Variable elimination is probably the single most important algorithm to understand for manipulating graphical models, because

- it solves the most important problem, “inference”, computing the conditional distribution over one set of variables given assignments to another;
- it works for any directed or undirected graphical model over discrete variables;
- it’s easy to understand and re-derive;
- with the right elimination order, it’s as efficient as any other method that works for arbitrary probability tables (that is, that relies only on conditional independencies and not the numeric values of the probability distribution).

1 Moralization

VE can be done on either directed or undirected graphs but it’s convenient to consider only the undirected (Markov random field) case, and for problems on directed graphs (Bayes’ nets), convert the BN net to an MRF prior to elimination. The conversion process is called moralization, since it involves the “marriage” of a child’s parents.

For a directed graphical model, let $G$ be the conditional independence graph and $P$ the probability distribution. Let $G^m$ and $P'$ be the same after moralization. Moralization exactly maintains the probability distribution: $P' = P$. But it may weaken the correspondence between the independence graph and the distribution: although $G^m$ remains an I-map of $P$, some of the conditional independencies encoded in $G$ may be missing from $G^m$.

Given the directed graph $G$, we know that $P$ factors
\[ P(x) = \prod_{x_i \in V} P_{X_i}(x_i|\text{pa}(x_i)). \]

And we know that \( P' \) must factor

\[ P'(x) = \prod_i \Phi_i(x) \]

The correspondence \( P'(x) = P(x) \) is achieved by creating a potential for every family in \( G \):

\[ \Phi_{X_i}(x) = P_{X_i}(x_i|\text{pa}(x_i)). \]

By the Hammersley-Clifford theorem, \( G^m \) is an I-map of \( P \) if the domain of every potential function is a (not necessarily maximal) clique in \( G^m \), or equivalently, if every family in \( G \) is a clique in \( G^m \). Thus, two equivalent methods of constructing the edges for \( G^m = (V, E^m) \):

- \( E^m \) is the union of cliques over all families of \( G \);
- \( E^m \) is \( E \) after adding edges between the parents of each variable (marrying the parents to create cliques).

\( G^m \) may contain cliques that do not correspond to families of \( G \); there are no corresponding potentials in the factorization of \( P' \).

Here’s an example of moralization:
Notice that $G^m$ has lost numerous conditional independencies: $E \perp F, F \perp G, A \perp C$, etc.

### 2 Elimination Algorithm

Variable elimination works by eliminating variables from an undirected graphical model one-by-one, until only the variables of interest are left. As each (unobserved) variable $V_i$ is eliminated, the set of potentials involving $V_i$ is replaced by a single new potential $\Psi_{V_i}$ constructed by marginalizing over $V_i$. Graphically, this corresponds to eliminating the variable and its edges, and fully connecting its neighbors - since with the elimination of $V_i$ they are generally no longer conditionally independent.

The input to variable elimination:

- $V$, a set of variables;
- $\Omega = \{\Phi\}$, a set of potential functions over $V$;
- $O = \{\langle X_j = x_j \rangle\}$, observations of a subset of $V$ (possibly empty);
- $\pi$, the elimination order, an ordered subset of $V$; for simplicity, $\pi$ is assumed to include the observation variables $O$. 

\[ P = P_B P_C P_E P_F P_G \]
\[ P_{D|EFG} P_{A|BCD} \]

\[ P = \Phi_B \Phi_C \Phi_E \Phi_F \Phi_G \]
\[ \Phi_{DEFG} \Phi_{ABCD} \]
The output:

• $\Phi_Z$, a table over $Z = V - \pi$, the set of variables not eliminated. Each entry $\Phi_Z(z)$ is the joint probability of $Z = z$ and the observations, marginalized over the eliminated variables.

$$
\Phi_Z = \text{ELIMINATE}(V, \Omega = \{\Phi\}, \pi = V_1 \ldots V_m, O = \{(X_j = x_j)\})
$$

for $V_i$ in $\pi$

$\Omega_i = \{\Phi \in \Omega | V_i \in \text{domain}(\Phi)\}$ // potentials involving $V_i$

if $\langle V_i = v \rangle \in O$

// $V_i$ is observed: create new potentials by fixing $V_i$

for $\Phi \in \Omega_i$

$N = \text{domain}(\Phi) - \{V_i\}$

$\Phi' = \text{new potential over variables } N$

for each config $n$ in $N$

$\Phi'(n) = \Phi(n, v)$

$\Omega = \Omega \cup \{\Phi'\}$

else

// compute new potential by marginalizing over $V_i$

$N = \bigcup_{\Phi \in \Omega_i} \text{domain}(\Phi) - \{V_i\}$ // neighbors of $V_i$

