Exact Inference: Variable Elimination
10708, Fall 2020
Pradeep Ravikumar

1 Introduction

Let $X = (X_1, \ldots, X_p)$ be the random vector associated with a PGM. Earlier, we were interested in marginal (conditional) probabilities, which capture the uncertainty in values of a particular variable, conditioned on available evidence. Here we are interested in the *most likely configuration* of a subset of variables instead. In our medical diagnosis running example, we are interested in the most likely values for a set of binary variables corresponding to diseases of interest, conditioned on some observed symptoms.

The so-called MAP (for Maximum A Posteriori) query takes the form:

$$\arg\max_{x_F} P(x_F | X_E = x_E),$$

where $X_F \cup x_E = \mathcal{X}$.

Note that $\arg\max_{x_F} P(x_F | X_E = x_E) = \arg\max_{x_F} P(x_F, X_E = x_E)$. Consider the evidence-incorporated PGM:

$$\widetilde{P}(X) = P(X) \prod_{X_e \in X_E} \delta_{x_e}(X_e),$$

which simply introduces some additional node-potentials specifying the evidence. It can be then be seen that the MAP query reduces to:

$$\arg\max_{x} \widetilde{P}(x),$$

so that we will be considering this unconditioned form in the sequel, assuming that any evidence has been incorporated via node-potentials.

The reason we make this distinction is that there is yet another, and more complex, query called the marginal-MAP query that takes the form:

$$\arg\max_{x_F} \sum_{x_R} P(x_F, x_R | X_E = x_E),$$

where $X_F \cup X_E \cup X_R = \mathcal{X}$. Again by incorporating evidence potentials this can be stated simply as:

$$\arg\max_{y} \sum_{z} P(y, z),$$

where $Y \cup Z = \mathcal{X}$. This can be seen to encompass MAP queries when $Z = \varnothing$, but when Z is not null, this is a more complex query since it involves both marginalization and maximization.

2 Complexity

Just as with computing marginals, MAP estimation also turns out to be computationally intractable for general PGMs.

Theorem 1 *Consider the following decision problem: Given a discrete DGM over $\mathcal{X}$, a number $\tau > 0$, decide if there exists an assignment x to $\mathcal{X}$ such that $P(x) > \tau$. This decision problem is NP-complete.*

Since marginal MAP is a generalization of MAP, it follows that it is NP-hard, and in fact we have the following theorem.

Theorem 2 *Consider the following decision problem: Given a discrete DGM over $\mathcal{X}$, a subset $\mathbf{Y} \subset \mathcal{X}$, a number $\tau > 0$, decide if there exists an assignment y to $\mathbf{Y}$ such that $P(y) > \tau$. This decision problem is NP^{PP}-complete.*

For certain graphs e.g. polytrees (where the undirected skeleton is a tree), MAP in corresponding DGMs can be performed in linear time (as we will be seeing shortly). Whereas even for polytrees, marginal MAP is NP-hard.

3 Warm up

As with computing marginals, we assume we have a set of factors Φ over $\mathcal{X}$ that specify a potentially unnormalized joint distribution over $\mathcal{X}$:

$$\widetilde{P}(X) = \prod_{\phi \in \Phi} \phi(X).$$

When computing the MAP, we do not care about the normalization constant:

$$\arg\max_x P(x) = \arg\max_x \frac{\widetilde{P}(x)}{Z} = \arg\max_x \widetilde{P}(x).$$

The problem

$$\max_x \widetilde{P}(x) = \max_x \prod_{\phi \in \Phi} \phi(x),$$

can be seen to be a *max-product* problem, in contrast to the sum-product nature of marginal computation. Since log is a monotonic function, we could also state it as:

$$\arg\max_x \log \widetilde{P}(x) = \arg\max_x \sum_{\phi \in \Phi} \log \phi(x),$$

which can be seen as a *max-sum* problem. A related class of problems are so-called energy minimization problems, where if we have that the factors are strictly positive, we can write them as: $\phi(x) = \exp(-E_\phi(x))$, so that the MAP problem reduces to:

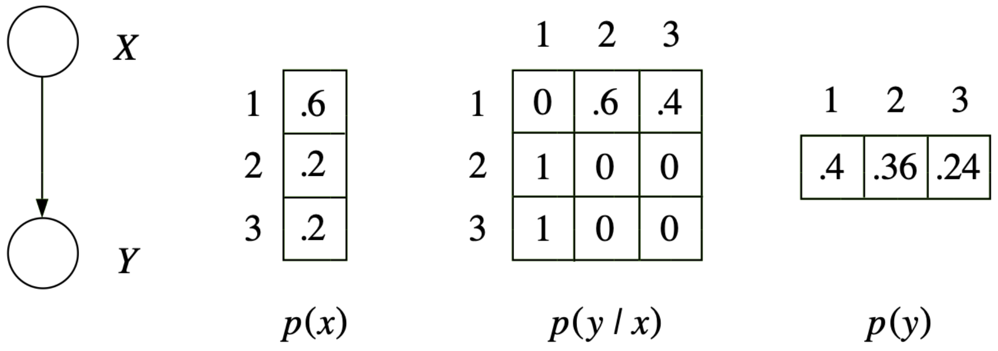
$$\arg\min_x \sum_{\phi \in \Phi} E_\phi(x),$$

which is an energy minimization problem. Such a transformation typically leads to much more numerically stable calculations as compared to the max-product form.

3.1 Would Marginals Suffice?

One thought might be that marginals, which we saw how to compute earlier via variable elimination and junction tree algorithms, might suffice to compute the most likely assignment. After all, if we have a single variable Y, and we knew its probability distribution $P(Y)$, we could compute its MAP assignment simply via $\arg\max_y P(Y)$. So if we have a collection of marginals $\{P(X_i)\}_{i=1}^p$, then perhaps the MAP assignment x^* would simply be:

$$x_i^* \in \arg\max_{x_i} P(X_i).$$



	$p(x)$
1	.6
2	.2
3	.2

$p(y\mid x)$	1	2	3
1	0	.6	.4
2	1	0	0
3	1	0	0

1	2	3
.4	.36	.24

Figure 1: Two Node DGM

Consider the two node DGM in Figure 1. It can be seen that if we computed the individual marginal based MAP assignments, we would get:

$$x^* = 1,\ y^* = 1,$$

but it can be seen that for the joint distribution $P(X, Y)$, this assignment has probability zero, while the MAP assignment for the joint distribution $P(X, Y)$ is $(2, 1)$.

Consider the six node DAG in Figure 2. Consider a discrete RV $X = (X_1, \ldots, X_6)$ associated with the DAG. Suppose for this DGM, we wish to compute $\max_x P(x)$.

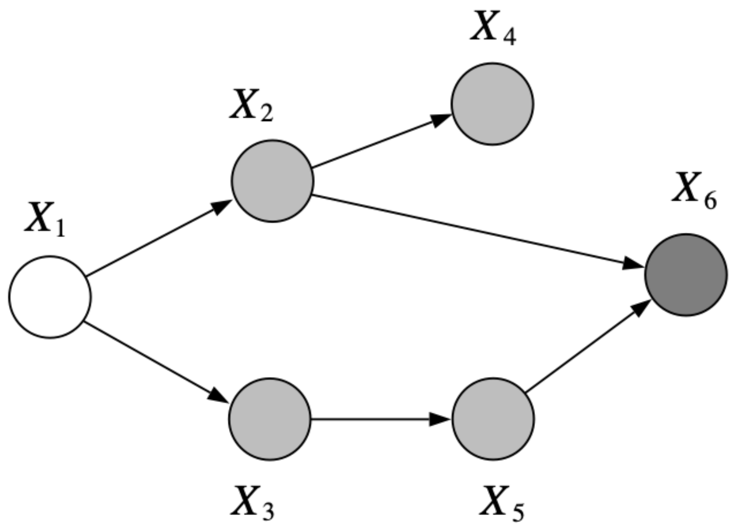


Figure 2: Six Node DGM

Brute Force Approach. One approach is to explicitly construct the probability table for $P(X)$. Suppose each variable takes K possible values. Then the probability table has size K^6. It can be seen that by visiting each of the K^6 entries of the table, we can compute the MAP assignment. But this is expensive, even more so if the number of variables were much larger. And the brute force approach does not really leverage the key promise of PGMs — local representations and hence small representational complexity — since the computations involved are global.

