Outline of Today’s Lecture

• Chordal graph theory
• Elimination and chordality
• Recognizing chordal graphs
• How to triangulate a graph (heuristics)
• Properties of conditional independence
• Markov properties on graphs
Books and Sources for Today

- Jordan: Chapters 17.
- Any good graph theory text.
Class Road Map

- L1: Tues, 3/28: Overview, GMs, Intro BNs.
- L2: Thur, 3/30: semantics of BNs + UGMs
- L3: Tues, 4/4: elimination, probs, chordal I
- L4: Thur, 4/6: chrdal, sep, decomp, elim
- L5: Tue, 4/11: chdl/elim, mcs, triang, ci props.
- L6: Thur, 4/13
- L7: Tues, 4/18
- L8: Thur, 4/20
- L9: Tue, 4/25
- L10: Thur, 4/27

- L11: Tues, 5/2
- L12: Thur, 5/4
- L13: Tues, 5/9
- L14: Thur, 5/11
- L15: Tue, 5/16
- L16: Thur, 5/18
- L17: Tues, 5/23
- L18: Thur, 5/25
- L19: Tue, 5/30
- L20: Thur, 6/1: final presentations
Announcements

• If you see a typo, please tell me during lecture
  – everyone will then benefit.
  – note, corrected slides will go on web.
• READING: Chapter 3 & 17 in Jordan’s book
• Lauritzen chapters 1-3 (on reserve in library)
• Check out CSE590AI this week (Weds, CSE609, 3:30-4:20)
  – *Compiling Relational Bayesian Networks for Exact Inference*, by Mark Chavira, Adnan Darwiche, and Manfred Jaeger
  – This is a “search” based method for inference of relational models, that pre-compiles such graph into logic equations. It has an implicit value-specific junction tree in it. We will be covering search based methods for inference in the upcoming weeks discussing what conditions are required for them to help.
• Reminder: TA discussions and office hours:
  – Office hours: Thursdays 3:30-4:30, Sieg Ground Floor Tutorial Center
  – Discussion Sections: Fridays 9:30-10:30, Sieg Ground Floor Tutorial Center Lecture Room
Summary of Last Time

- Junction trees, collection of cliques that satisfy r.i.p.
- Decomposable: proper decomposition \((A, B, C)\) and \(G[AC]\) \(G[BC]\) decomposable.
- **Thm:** triangulated \(\equiv\) decomposable \(\equiv\) min seps are complete.
- Min seps, every node connected to both sides.
- Examples of decomposable models (updated decomposition tree)
- The ability to use separators in a factorization.
- \(\exists\) multiple JTs for a set of maxcliques
- **Thm:** JT of maxcliques exist iff \(G\) is decomposable
- R.i.p. order \(\Leftrightarrow\) well ordered sequence in JT (BFS, DFS).
Examples: decomposition tree and factorization

\[ p(A, B, C, D, E, F, G, H, I, J, K) = \frac{p(A, C, D, F)p(B, G, H)p(C, B, H)p(I, E, J)p(E, I, D)p(C, K, H)p(D, K, I)p(D, K, F, C)}{p(C, D, F)p(C, H)p(B, H)p(D, I)p(E, I)p(C, K)p(D, K)} \]
Graph, decomposition tree, and junction tree
How can we tell if G is chordal?

**Theorem.** *Elimination yields a triangulated graph*

- **Consequences:** It means that if we sum out variables from a product of factors, the extra edges that were added due to elimination might as well have been there to begin with.

**Proof:** Induction, base case single node, obvious. Assume holds for \( \leq N \) prove for \( N + 1 \).

- Eliminating a node results in graph with \( N \) nodes which, by induction, once elimination is done, has no chordless cycles.
- The elimination process itself does not form any additional chordless cycles since it joins all neighbors of nodes (forming complete set). \( \Box \)
How can we tell if G is chordal?

**Definition:** *Simplicial node:* A node $\alpha$ is simplicial if $\text{bd}(\alpha)$ is complete (i.e., all neighbors are connected).

Note: simplicial nodes are generalizations of leaves in a regular tree.

**Definition:** *eliminatable:* A graph $G$ is eliminatable if all nodes can be successfully eliminated without adding any fill-in edges.

