What’s next

- Thus far: Variable elimination
  - (Often) Efficient algorithm for inference in graphical models

- Next: Understanding complexity of variable elimination
  - Will lead to cool junction tree algorithm later
Complexity of variable elimination – Graphs with loops

Moralize graph:
Connect parents into a clique and remove edge directions

Connect nodes that appear together in an initial factor

Eliminating a node – Fill edges

Eliminate variable
Connect neighbors
The induced graph $I_{\mathcal{F}}$ for elimination order $\mathcal{E}$ has an edge $X_i - X_j$ if $X_i$ and $X_j$ appear together in a factor generated by VE for elimination order $\mathcal{E}$ on factors $\mathcal{F}$.

Different elimination order can lead to different induced graph.

Elimination order: 
{G,C,D,S,I,L,H,J}
Induced graph and complexity of VE

- Structure of induced graph encodes complexity of VE!!!
- **Theorem:**
  - Every factor generated by VE subset of a maximal clique in $I_F$ corresponds to a factor generated by VE
  - Induced width (or treewidth)
    - Size of largest clique in $I_F$ minus 1
    - Minimal induced width – induced width of best order

Elimination order: \{C,D,I,S,L,H,J,G\}

Example: Large induced-width with small number of parents

Compact representation ⇒ Easy inference 🎉
Finding optimal elimination order

- **Theorem**: Finding best elimination order is NP-complete:
  - Decision problem: Given a graph, determine if there exists an elimination order that achieves induced width ≤ K

- **Interpretation**:
  - Hardness of finding elimination order in addition to hardness of inference
  - Actually, can find elimination order in time exponential in size of largest clique – same complexity as inference

Elimination order: {C,D,I,S,L,H,J,G}

Induced graphs and chordal graphs

- **Chordal graph**:
  - Every cycle $X_1 - X_2 - \ldots - X_k - X_1$ with $k \geq 3$ has a chord
  - Edge $X_i - X_j$ for non-consecutive $i$ & $j$

- **Theorem**:
  - Every induced graph is chordal

- “Optimal” elimination order easily obtained for chordal graph
Chordal graphs and triangulation

- **Triangulation**: turning graph into chordal graph
- **Max Cardinality Search**:
  - Simple heuristic
  - Initialize unobserved nodes $X$ as unmarked
  - For $k = |X|$ to 1
    - $X \leftarrow$ unmarked var with most marked neighbors
    - $\angle(X) \leftarrow k$
    - Mark $X$
- **Theorem**: Obtains optimal order for chordal graphs
- Often, not so good in other graphs!

Minimum fill/size/weight heuristics

- Many more effective heuristics
  - see reading
- **Min (weighted) fill heuristic**
  - Often very effective
- Initialize unobserved nodes $X$ as unmarked
- For $k = 1$ to $|X|
  - $X \leftarrow$ unmarked var whose elimination adds fewest edges
  - $\angle(X) \leftarrow k$
  - Mark $X$
  - Add fill edges introduced by eliminating $X$
- Weighted version:
  - Consider size of factor rather than number of edges
Choosing an elimination order

- Choosing best order is NP-complete
  - Reduction from MAX-Clique
- Many good heuristics (some with guarantees)
- Ultimately, can’t beat NP-hardness of inference
  - Even optimal order can lead to exponential variable elimination computation
- In practice
  - Variable elimination often very effective
  - Many (many many) approximate inference approaches available when variable elimination too expensive
  - Most approximate inference approaches build on ideas from variable elimination

Most likely explanation (MLE)

- Query: \[ \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n \mid e) \]

- Using defn of conditional probs:
  \[ \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n \mid e) = \arg\max_{x_1, \ldots, x_n} \frac{P(x_1, \ldots, x_n, e)}{P(e)} \]

- Normalization irrelevant:
  \[ \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n \mid e) = \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n, e) \]
Max-marginalization

Example of variable elimination for MLE – Forward pass
Example of variable elimination for MLE – Backward pass

MLE Variable elimination algorithm – Forward pass

- Given a BN and a MLE query $\max_{x_1,\ldots,x_n} P(x_1,\ldots,x_n,e)$
- Instantiate evidence $E=e$
- Choose an ordering on variables, e.g., $X_1, \ldots, X_n$
- For $i = 1$ to $n$, if $X_i \notin E$
  - Collect factors $f_1,\ldots,f_k$ that include $X_i$
  - Generate a new factor by eliminating $X_i$ from these factors
    
    $$g = \max_{x_i} \prod_{j=1}^{k} f_j$$

- Variable $X_i$ has been eliminated!
MLE Variable elimination algorithm
– Backward pass