Distributive Property. Consider a product of factors: $\phi(X) = \phi_1(X).\phi_2(X)$, and suppose we wish to compute: $\max_{x_s} \phi$. Suppose $X_s \notin \text{scope}(\phi_1)$. Then,

$$\max_{x_s} \phi_1.\phi_2 = \phi_1. \max_{x_s} \phi_2.$$

Thus, the max and the product terms can be exchanged (just as with sum and product). Note that after computing the max, the residual functions do not depend on X_s i.e. the variable X_s is eliminated. It is thus also called max-product elimination of variable X_s.

Let us consider the example in Figure 2, and apply this "max product variable elimination" principle.

$$\begin{aligned}
\max_{x_1,x_2,x_3,x_4,x_5,x_6} P(x) = &= \max_{x_1,x_2,x_3,x_4,x_5,x_6} P(x_1)P(x_2 \,|\, x_1)P(x_3 \,|\, x_1)P(x_4 \,|\, x_2)P(x_5|x_3)P(x_6 \,|\, x_2, x_5) \\
&= \max_{x_1} P(x_1) \max_{x_2} P(x_2 \,|\, x_1) \max_{x_3} P(x_3 \,|\, x_1) \max_{x_4} P(x_4 \,|\, x_2) \max_{x_5} P(x_5|x_3) \max_{x_6} P(x_6 \,|\, x_2, x_5) \\
&= \max_{x_1} P(x_1) \max_{x_2} P(x_2 \,|\, x_1) \max_{x_3} P(x_3 \,|\, x_1) \max_{x_4} P(x_4 \,|\, x_2) \max_{x_5} P(x_5|x_3) m_6(x_2, x_5) \\
&= \max_{x_1} P(x_1) \max_{x_2} P(x_2 \,|\, x_1) \max_{x_3} P(x_3 \,|\, x_1) \max_{x_4} P(x_4 \,|\, x_2) m_5(x_3, x_2) \\
&= \max_{x_1} P(x_1) \max_{x_2} P(x_2 \,|\, x_1) \max_{x_3} P(x_3 \,|\, x_1) m_4(x_3, x_2) \\
&= \max_{x_1} P(x_1) \max_{x_2} P(x_2 \,|\, x_1) m_3(x_1, x_2) \\
&= \max_{x_1} P(x_1) m_2(x_1)
\end{aligned}$$

so that all calculations are the same as sum-product variable elimination except that sums are replaced with max.

Commutative Semiring. So why the commonality between sum product and max product (all calculations seem similar: we just subtitute sum for max)? And it can be shown this also extends to the max-sum version of the MAP problem.

The commonality to both is the notion of a commutative semiring. This is a set endowed with two operations, $+$ and $\times$ where

- $+$ and $\times$ are associative $(a + (b + c) = (a + b) + c)$, commutative $(a + b = b + a)$
- $\times$ is distributive over $+$: $a \times b + a \times c = a \times (b + c)$

It can be seen that the set of reals with sum as $+$, and product as $\times$ form a semiring. As do max as $+$, and product as $\times$; or with max as $+$, and sum as $\times$. It is these sets of properties associated with a commutative semiring which allows us to perform the variable elimination calculations.

4 Variable Elimination

Definition 3 (Factor Marginalization) *Let $\mathbf{X}$ be a set of variables, and let $\phi(\mathbf{X})$ be some factor over these variables. Suppose $Y \in \mathbf{X}$. We then define the factor maximization of Y in $\phi(\cdot)$ as new factor ψ with scope $\mathbf{X} - \{Y\}$ s.t.:*

$$\psi(\mathbf{X} - \{Y\}) = \max_Y \phi(\mathbf{X}).$$

This is the key operation we will require when performing variable elimination, as shown in Algorithm 1.