**Theorem.** *Any eliminatable graph is triangulated.*

**Proof:** Base case, 1 node, obvious. Assume true for $N$, prove for $N + 1$. So, $G$ is an eliminatable graph with $N + 1$ nodes, so has at least one simplicial node called $a$. By induction, since $G[V \setminus \{a\}]$ is triangulated. But the addition of $a$ doesn’t add any chordless cycles (since simplicial). Therefore, $G$ is triangulated. □

- Note: as we learn more about triangulated graphs, you’ll see that we will use different properties in different circumstances to understand what advantages we have with triangulated graphs.
How can we tell if $G$ is chordal?

- this gives us an easy way to see if a graph is chordal, i.e., can we find an elimination ordering that does not produce any fill-in edges.
How can we tell if $G$ is chordal?

**Definition:** A *perfect elimination ordering* is an elimination ordering on a graph $G = (V, E)$ that does not produce any additional fill-in edges.

- Note: previous theorem says that for a graph to be triangulated, there has to be at least one perfect elimination ordering (but not all orderings need be perfect).
- Consider a given perfect ordering $\sigma = (v_1, \ldots, v_n)$. At each step, a node $v_i$ and its neighbors with higher number form a clique, but is it a maxclique? Note: Once eliminated, a node (say $v_i$) can’t be involved in any future eliminations, so for a node and neighbors to be a maxclique, it can’t be a subset of any previous node/neighbors during elimination.
- But how do we find a perfect elimination order if it exists? Is this an NP-complete problem? (i.e., detecting chordal graphs)?
How can we recognize chordal graphs?

- Many combinatorial problems are NP-complete optimization problems (unfortunately), but there are a small splinter of interesting algorithms that are still provably polynomial. This is one of them.
- Can we also identify the maxcliques in a triangulated graph? Yes.
- But first, consider in general the *clique number* of a graph:
  \[ \omega(G) = \max\{|V'| : V' \subseteq V \text{ and } V' \text{ is a clique in } G\} \]

- We cannot compute \( \omega(G) \) efficiently unless P=NP. Therefore, finding the maxcliques is an NP-complete problem. However, if the graph is triangulated, things change.
- Maximum cardinality search (Tarjan,Yannakakis,1984) is a simple greedy algorithm that provably detects if the graph is chordal, and optimally returns the cliques in r.i.p. order. It can be made to run in \( O(|V| + |E|) \) time using a Fibonacci heap.
Maximum Cardinality Search (MCS)

- Maximum Cardinality Search (MCS) algorithm.

**Input:** A graph $G = (V, E)$, $n = |V|$

**Output:** An MCS ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of $V$

for $i = |V|$ downto 1 do

chose a vertex $v \in V$ with max number of prev. numbered neighbors;

number $v$ with the label $i$;

$\pi_i = \text{ne}(v) \cap \text{prev. numbered nodes}$;

if $\pi_i$ is not complete then

Output Not Chordal and return

end

$\sigma(i) \leftarrow v$;

$V \leftarrow V \setminus \{v\}$;

end
Maximum Cardinality Search (MCS)

- To find cliques in r.i.p. order, we can do:

  **Input:** \( \sigma, \{\pi_i\}_{i=1}^{n} \) from MCS assuming graph is chordal
  **Output:** r.i.p. clique ordering
  **for** \( i = 1 \) **to** \(|V|\) **do**
    **if** \(|\pi_i| \geq |\pi_{i+1}| \) **or** \( i = |V| \) **then**
    output clique of nodes in \( \pi_i \cup \{\sigma(i)\} \)
  **end**
  **end**

- So if the \( \pi \)'s don’t grow (shrink or stay the same in cardinality), or we’re at the last \( \pi \), we’ve got a maxclique.
How can we recognize chordal graphs?

- MCS algorithm in action.
- What’s the difference between what happens on the left vs. on the right?

- MCS produces an ordering that, if graph is chordal, is a reverse perfect elimination ordering.
How can we recognize chordal graphs?