- \{x_1^*, \ldots, x_n^*\} will store maximizing assignment
- For \( i = n \) to 1, If \( X_i \notin E \)
  - Take factors \( f_1, \ldots, f_k \) used when \( X_i \) was eliminated
  - Instantiate \( f_1, \ldots, f_k \) with \( \{x_{i+1}^*, \ldots, x_n^*\} \)
    - Now each \( f_j \) depends only on \( X_i \)
  - Generate maximizing assignment for \( X_i \):
    \[
    x_i^* \in \arg\max_{x_i} \prod_{j=1}^{k} f_j
    \]

What you need to know about VE

- Variable elimination algorithm
  - Eliminate a variable:
    - Combine factors that include this var into single factor
    - Marginalize var from new factor
  - Cliques in induced graph correspond to factors generated by algorithm
  - Efficient algorithm (“only” exponential in induced-width, not number of variables)
    - If you hear: “Exact inference only efficient in tree graphical models”
    - You say: “No!! Any graph with low induced width”
    - And then you say: “And even some with very large induced-width” (special recitation)
  - Elimination order is important!
    - NP-complete problem
    - Many good heuristics
  - Variable elimination for MLE
    - Only difference between probabilistic inference and MLE is “sum” versus “max”
What if I want to compute $P(X_i|x_0,x_{n+1})$ for each $i$?

Compute:

$P(X_i | x_0, x_{n+1})$

Variable elimination for each $i$?

Variable elimination for every $i$, what's the complexity?

Reusing computation

Compute:

$P(X_i | x_0, x_{n+1})$
Cluster graph

- **Cluster graph**: For set of factors $F$
  - Undirected graph
  - Each node $i$ associated with a cluster $C_i$
  - *Family preserving*: for each factor $f_j \in F$,
    - $\exists$ node $i$ such that scope[$f_j$] $\subseteq C_i$
  - Each edge $i - j$ is associated with a separator $S_{ij} = C_i \cap C_j$

Factors generated by VE

- Elimination order:
  - $\{C, D, I, S, L, H, J, G\}$
Cluster graph for VE

- **VE generates cluster tree!**
  - One clique for each factor used/generated
  - Edge $i \rightarrow j$, if $f_i$ used to generate $f_j$
  - "Message" from $i$ to $j$ generated when marginalizing a variable from $f_i$
  - Tree because factors only used once

- **Proposition:**
  - "Message" $\delta_{ij}$ from $i$ to $j$
  - $\text{Scope}[\delta_{ij}] \subseteq S_{ij}$

---

Running intersection property

- **Running intersection property (RIP)**
  - Cluster tree satisfies RIP if whenever $X \in C_i$ and $X \in C_j$ then $X$ is in every cluster in the (unique) path from $C_i$ to $C_j$

- **Theorem:**
  - Cluster tree generated by VE satisfies RIP
Constructing a clique tree from VE

- Select elimination order
- Connect factors that would be generated if you run VE with order
- Simplify!
  - Eliminate factor that is subset of neighbor

Find clique tree from chordal graph

- Triangulate moralized graph to obtain chordal graph
- Find maximal cliques
  - NP-complete in general
  - Easy for chordal graphs
  - Max-cardinality search
- Maximum spanning tree finds clique tree satisfying RIP!!!
  - Generate weighted graph over cliques
  - Edge weights \((i,j)\) is separator size – \(|C_i \cap C_j|\)
Clique tree & Independencies

- **Clique tree (or Junction tree)**
  - A cluster tree that satisfies the RIP
- **Theorem**:
  - Given some BN with structure $G$ and factors $F$
  - For a clique tree $T$ for $F$ consider $C_i - C_j$ with separator $S_{ij}$:
    - $X$ – any set of vars in $C_i$ side of the tree
    - $Y$ – any set of vars in $C_j$ side of the tree
  - Then, $(X \perp Y \mid S_{ij})$ in BN
  - Furthermore, $I(T) \subseteq I(G)$

Variable elimination in a clique tree 1

- **Clique tree for a BN**
  - Each CPT assigned to a clique
  - Initial potential $\pi_0(C_i)$ is product of CPTs
Variable elimination in a clique tree 2

- VE in clique tree to compute P(X_i)
  - Pick a root (any node containing X_i)
  - Send messages recursively from leaves to root
    - Multiply incoming messages with initial potential
    - Marginalize vars that are not in separator
  - Clique ready if received messages from all neighbors

Belief from message

- Theorem: When clique C_i is ready
  - Received messages from all neighbors
  - Belief π_i(C_i) is product of initial factor with messages:
Choice of root

- Message does not depend on root!!!