Algorithm 1 Variable Elimination

```
Input: Set of factors $\Phi$, ordering $\sigma$ of variables to be eliminated
for $i = 1, \dots, p$ do
    Find all factors in $\Phi$ that reference variable $X_{\sigma(i)}$, and remove them from $\Phi$
    Let $\Phi_i(T_i)$ denote the product of these factors, where $T_i = \text{scope}(\phi)$ (i.e. set of variables
    referenced by $\phi(\cdot)$
    Factor maximize out variable $X_{\sigma(i)}$ from $\Phi_i$ and suppose $m_i(S_i)$ is the new factor. Add
    this new factor to $\Phi$
end for
for $i = p, \dots, 1$ do
    $x^*_{\sigma(i)} = \arg\max_{x_{\sigma}(i)} \Phi_i(x_\sigma(i), x^*_{\sigma(i+1)}, \dots, x^*_{\sigma(p)})$
    Return $x^*$
end for
```

Note that the algorithm has two stages: the first stage eliminates variables with respect to a particular ordering, and generates the VE induced factors $\{\Phi_i\}_{i=1}^p$. In the second stage — which was not present in sum-product elimination — we then do a traceback to figure out the maximizing configuration. Here we go back to front with respect to the ordering σ: since $X_{\sigma(p)}$ was the last to be eliminated, its generated factor $\Phi_p(X_{\sigma(p)})$ has the information to specify the MAP assignment value for $X_{\sigma(p)}$. We then set the value of $X_{\sigma(p)}$ to that MAP assignment value, and continue with the remaining variables similarly.

The following theorem states that the algorithm does generate the correct answer.

Theorem 4 *Let* $\mathbf{X}$ *be some set of variables, and let* Φ *be some set of factors over these variables, so that* $\cup_{\phi\in\Phi} \mathit{scope}(\phi) \subseteq \mathbf{X}$*. Let* $\sigma \in \mathcal{S}_p$ *be an ordering of the variables that we eliminate using the variable elimination algorithm in Algorithm 1. Then,* x^* *generated by VE satisfies:*

$$x^* = \arg\max_x \prod_{\phi\in\Phi} \phi(x),$$

and the single factor Φ_0 *of empty scope obtained after eliminating all variables satisfies:*

$$\Phi_0 = \max_x \prod_{\phi\in\Phi} \phi(x).$$

5 Junction Tree Message Passing

One can also extend the sum-product junction tree message passing to a corresponding max-product algorithm. Suppose we have a junction tree T with clique factors $\{C_i\}_{i=1}^k$. Given a set of factors Φ, we assume we assign each factor $\phi \in \Phi$ to some clique factor $\alpha(\phi) \in [k]$ so that we get clique factor functions:

$$\Psi_i(C_i) = \prod_{\phi \mid \alpha(\phi)=i} \phi,$$

and where each factor ϕ is assigned to a unique clique factor function so that

$$\prod_{\phi \in \Phi} \phi = \prod_{i=1}^{k} \psi_i.$$

Max-Product Message. We can then compute the message from clique factor C_i to clique factor C_j as:

$$m_{i \to j} = \max_{C_i \setminus C_j} \psi_i \prod_{k \in \text{NBRS}_i - \{j\}} m_{k \to i}.$$

We pass these messages following the same protocol as with sum-product messages.

Beliefs. Once all possible messages are passed, i.e. once in each direction of the edges of the clique tree, then we can compute *max-marginal beliefs* at each clique factor as:

$$\beta_i(C_i) = \psi_i \prod_{j \in \text{NBRS}_i} m_{j \to i}.$$

Note that these beliefs β_i are functions over the clique variables. Now in the case of sum-product message passing, these converged to the marginals over the clique variables. But what could they converge to in this max-product case? For this we need the notion of a *max-marginal.*

Definition 5 (Max-Marginal) *For any unnormalized probability distribution $\widetilde{P}$ over $\mathcal{X}$, its max-marginal over a subset of variables $Y \subset \mathcal{X}$ is given as:*

$$\textit{max-marginal}_{\widetilde{P}_\Phi}(y) = \max_{x \,:\, x[Y]=y} \widetilde{P}_\Phi(x).$$

Thus, it is a function over a subset of variables Y, where for any configuration y of the variables Y, it specifies the max probability over assignments to the other variables.