- Moreover, MCS can be used to triangulate the graph. If $\pi_i$ is not complete at any time (so that the graph is not triangulated) we just complete it (add fill-in edges so that $\pi_i$ is complete). Then, when done with MCS, we run the elimination algorithm using the reverse of the ordering produced by MCS. (Q: must we complete the cliques above to get a triangulated graph if we run elimination after?)
- If original graph is chordal, running MCS order in reverse will provide us a perfect elimination order.
- Complexity $O(|V| + |E|)$ using amortized analysis. Use Fibonacci heap to determine next node to label (i.e., the one that has maximum cardinality set of previously labeled neighbors).
MCS with Junction Tree

r.i.p. order:

**Definition:** A DAG is *perfect* if the parents of every node form a complete set.

- **examples:**

**Theorem:** Given a well-ordered (topological) perfect DAG $G = (V, E)$, with order $(v_1, \ldots, v_n)$, its undirected version (drop arrows) is chordal with $(v_n, \ldots, v_1)$ is a perfect elimination order.

- **Intuition:** Parent sets separate non-descendants (later nodes in the ordering) from the ancestors other than parents (a node’s parents form a separator in a decomposition with all but the child on the left, and the child and parents on the right, and this is true for all nodes).
_triangulation heuristics

- Many ways to triangulate a graph, all typically want to minimize clique size (to minimize number of rvs simultaneously in a table).
- Could run MCS and complete π’s, but this typically produces a poor triangulation. Finding optimal triangulation is NP-complete optimization problem (Arnborg, 1982), so we resort to heuristics, 3 are quite often used:
  1) \textit{Minimize fill-in}: during elimination, choose the next node to eliminate that requires the smallest number of additional fill-in edges.
  2) \textit{Minimum Degree}: (also called minimum size). At elimination time, choose the next node to eliminate that has the smallest edge degree.
  3) \textit{Minimum weight}: At elimination time, choose the next node to eliminate that has the smallest weight \( w(v) \), where \( w(v) = \prod_{w \in \text{ne}(v)} s(v) \) and where \( s(v) \) is the number of values \( v \) can take on (cardinality of the random variable).
Triangulation Heuristics

- The different heuristics can make different decisions. Consider the following graph. For each heuristic, which node is eliminated first? Min-fill, Min-Size, Min-weight.

![Triangulation Heuristics Diagram]
Are all trees of maxcliques JTs?

- There are many possible trees of cliques, when considering a tree from the graph of cliques.

  ![Graph with cliques](image)

  Doesn’t satisfy r.i.p.

  ![Graph with cliques](image)

  Does satisfy r.i.p.

- What is the difference? In right case, cardinality of separator is larger than on the right.

- Weight of a tree of cliques $w(T)$ is the sum of cardinalities of the separator sets.
Are all trees of maxcliques JTs?

**Theorem:** A tree of cliques $T$ is a Junction tree iff it is a maximal spanning tree on the graph of cliques, with edge weights set according to the cardinality of the separator sets between the two cliques.

- So we can use Kruskal’s or Prim’s algorithm for MST to find the JT (that is if we don’t want to use MCS to get r.i.p. order). Prim is $O(|E| + |V| \log |V|)$ again using Fibonacci heap.
Are all trees of maxcliques JTs?

- Consider a tree of $M$ cliques with cliques $C_i$ and separators $S_j$, and consider node $x_k$.
- Num times $x_k$ appears in separator sets is $\sum_{j=1}^{M-1} 1\{x_k \in S_j\}$
- Num times $x_k$ appears in maxcliques is $\sum_{i=1}^{M} 1\{x_k \in C_i\}$
- In a tree, we have that $\sum_{j=1}^{M-1} 1\{x_k \in S_j\} \leq \sum_{i=1}^{M} 1\{x_k \in C_i\} - 1$, since if $x_k \in S_j$, then $x_k$ is also in the neighboring cliques of $S_j$.
- This becomes an equality when subgraph of $T$ induced by $x_k$ is a tree (this is the JT condition).
Are all trees of maxcliques JTs?

