r^j) = 1$

$\sum q(x_r^j) = \sum q(x_{r'}^{k})$

$q \geq 0$

MAP via Dual Decomposition

$p(x_1 \cdots x_n) = \frac{1}{Z} \cdot \prod_c \phi(x_c)$

$\max \log p(x_1 \cdots x_n) \Rightarrow \max \sum_c \log \phi(x_c)$

if the graphical model has unary and binary factors only

can rewrite $p(x_1 \cdots x_n)$ as $\arg\max_x \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j)$

$\min_{x_1, x_2, y} f_1(x_1, y) + f_2(x_2, y)$

$\min_{x_1, x_2, y_1, y_2} f_1(x_1, y_1) + f_2(x_2, y_2)$ s.t. $y_1 = y_2$

$L(x_1, x_2, y_1, y_2, v) = f_1(x_1, y_1) + f_2(x_2, y_2) + v^T(y_1 - y_2)$

$g(v) = \inf_{x_1, x_2, y_1, y_2} L = \inf_{x_1, y_1} f_1(x_1, y_1) + v^T y_1 + \inf_{x_2, y_2} f_2(x_2, y_2) - v^T y_2$

$\nabla g(v) = y_1 - y_2$

when $y_1 = y_2$ $g(v) = \inf_{x_1, x_2, y_1, y_2} L = \inf_{x_1, x_2, y_1, y_2} f_1 + f_2$ achieve optimal

while True:

$x_1, y_1 = \arg\min_{x_1, y_1} f_1(x_1, y_1) + v^T y_1$

$x_2, y_2 = \arg\min_{x_2, y_2} f_2(x_2, y_2) - v^T y_2$

if $|y_1 - y_2| \le \varepsilon$:

return $x_1, x_2, (y_1 + y_2)/2$

$v_t += \alpha (y_1 - y_2)$

$\min_{x_1, x_2} f_1(x_1) + f_2(x_2)$

s.t. $x_1 \in C_1, x_2 \in C_2$ $h_1(x_1) + h_2(x_2) \le 0$

$L(x_1, x_2, \lambda) = f_1(x_1) + f_2(x_2) + \lambda^T [h_1(x_1) + h_2(x_2)]$

while True:

$x_1 = \inf_{x_1} f_1(x_1) + \lambda^T h_1(x_1)$ s.t. $x_1 \in C_1$

$x_2 = \inf_{x_2} f_2(x_2) + \lambda^T h_2(x_2)$ s.t. $x_2 \in C_2$

$\lambda = [\lambda + \alpha_k [h_1(x_1) + h_2(x_2)]]_+$

$\min_x -\sum_{i \in V} \log \theta_i(x_i) - \sum_{f \in F} \log \theta_f(x_f)$ f are multivariate potentials

s.t. $x_i^f = x_i$ $\forall f, i$

$x_i \in \text{val}(X_i)$

$L(x_i, x_f, v) = -\sum_{i \in V} \log \theta_i(x_i) - \sum_{f \in F} \log \theta_f(x_f) + \sum_f \sum_{i \in \text{scope}(f)} v_{f,i}(x_i^f - x_i)$

$= \sum_{i \in V} \left[-\log \theta_i(x_i) - \sum_{f : i \in \text{scope}(f)} v_{f,i} x_i \right] + \sum_f \left[-\log \theta_f(x_f) + \sum_{i : i \in \text{scope}(f)} v_{f,i} x_i^f \right]$

repeat {

subproblem for node potentials

$\min_{x_i} -\log \theta_i(x_i) - \sum_{f : i \in \text{scope}(f)} v_{f,i} x_i$

s.t. $x_i \in \text{scope}[x_i]$

subproblem for clique potentials

$\min_{x_f} -\log \theta_f(x_f) + \sum_{i \in \text{scope}(f)} v_{f,i} x_i^f$

s.t. $x_f \in \text{val}(f)$

} Solved by enumeration

if $x_{f,i} = x_i \; \forall f, i$

return $\{x_f\}$ $\{x_i\}$

$v_{f,i} += x_{f,i} - x_i$

}

$$\inf_{x_i \in C_i} \sum_i \left[-\ln \theta_i(x_i) - \sum_{f:\, i \in \mathrm{scope}(f)} V_{fi}(x_i) \right] + \inf_{x_f \in C_f} \sum_f \left[-\ln \theta_f(x_f) + \sum_{i \in \mathrm{scope}(f)} V_{fi}(x_{fi}) \right]$$

$$= \inf_{x_i \in C_i,\, x_f \in C_f} -\sum_i \ln \theta_i(x_i) - \sum_f \ln \theta_f(x_f) + \sum_f \sum_{i \in \mathrm{scope}(f)} V_{fi}(x_{fi} - x_i)$$

$$\le \inf_{\substack{x_i \in C_i,\, x_f \in C_f \\ x_{fi} = x_i}} -\sum_i \ln \theta_i(x_i) - \sum_f \ln \theta_f(x_f)$$

$$= p^*$$

weak duality

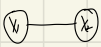
when the algorithm converges, $x_i = \arg\min_{x_i} -\ln \theta_i(x_i^*) - \sum_{f:\, i \in \mathrm{scope}(f)} V_{fi}^*(x_i^*)$

$$x_f = \arg\min_{x_f} -\ln \theta_f(x_f^*) + \sum_{i \in \mathrm{scope}(f)} V_{fi}^*(x_{fi}^*)$$

$x_{fi}^* = x_i^*$

$$\tilde{f}(x_i^*, x_f^*, V_{fi}^*) = \inf_{\substack{x_i \in C_i \\ x_f \in C_f}} \tilde{f}(x_i, x_f, V_{fi}^*) = \inf_{\substack{x_i \in C_i, \\ x_f \in C_f \\ x_{fi} = x_i}} f(x_i, x_f) = p^*$$

strong duality

eg. 

$C_1 = C_2 = \{0, 1\}$ $\quad C_{12} = \{(0,0), (0,1), (1,0), (1,1)\}$

$x_1 \in C_1 \quad x_2 \in C_2 \quad x_{12} \in C_{12}$

$$\min_{x_1, x_2, x_{12}} -\sum_{i=1,2} \ln \theta_i(x_i) - \ln \theta_{12}(x_{12})$$

s.t. $x_1 \in C_1 \quad x_2 \in C_2 \quad x_{12} \in C_{12}$

$x_{12}[1] = x_1 \quad x_{12}[2] = x_2$

$$\tilde{f}(x_1, x_2, V_{12}) = -\ln \theta_1(x_1) - \ln \theta_2(x_2) - \ln \theta_{12}(x_{12}) + V_{12}[1](x_{12}[1] - x_1) + V_{12}[2](x_{12}[2] - x_2)$$

while True:

$x_1 = \arg\min_{x_1} -\ln \theta_1(x_1) - V_{12}[1] x_1$

$x_2 = \arg\min_{x_2} -\ln \theta_2(x_2) - V_{12}[2] x_2$

$x_{12} = \arg\min_{x_{12}} -\ln \theta_{12}(x_{12}) + V_{12}[1] \cdot x_{12}[1] + V_{12}[2] x_{12}[2]$

if $x_{12}[1] = x_1$ and $x_{12}[2] = x_2$:

return x_{12}

$V_{12}[1] \mathrel{+}= x_{12}[1] - x_1$

$V_{12}[2] \mathrel{+}= x_{12}[2] - x_2$