We have the following theorem.

Theorem 6 *Given a set of factors Φ over $X = (X_1, \dots, X_p)$, let $\widetilde{P} = \prod_{\phi \in \Phi} \phi$. Note that $\widetilde{P}(X)$ is equal to the joint density $P(X)$ in the case of DAG factors, and is equal to the unnormalized density in the case of UG factors. Then the max-marginal beliefs $\{\beta_i\}_{i \in [m]}$ calculated at the end of message passing in a clique tree with clique factors $\{C_i\}_{i=1}^m$ satisfy:*

$$\beta_i(c_i) = \textit{max-marginal}_{\widetilde{P}_\Phi}(c_i) = \max_{x : x[C_i] = c_i} \widetilde{P}(x).$$

And because the max-marginal beliefs over adjacent cliques must agree we have that:

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

5.1 Belief-Update Message Passing

We can also define the corresponding belief-update version of the max-product messages.

Here, we will make use of two sets of max-marginal belief functions: $\beta_i(C_i)$ for all the clique factors $\{C_i\}_{i=1}^k$, as well as μ_{ij} for the separating sets $S_{ij} = C_i \cup C_j$ for all $(i,j) \in E(T)$.

We can then compute the message from clique factor C_i to clique factor C_j as:

$$\begin{aligned} m_{i \to j}^{\text{HUGIN}} &= \max_{C_i - S_{ij}} \beta_i \\ \beta_j &= \beta_j \frac{m_{i \to j}^{\text{HUGIN}}}{\mu_{ij}} \\ \mu_{ij} &= m_{i \to j}^{\text{HUGIN}}. \end{aligned}$$

Thus, the HUGIN message consists of what clique factor $\psi_i(C_i)$ believes should be the max-marginal over S_{ij}. We then use this to update both the belief of the recipient clique factor, as well as the belief of the separating set.

The message passing protocols are as with the sum-product messages.

We have the following theorem.

Theorem 7 *Given a set of factors Φ over $X = (X_1, \dots, X_p)$, let $\widetilde{P} = \prod_{\phi \in \Phi} \phi$. Note that $\widetilde{P}(X)$ is equal to the joint density $P(X)$ in the case of DAG factors, and is equal to the unnormalized density in the case of UG factors. Then the beliefs $\{\beta_i\}_{i \in [m]}$, and $\{S_{ij}\}_{(i,j) \in E(T)}$ calculated at the end of HUGIN message passing in a clique tree with clique factors $\{C_i\}_{i=1}^m$ satisfy:*

$$\begin{aligned} \beta_i(c_i) &= \textit{max-marginal}_{\widetilde{P}_\Phi}(c_i) = \max_{x : x[C_i] = c_i} \widetilde{P}(x) \\ \mu_{ij}(s_{ij}) &= \textit{max-marginal}_{\widetilde{P}_\Phi}(s_{ij}) = \max_{x : x[S_{ij}] = s_{ij}} \widetilde{P}(x). \end{aligned}$$

5.2 Reparameterization

As with the sum-product messages, we can show that the intermediate messages in max-product message passing, and the intermediate beliefs in both satisfy:

$$\beta_i(C_i) = \psi_i \prod_{j \in \text{NBRS}_i} m_{j \to i},$$
$$\mu_{ij}(S_{ij}) = m_{i \to j}\, m_{j \to i},$$

provided they are initialized to also satisy these constraints.

And what's more, these intermediate beliefs form a reparameterization of the unnormalized distribution so that:

Theorem 8 *At any stage of max-product message passing, or its belief update variant, we have that:*

$$\widetilde{P}_\Phi(X) = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i,j) \in E(T)} \mu_{ij}(S_{ij})}.$$

6 Decoding Max-marginals

So far there seems to be a strong commonality between sum-product and max-product versions of the algorithms. In the former, we obtain clique and separating set marginals, and in the latter, we get corresponding max-marginals. There is however a crucial distinction: in the former case, the marginals were exactly what we were after. Whereas, here, it is not clear how we would use the max-marginals. Recall, that what we were actually after was the MAP assignment. Could we obtain the MAP assignment from the max-marginals?