Total weight in tree \( w(T) = \sum_{j=1}^{M-1} |S_j| \)

\[ = \sum_{j=1}^{M-1} \sum_{k=1}^{N} 1\{x_k \in S_j\} \]

\[ = \sum_{k=1}^{N} \sum_{j=1}^{M-1} 1\{x_k \in S_j\} \]

\[ \leq \sum_{k=1}^{N} \left[ \sum_{j=1}^{M} 1\{x_k \in C_i\} - 1 \right] \]

\[ = \sum_{i=1}^{M} \sum_{k=1}^{N} 1\{x_k \in C_i\} - N \]

\[ = \sum_{i=1}^{M} |C_i| - N \]

• \( w(T) \) is maximized when we have equality on the right, and that happens when all nodes induce a tree in the graph of cliques (i.e., the JT property). Therefore, finding the maximum spanning tree in the graph of cliques forms a JT. This is a very standard way to get a JT in real inference procedures.

• But why do we care so much about JTs?
Junction Trees -> Factorization

- As we will see, the JT is formed either explicitly or implicitly whenever we wish to do exact probabilistic inference optimally, i.e., compute $p(x_F|x_E) = \sum_{x_H} p(x_F, X_H|X_E)$. This is even true for all the search based methods, a value-specific JT is formed (we will see this when we discuss time-space tradeoffs).
- The JT expresses things in terms of a decomposition, or factorization, where the separators in a JT correspond to the separators that are conditioned on, all of which are used when forming the decomposition of the distribution.
- But before we get there, we need to be a bit more formal about the semantics of MRFs and BNs, and of conditional independence (and factorization) in general. This is our next topic.
Properties of Conditional Independence

**Definition:** conditional independence: given r.v.s $X$, $Y$, and $Z$, we say that $X \perp Y|Z$ if $p(x, y|z) = p(x|z)p(y|z)$, for all $x, y, z$ when $p(z) > 0$. We can think of these as mass functions or densities (for continuous r.v.s). $X$, $Y$, or $Z$ could be scalars or vectors. $X \perp Y$ is *marginal independence*.

- Properties of CI (the ternary relation $X \perp Y|Z$).
  - C1) $X \perp Y|Z \Rightarrow Y \perp X|Z$. In other words, multiplication (influence) is commutative.
  - C2) $X \perp Y|Z$ and $U = h(X) \Rightarrow U \perp Y|Z$. Intuition: $U$ has less information than $X$, so it is not going to tell us any more about $Y$ than $X$ is, conditional or unconditional on $Z$.
  - C3) $X \perp Y|Z$ and $U = h(X) \Rightarrow X \perp Y|(Z, U)$. Knowing part of $X$ (i.e., $h(X)$) in addition to $Z$ will not tell us any more about $Y$ given $Z$.
  - C4) $X \perp Y|Z$ and $X \perp W|\{Y, Z\} \Rightarrow X \perp \{W, Y\}|Z$. Marginal case, $X \perp Y$ and $X \perp W|Y \Rightarrow X \perp \{W, Y\}$.
Properties of Conditional Independence

More Intuition:

C3) $X \perp Y|Z$ and $U = h(X)$ $\Rightarrow$ $X \perp Y|(Z, U)$.

Ex: $\{X_1, X_2\} \perp Y|Z \Rightarrow \{X_1, X_2\} \perp Y|\{Z, X_2\} \Rightarrow X_1 \perp Y|\{Z, X_2\}$. This means that if $X \perp \{Y, Z\}$ then $X \perp Y|Z$ and $X \perp Z|Y$!

C4) $X \perp Y|Z$ and $X \perp W|\{Y, Z\}$ $\Rightarrow$ $X \perp \{W, Y\}|Z$. Marginal case, $X \perp Y$ and $X \perp W|Y$ $\Rightarrow$ $X \perp \{W, Y\}$. We can view this using a graph:

Another property sometimes holds (i.e., when the distributions are positive) but not always, sort of like a converse to C3.