Root: node 5

Root: node 3

“Cache” computation: Obtain belief for all roots in linear time!!

Shafer-Shenoy Algorithm
(a.k.a. VE in clique tree for all roots)

- Clique $C_i$ ready to transmit to neighbor $C_j$ if received messages from all neighbors but $j$
  - Leaves are always ready to transmit
- While $\exists C_i$ ready to transmit to $C_j$
  - Send message $\delta_{i \rightarrow j}$
- Complexity: Linear in # cliques
  - One message sent each direction in each edge
- Corollary: At convergence
  - Every clique has correct belief
Calibrated Clique tree

- Initially, neighboring nodes don’t agree on “distribution” over separators
- **Calibrated clique tree:**
  - At convergence, tree is calibrated
  - Neighboring nodes agree on distribution over separator

Answering queries with clique trees

- Query within clique
  - Incremental updates – Observing evidence $Z=z$
    - Multiply some clique by indicator $1(Z=z)$

- Query outside clique
  - Use variable elimination!
Message passing with division

- Computing messages by multiplication:

- Computing messages by division:

Lauritzen-Spiegelhalter Algorithm (a.k.a. belief propagation)

- Initialize all separator potentials to 1
  - $\mu_{ij} \leftarrow 1$
- All messages ready to transmit
- While $\exists \delta_{i \rightarrow j}$ ready to transmit
  - $\mu_{ij}' \leftarrow$
  - If $\mu_{ij}' \neq \mu_{ij}$
    - $\delta_{i \rightarrow j} \leftarrow$
    - $\pi_{j} \leftarrow \pi_{j} \times \delta_{i \rightarrow j}$
    - $\mu_{ij} \leftarrow \mu_{ij}'$
    - $\forall$ neighbors $k$ of $j, k \neq i, \delta_{j \rightarrow k}$ ready to transmit
- Complexity: Linear in # cliques
  - for the “right” schedule over edges (leaves to root, then root to leaves)
- Corollary: At convergence, every clique has correct belief
VE versus BP in clique trees

- VE messages (the one that multiplies)
- BP messages (the one that divides)

Clique tree invariant

- Clique tree potential:
  - Product of clique potentials divided by separators potentials

- Clique tree invariant:
  - $P(X) = \pi_T(X)$
Belief propagation and clique tree invariant

- **Theorem**: Invariant is maintained by BP algorithm!

- BP reparameterizes clique potentials and separator potentials
  - At convergence, potentials and messages are marginal distributions

Subtree correctness

- **Informed message** from i to j, if all messages into i (other than from j) are informed
  - Recursive definition (leaves always send informed messages)

- **Informed subtree**: All incoming messages informed

- **Theorem**: Potential of connected informed subtree \( T' \) is marginal over scope[\( T \)]

- **Corollary**: At convergence, clique tree is *calibrated*
  - \( \pi_i = P(\text{scope}[\pi_i]) \)
  - \( \mu_j = P(\text{scope}[\mu_j]) \)
Clique trees versus VE

- Clique tree advantages
  - Multi-query settings
  - Incremental updates
  - Pre-computation makes complexity explicit

- Clique tree disadvantages
  - Space requirements – no factors are “deleted”
  - Slower for single query
  - Local structure in factors may be lost when they are multiplied together into initial clique potential

Clique tree summary

- Solve marginal queries for all variables in only twice the cost of query for one variable
- Cliques correspond to maximal cliques in induced graph
- Two message passing approaches
  - VE (the one that multiplies messages)
  - BP (the one that divides by old message)
- Clique tree invariant
  - Clique tree potential is always the same
  - We are only reparameterizing clique potentials
- Constructing clique tree for a BN
  - from elimination order
  - from triangulated (chordal) graph
- Running time (only) exponential in size of largest clique
  - Solve exactly problems with thousands (or millions, or more) of variables, and cliques with tens of nodes (or less)