Example 9 *Consider an XOR-like distribution $P(X_1, X_2)$, over binary variables $X_1, X_2 \in \{0, 1\}$:*

$$P(X_1, X_2) = 0.1\mathbb{I}[X_1 = X_2] + 0.4\mathbb{I}[X_1 \neq X_2].$$

For each value of X_1, there is a corresponding value of X_2 so that the joint probability is maximized and equal to 0.4, and similarly for X_2, so that we have:

$$\textit{max-marginal}_P(x_1) = 0.4$$
$$\textit{max-marginal}_P(x_2) = 0.4.$$

Thus, setting the assingment at X_1 and X_2 independently looking just at their max-marginals would allow for selecting $(0, 0)$ as a configuration, which however has probability of just 0.1 and is not a MAP configuration.

As the example demonstrates, even though the definition of the MAP marginal maximizes over the other variables, when there is ambiguity at the local level of individual max-marginals, we do not have information about what value to pick to get a joint assignment that has highest probability. But if each max-marginal has a unique value, then intuitively, maximizing locally would be OK. The following theorem formalizes this latter intuition.

Theorem 10 *A configuration $x^* = (x_1^*, \dots, x_p^*)$ is the unique MAP assignment for the (unnormalized) distribution $\widetilde{P}_\Phi$ iff the max-marginals $\{\textit{max-marginal}_{\widetilde{P}_\Phi}(X_i)\}_{i=1}^p$ have unique optimizing values*

$$x_i^* = \arg\max_{x_i} \textit{max-marginal}_{\widetilde{P}_\Phi}(x_i), \quad i \in [p].$$

At the end of the max-product message passing, we have that the max-marginal beliefs $\{\beta_i\}_{i \in V_T}$ and $\{\mu_{ij}\}_{(i,j) \in E(T)}$ exactly correspond to the true max-marginals of the unnormalized distribution $\widetilde{P}_\Phi$. And consequently are also calibrated so that:

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

We can moreover talk about an assignment x^* that is locally optimal in the sense of the definition below.

Definition 11 *Suppose we have a set of calibrated max-marginal beliefs $\{\beta_i\}_{i \in V_T}$ and $\{\mu_{ij}\}_{(i,j) \in E(T)}$ for a clique tree T. We say that an assignment x^* is locally optimal wrt clique tree T if:*

$$x^*[C_i] \in \arg\max_{c_i} \beta_i(c_i).$$

Thus a locally optimal x^* maximizes each of the max-marginal beliefs. Note that a locally optimal assignment with respect to a general clique tree need not also be a globally optimal MAP assignment. But it will be if T is a junction tree.

Theorem 12 *Suppose we have a set of calibrated max-marginal beliefs $\{\beta_i\}_{i \in V_T}$ and $\{\mu_{ij}\}_{(i,j) \in E(T)}$ for a junction tree T wrt some factors Φ over $\mathcal{X}$. Then an assignment x^* is locally optimal wrt the junction tree T iff it is the globally optimal MAP assignment for $\widetilde{P}_\Phi$.*

Proof. We know that if T is a junction tree, then calibrated beliefs (local consistency) entails that the beliefs are also the true max-marginal beliefs for the (unnormalized) distribution $\widetilde{P}_\Phi$. Thus, if x^* is the true MAP assignment then it also maximizes the individual clique max-marginals. For the only if direction, we use the reparameterization of the (unnormalized) distribution $\widetilde{P}_\Phi$:

$$\begin{aligned}\widetilde{P}_\Phi &= \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i,j) \in E(T)} \mu_{ij}(S_{ij})} \\ &= \beta_r(C_r) \prod_{i \neq r} \frac{\beta_i(C_i)}{\mu_{i\mathrm{pa}(i)}(S_{i\mathrm{pa}(i)})},\end{aligned}$$

where in the second line we have re-arranged the terms according to any rooting of the clique tree, say with root C_r and where $\mathrm{pa}(i)$ is the parent clique of C_i in the rooted tree. Then, if x^* maximizes the individual clique max-marginals, it can be shown with some calculations that is also maximizes the local ratios $\frac{\beta_i(C_i)}{\mu_{i\mathrm{pa}(i)}(S_{i\mathrm{pa}(i)})}$, so that it follows that it is also the global MAP assignment. $\square$

Thus if we have a junction tree, with local max-marginals, then the task of computing a MAP assignment can be reduced to the task of computing a locally optimal assignment. This decoding task can be accomplished by a traceback procedure similar to the VE traceback.