C5) $X \perp Y|Z$ and $X \perp Z|Y$ $\Rightarrow$ $X \perp \{Y, Z\}$
Properties of Conditional Independence

Example, when C5 does not hold: \( X = Y = Z, p(X = 1) = p(X = 0) = 1/2, \)
implying that \( p(X = Y = Z = 1) = p(X = Y = Z = 0) = 1/2, \) and all other
assignments have probability 0. But then,
• \( p(x|z) = p(x, z)/p(z) = (1/2)1\{x = z\}/(1/2) = 1\{x = z\}. \)
• \( p(y|z) = 1\{y = z\}. \)
• \( p(x, y|z) = p(x, y, z)/p(x) = (1/2)1\{x = y = z\}/(1/2) = 1\{x = y = z\}. \)
• but we’ve got
\( p(x, y|z) = 1\{x = y = z\} = 1\{x = y\}1\{y = z\} = p(x|z)p(y|z), \) so \( X \perp Y | Z \)
(and similarly \( X \perp Z | Y \).)
• But we do not have that \( X \perp \{Y, Z\}. \)
Properties of Conditional Independence

**Theorem:** When a density is strictly positive (i.e., $p(x, y, z) > 0, \forall x, y, z$, such as a Gaussian) then property C5 holds.

**Proof:** We’ve got that $p(x, y, z) > 0$ and $X \perp Y|Z$, and $X \perp Z|Y$. Then $\exists$ functions $k$, $\ell$, $g$, and $h$ such that:

$$p(x, y, z) = k(x, z)\ell(y, z) = g(x, y)h(y, z)$$

where $k, \ell, g, h$ are strictly positive. This means that:

$$g(x, y) = \frac{k(x, z)\ell(y, z)}{h(y, z)}$$

We fix $z = z_0$ where $z_0$ is arbitrary. Then $g(x, y) = \pi(x)f(y)$ where $\pi(x) = k(x, z_0)$ and $f(y) = \ell(y, z_0)/h(y, z_0) \forall z_0$.

Then, $p(x, y, z) = g(x, y)h(y, z) = \pi(x)(f(y)h(y, z))$, or that $X \perp \{Y, Z\}$. 
Properties of Conditional Independence

- Conditional independence and "knowledge", or "information". We can consider \( \perp \) as irrelevance, and \( | \) as knowing, so that \( X \perp Y|Z \) is "knowing Z (or having read Z), then knowing Y is irrelevant to knowing X".
- Note that conditional independence is related to conditional mutual information, or \( I(X;Y|Z) \), where:

\[
I(X;Y|Z) = H(X|Z) - H(X|Z,Y) = E_{p(x,y,z)} \log \frac{p(x,y|z)}{p(x|z)p(y|z)}
\]

- We have that \( X \perp Y|Z \iff I(X,Y|Z) = 0 \), and in a loose sense, the degree to which \( I(X,Y|Z) > 0 \) corresponds to the degree to which \( X \perp Y|Z \) is not true.
- A *semi-graphoid* is any algebraic structure satisfying axioms C1-C4. A *graphoid* also satisfies C5. For example, graph separation on undirected graphs satisfy C1-C4.
Properties of Conditional Independence

- Example graph separation properties.

C1) If $C$ separates $A$ from $B$ then $C$ separates $B$ from $A$.

C2) If $C$ separates $A$ from $B$, and $U \subseteq A$, then $C$ separates $U$ from $B$.

C3) If $C$ seps. $A$ from $B$, and $U \subseteq B$ then $C \cup U$ seps. $A$ from $B$.

C4) If $C$ seps. $A$ from $B$, and $B \cup D$ seps. $A$ from $D$, then $C$ seps $A$ from $B \cup D$. 
Markov Properties of Graphs

- Graphs have Markov properties (CI statements that they might imply). These properties might or might not correspond to each other. In order to deepen our knowledge of the semantics of MRFs and BNs, it is crucial to understand the relationship between these properties. There are 4 crucial properties for MRFs.
  
- (P) **pairwise markov property**, relative to $G = (V, E)$, if for any pair $(\alpha, \beta)$ of non-adjacent vertices, we have that $\alpha \perp \beta | V \setminus \{\alpha, \beta\}$.

- (L) **local Markov property**: relative to $G$, if for any $\alpha \in V$, we have that $\alpha \perp V \setminus cl(\alpha) | bd(\alpha)$.

- (G) **global Markov property**: relative to $G$, if for any triple $(A, B, S)$ of disjoint subsets of $V$ with $S$ separating $A$ from $B$ in $G$, we have that $A \perp B | S$