$\Phi' = \text{new potential over variables } N$

for each config $n$ in $N$

$\Phi'(n) = \sum_{v \in V_i} \prod_{\Phi \in \Omega_i} \Phi(n, v)$

$\Omega = \Omega \cup \{\Phi'\}$

// eliminate $V_i$ and its potentials

$V = V - \{V_i\}$

$\Omega = \Omega - \Omega_i$

// compute table over remaining variables

$\Phi_Z = \text{new potential over remaining variables } V$

for each config $x$ in $V$

$\Phi_Z(x) = \prod_{\psi \in \Omega} \Phi(x)$

return $\Phi_Z$
3 Example

The factorization $P = P_A P_B | A P_C | A P_D | A B P_E | B C P_F | E$ with directed graph

![Directed Graph](image)

after moralization has factorization $P = \Phi_A \Phi_{AB} \Phi_{AC} \Phi_{ABD} \Phi_{BCE} \Phi_{EF}$ and graph

![Moralized Graph](image)

Suppose the goal is to compute $P_{C|D|F=1}$; this is accomplished by first computing the joint $P_{C|D|F=1}$ using VE, by eliminating $A, B, E$ and $F$. After choosing an elimination order (here, arbitrarily, $FEAB$), VE gives a table of over the remaining variables:

$$\Phi_{CD} = \text{ELIMINATE} (\{ABCDEF\}, \{\Phi_A \Phi_{AB} \Phi_{AC} \Phi_{ABD} \Phi_{BCE} \Phi_{EF}\}, \{FEAB\}, \{(F = 1)\})$$

and then
\[ P(C = c, D = d|F = 1) = \frac{\Phi_{CD}(c, d)}{\sum_{cd} \Phi_{CD}(c, d)} \]

The intermediate stages of variable elimination are as follows:
<table>
<thead>
<tr>
<th>Var. Neigh New factorization</th>
<th>New graph</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[diagram]</td>
</tr>
<tr>
<td></td>
<td>$P = \Phi_A \Phi_{AB} \Phi_{AC} \Phi_{ABD} \Phi_{BCE} \Phi_{EF}$</td>
</tr>
<tr>
<td>$F$ $E$</td>
<td>[diagram]</td>
</tr>
<tr>
<td></td>
<td>$P = \Phi_A \Phi_{AB} \Phi_{AC} \Phi_{ABD} \Phi_{BCE} \Phi_{E}^1$</td>
</tr>
<tr>
<td></td>
<td>$\Phi_{E}^1(e) = \Phi_{EF}(e, 1)$</td>
</tr>
<tr>
<td>$E$ $B, C$</td>
<td>[diagram]</td>
</tr>
<tr>
<td></td>
<td>$P = \Phi_A \Phi_{AB} \Phi_{AC} \Phi_{ABD} \Phi_{BC}^2$</td>
</tr>
<tr>
<td></td>
<td>$\Phi_{BC}^2(bc) = \sum_e \Phi_{BCE}(bce) \Phi_{E}^1(e)$</td>
</tr>
<tr>
<td>$A$ $B, C, D$</td>
<td>[diagram]</td>
</tr>
<tr>
<td></td>
<td>$P = \Phi_{BC}^2 \Phi_{BCD}^3$</td>
</tr>
<tr>
<td></td>
<td>$\Phi_{BCD}^3(bcd) = \sum_a \Phi_A(a) \Phi_{AB}(ab)$</td>
</tr>
<tr>
<td></td>
<td>$\Phi_{AC}(ac) \Phi_{ABD}(abd)$</td>
</tr>
<tr>
<td>$B$ $C, D$</td>
<td>[diagram]</td>
</tr>
<tr>
<td></td>
<td>$P = \Phi_{CD}^4$</td>
</tr>
<tr>
<td></td>
<td>$\Phi_{CD}^4(cd) = \sum_b \Phi_{BC}^2(bc) \Phi_{BCD}^3(bcd)$</td>
</tr>
</tbody>
</table>
4 Computational complexity and induced width

The computational work done in VE is primarily the calculation of new potentials. A potential over $n$ binary variables is a table of $2^n$ values, each computed by summing a product of potentials over an eliminated variable. With proper implementation, the cost of computing the potential product is a constant rather than growing linearly with the number of potentials (Q: why?), so the total computational cost of computing a new potential over $n$ variables is $2^n + 1$.

If $c$ is the maximum number of variables in any potential under a particular elimination order $\pi$, it is clear the the time complexity of variable elimination is $O(|V|2^{c+1})$. However finding the elimination order that minimizes $c$ is an NP-complete problem!

Define the induced width of $G$ under $\pi$, $w_\pi$, to be the maximum size of the neighbor set $N$ in the execution of ELIMINATE for an ordering $\pi$ over all the variables in the graph (total elimination). Define the elimination width to be the minimum of $w_\pi$ over all elimination orders. It is not hard to see that the elimination width of an unconnected graph is 0, of any tree is 1 (achieving this requires the right choice of elimination order), and of a loop is 2.

There are a variety of different ways of defining graph complexity that are all equivalent; a graph’s elimination width can be shown to be equal to its tree width, a complexity measure related to the decomposition of the graph into a tree of cliques that we will explore as part of the junction tree algorithm.