We first pick a rooting of the clique tree, and pick any ordering σ over the cliques that proceed from the leafs onto the root. Say the first leaf clique node in the ordering is $C_{\sigma(1)}$. We then compute a locally optimal assignment $x^*_{C_{\sigma(1)}}$ given the max-marginal at $C_{\sigma(1)}$:

$$x^*_{C_{\sigma(1)}} \in \arg\max_{c_{\sigma(1)}} \beta_{\sigma(1)}(c_{\sigma(1)}).$$

Then for future clique nodes $C_{\sigma(i)}$:

$$x^*_{C_{\sigma(i)}} \in \arg \max_{c_{\sigma(i)} \,:\, c_{\sigma(i)}[C_{\sigma(<i)}] = x^*_{C_{\sigma(<i)}}} \beta_{\sigma(i)}(c_{\sigma(i)}).$$

7 Linear Programming Relaxations

Consider a discrete PGM, with energy:

$$E(x) = \sum_{j=1}^{k} E_j(x_{C_j}).$$

Then the MAP assignment problem is simply: $\min_x E(x)$. This can be restated as the following integer program:

$$\begin{aligned}
\inf_{\{\mu_{C_j}(x_{C_j})\}} \quad & \sum_{j=1}^{k} \sum_{x_{C_j}} E_j(x_{C_j}) \mu_{C_j}(x_{C_j}) \\
\text{s.t. } & \mu_{C_j}(x_{C_j}) \in \{0, 1\} \\
& \sum_{x_{C_j}} \mu_{C_j}(x_{C_j}) = 1 \\
& \sum_{x_{C_i \setminus S_{ij}}} \mu_{C_i}(x_{C_i}) = \sum_{x_{C_j \setminus S_{ij}}} \mu_{C_j}(x_{C_j}), \quad C_i \cap C_j \neq \varnothing.
\end{aligned} \tag{1}$$

This restates the MAP problem in terms of binary local assignment variables $\{\mu_{C_j}(x_{C_j})\}$, so that for each C_j, there is only one value $x_{C_j}^{(j)}$ for which $\mu_{C_j}(x_{C_j}^{(j)}) = 1$. The local assignment variables are moreover locally consistent in the sense that they agree on any intersection of clique scopes. It is a simple exercise to see that this is just a restating of the MAP assignment problem.

Proposition 13 *Any feasible assignment to the integer variables $\{\mu_{C_j}(x_{C_j})\}$ that satisfy the constraints of the integer program in* (1) *corresponds to a single assignment $x_1, \ldots, x_p$ to the variables $X_1, \ldots, X_p$.*

This restating of the MAP problem as an integer program in and by itself is not helpful: both the number of integer variables, as well as the number of constraints seem fairly large, and the IPs in general are computationally intractable.

The key advantage is that we could consider the following simple linear programming (LP) relaxation of the above IP:

$$\begin{aligned}
\inf_{\{\mu_{C_j}(x_{C_j})\}} \quad & \sum_{j=1}^{k} \sum_{x_{C_j}} E_j(x_{C_j}) \mu_{C_j}(x_{C_j}) \\
\text{s.t. } & \mu_{C_j}(x_{C_j}) \geq 0 \\
& \sum_{x_{C_j}} \mu_{C_j}(x_{C_j}) = 1 \\
& \sum_{x_{C_i \setminus S_{ij}}} \mu_{C_i}(x_{C_i}) = \sum_{x_{C_j \setminus S_{ij}}} \mu_{C_j}(x_{C_j}), \quad C_i \cap C_j \neq \varnothing.
\end{aligned} \tag{2}$$

Here, we simply relax the constraint that the local variables $\{\mu_{C_j}(x_{C_j})\}$ be binary variables, and instead relax them to take values in the interval $[0, 1]$. We will revisit the structure of this LP in greater detail when we study approximate and variational inference, but for now, note that the solution to the LP program then no longer need specify a unique assignment x to the PGM variables, so that what we have instead are so-called pseudo-max-marginals.

