EE512A – Advanced Inference in Graphical Models

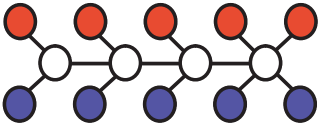
— Fall Quarter, Lecture 7 —

http://j.ee.washington.edu/~bilmes/classes/ee512a_fall_2014/

Prof. Jeff Bilmes

University of Washington, Seattle
Department of Electrical Engineering
http://melodi.ee.washington.edu/~bilmes

Oct 20th, 2014



Announcements

- Reading assignments, posted to our canvas announcements page (`https://canvas.uw.edu/courses/914697/announcements`): `intro.pdf`, `ugms.pdf` on undirected graphical models, and `tree_inference.pdf` on trees.
- Homework 1 is out, due Tuesday (10/21) at 11:45pm, electronically via our assignment dropbox (`https://canvas.uw.edu/courses/914697/assignments`).

Class Road Map - EE512a

- L1 (9/29): Introduction, Families, Semantics
- L2 (10/1): MRFs, elimination, Inference on Trees
- L3 (10/6): Tree inference, message passing, more general queries, non-tree)
- L4 (10/8): Non-trees, perfect elimination, triangulated graphs
- L5 (10/13): triangulated graphs, k-trees, the triangulation process/heuristics
- L6 (10/15): multiple queries, decomposable models, junction trees
- L7 (10/20): junction trees, intersection graphs, inference on junction trees
- L8 (10/22):
- L9 (10/27):
- L10 (10/29):
- L11 (11/3):
- L12 (11/5):
- L13 (11/10):
- L14 (11/12):
- L15 (11/17):
- L16 (11/19):
- L17 (11/24):
- L18 (11/26):
- L19 (12/1):
- L20 (12/3):
- Final Presentations: (12/10):

Finals Week: Dec 8th-12th, 2014.

Decomposition of G and Decomposable graphs

Repeat of both definitions, but on one page.

Definition 7.2.3 (Decomposition of G)

A *decomposition* of a graph $G = (V, E)$ (if it exists) is a partition (A, B, C) of V such that:

- C separates A from B in G.
- C is a clique.

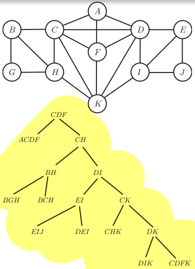
if A and B are both non-empty, then the decomposition is called *proper*.

Definition 7.2.4

A graph $G = (V, E)$ is decomposable if either: 1) G is a clique, or 2) G possesses a **proper** decomposition (A, B, C) s.t. both subgraphs $G[A \cup C]$ and $G[B \cup C]$ are decomposable.

Note part 2. It says *possesses*. Bottom of tree might affect top.

Decomposable & numerator/denominator factorization



- Internal nodes in tree are complete graphs that are also separators.
- Decomposable models factor in a useful way.
- With G decomposable, any $p \in \mathcal{F}(G, \mathcal{M}^{(\mathrm{f})})$ can be written as a numerator/denominator of form:

$$
\begin{aligned}
& p(A,B,C,D,E,F,G,H,I,J,K) \\
&= \frac{p(A,C,D,F)p(B,C,D,E,F,G,H,I,J,K)}{p(C,D,F)} \\
&= \frac{p(A,C,D,F)}{p(C,D,F)} \left(\frac{p(B,C,G,H)p(C,D,E,F,H,I,J,K)}{p(C,H)} \right) \\
&= \dots
\end{aligned}
$$

$$
= \frac{p(A,C,D,F)p(B,G,H)p(C,B,H)p(I,E,J)p(E,I,D)p(C,K,H)p(D,K,I)p(D,K,F,C)}{p(C,D,F)p(C,H)p(B,H)p(D,I)p(E,I)p(C,K)p(D,K)}
$$

Decomposable models

- When $d(S) > 2$, separator marginal use more than once in the denominator
- The general form of the factorization becomes:

$$p(x) = \frac{\prod_{C \in \mathcal{C}(G)} p(x_C)}{\prod_{S \in \mathcal{S}(G)} p(x_S)^{d(S)-1}} \tag{7.2}$$

where $d(S)$ is the shattering coefficient of separator S.

- Any decomposable model can be written this way
- 4-cycle is not decomposable. Two independence properties that can't be used simultaneously.

$$p(x_1, x_2, x_3, x_4) = \frac{p(x_1, x_2, x_4)p(x_1, x_3, x_4)}{p(x_1, x_4)} = \frac{p(x_1, x_2, x_3)p(x_2, x_3, x_4)}{p(x_2, x_3)} \tag{7.3}$$

Decomposable models

Proposition 7.2.3

All of the maxcliques in a graph lie on the leaf nodes of the binary decomposition tree

Proof.