But it is nonetheless still *possible* for the LP to provide an integer solution, and when it does, it exactly corresponds to a MAP assignment. We then say that the LP relaxation is tight. There is a large literature on conditions under which any LP relaxation is tight, so we won't discuss it at length there.

8 Local Search, Graph Cuts

As discussed in the previous section, the MAP assignment problem is essentially a combinatorial optimization problem, specifically an integer linear program. While intractable, there are many powerful heuristic approaches for solving this, including local search based methods, such as branch and bound. These in general do not leverage the graphical model structure.

There is however one class of local search methods that do use the graphical model structure, and scale even to very large-scale PGMs, and are based on graph cuts.

We will be using the notion of a graph cut. Here, suppose we have a weighted graph (V, E, W), where $W(s, t)$ is the weight of any edge $(s, t) \in E$. Then, a graph cut is a partition V_1, V_2 of the set of nodes V, and the cost of this partition is equal to the weight of the cut edges:

$$\text{cost}(V_0, V_1) = \sum_{s \in V_0, t \in V_1} W(s, t).$$

The minimal cut is a graph cut with the least cost. When the edge weights are non-negative, such a minimal graph cuts can be computed in polynomial time.

How do we relate such graph cuts to PGMs and MAP assignments? Consider for now the case where the variables are all binary, and the following "Potts Model" pairwise UGM energy, with factors of size at most two:

$$E(x) = \sum_{s \in V} \theta_s(x_s) + \sum_{(s,t) \in E} \theta_{st} \mathbb{I}[x_s \neq x_t].$$

Any graph cut partition V_0, V_1 of the set of nodes could be viewed as an assignment x to the set of variables $\mathcal{X}$: $x[V_0] = 0$, and $x[V_1] = 1$. And suppose we introduce two nodes Z_0 and Z_1 where we connect all nodes X_s to Z_i with an edge of weight $\theta_s(1 - i)$. And any pair of adjacent nodes X_s, X_t with an edge of weight θ_{st}. Now consider any graph cut partition V_0, V_1 of $V \cup \{Z_0, Z_1\}$, where we restrict $Z_0 \in V_0, Z_1 \in V_1$. Since $Z_0 \in V_0$, then for all $X_t \in V_1$, edges (X_t, Z_0) will be cut, with cost $\theta_t(1)$. And for all $X_s \in V_0$, edges (X_s, Z_0) will be cut, with cost $\theta_s(0)$. And if (X_s, X_t) lie in different components of the partition, the edge (X_s, X_t) will be cut, with cost θ_{st}. It can be seen that this is precisely the energy of the assignment $x[V_1, V_2]$. Thus, minimizing the graph cut cost and minimizing the energy are equivalent.

Now, we can always ensure that all edge weights are non-negative by simply translating the weights:

$$\begin{aligned}\theta_s &\leftarrow \theta_s - \min_{x_s} \theta_s(x_s)\\ \theta_{st} &\leftarrow \theta_{st} - \min_{(s,t)\in E} \theta_{st}.\end{aligned}$$

These translations only change the energy by a constant (i.e. not a function of the configuration x), and hence do not change the MAP assignment. We can thus solve the resulting minimal graph cut problem, and hence the MAP problem, in polynomial time.

While the above algorithm was specialized to the case of Potts models, this can be generalized to the more general case of discrete UGMs, where we iteratively specify min-cut problems to perform *local search* improvements to the MAP assignment (Boykov, Veksler, Zabih, 2001). While these are however not guaranteed to provide the global MAP assignment, they form one of the most popular classes of approximate inference algorithms for computing MAP assignments in discrete UGMs.