For a decomposable model, the base case (leaf node) is a clique, otherwise it would not be decomposable. If a leaf was not a maxclique (and only a clique), then that means it is contained in a maxclique, and got split by a separator corresponding to that leaf's parent, but this is impossible since a maxcliques have no separator. $\square$

Proposition 7.2.4

The (nec. unique) set of all minimal separators of graph are included in the non-leaf nodes of the binary decomposition tree. $d(S) - 1$ is the number of times the minimal separator S appears as a given non-leaf node.

Triangulated $\equiv$ decomposable

Theorem 7.2.3

A given graph $G = (V, E)$ is triangulated iff it is decomposable.

Proof.

First, recall from Lemma 4.5.6 that a graph is triangulated iff it is decomposable. To prove the current theorem, we will first show (by induction) that decomposability implies that the graph is triangulated). Next, for the converse, we'll show (also by induction on $n = |V|$) that every minimal separator complete in G implies decomposable.

Tree decomposition (definition)

Definition 7.2.3 (tree decomposition)

Given a graph $G = (V, E)$, a tree-decomposition of a graph is a pair $(\{C_i : i \in I\}, T)$ where $T = (I, F)$ is a tree with node index set I, edge set F, and $\{C_i\}_i$ (one for each $i \in I$) is a collection of subsets of $V(G)$ such that:

1. $\cup_{i \in I} C_i = V$
2. for any $(u, v) \in E(G)$, there exists $i \in I$ with $u, v \in C_i$
3. for any $v \in V$, the set $\{i \in I : v \in C_i\}$ forms a connected subtree of T

Cluster graphs

Definition 7.2.4 (Cluster graph)

Consider forming a new graph based on G where the new graph has nodes that correspond to clusters in the original G, and has edges existing between two (cluster) nodes only when the corresponding clusters have a non-zero intersection. That is, let $\mathcal{C}(G) = \{C_1, C_2, \ldots, C_{|I|}\} =$ be a set of $|I|$ clusters of nodes $V(G)$, where $C_i \subseteq V(G), i \in I$. Consider a new graph $\mathcal{J} = (I, \mathcal{E})$ where each node in $\mathcal{J}$ corresponds to a set of nodes in G, and where edge $(i,j) \in \mathcal{E}$ if $C_i \cap C_j \neq \emptyset$. We will also use $S_{ij} = C_i \cap C_j$ as notation.

So two cluster nodes have an edge between them iff there is non-zero intersection between the nodes.

Cluster Trees

If we relax the definition a bit (i.e., drop the requirement for an edge if there exists intersection), and the graph is a tree, then we have what is called a cluster tree.

Definition 7.2.4 (Cluster Tree)

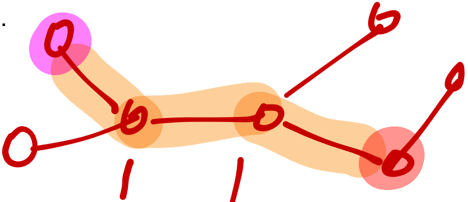
Let $\mathcal{C} = \{C_1, C_2, \ldots, C_{|I|}\}$ be a set of node clusters of graph $G = (V, E)$. A cluster tree is a tree $\mathcal{T} = (I, \mathcal{E}_T)$ with vertices corresponding to clusters in $\mathcal{C}$ and edges corresponding to pairs of clusters $C_1, C_2 \in \mathcal{C}$. We can label each vertex in $i \in I$ by the set of graph nodes in the corresponding cluster in G, and we label each edge $(i, j) \in \mathcal{E}_T$ by the cluster intersection, i.e., $S_{ij} = C_i \cap C_j$.

Cluster Intersection Property (c.i.p.)

Definition 7.2.4 (Cluster Intersection Property)

We are given a cluster tree $\mathcal{T} = (I, \mathcal{E}_T)$, and let C_1, C_2 be any two clusters in the tree. Then the cluster intersection property states that $C_1 \cap C_2 \subseteq C_i$ for all C_i on the (by definition, necessarily) unique path between C_1 and C_2 in the tree $\mathcal{T}$.

- A given cluster tree might or might not have that property.
- Example on the next few slides.



Running Intersection Property (r.i.p.)

Definition 7.2.4 (Running Intersection Property (r.i.p.))

Let $C_1, C_2, \ldots, C_\ell$ be an ordered sequence of subsets of $V(G)$. Then the ordering obeys the running intersection property (r.i.p.) property if for all $i > 1$, there exists $j < i$ such that $C_i \cap (\cup_{k<i} C_k) = C_i \cap C_j$.

- Cluster j acts as a representative for all of i's history.
- r.i.p. is defined in terms of clusters of nodes in a graph.
- r.i.p. holds on an (unordered) set of clusters if such an ordering can be found.

Running Intersection Property (r.i.p.)

Given sequence of clusters $C_1, C_2, \ldots, C_\ell$. Define the history (accumulation) of sequence at position i:

$$H_i = C_1 \cup C_2 \cup \cdots \cup C_i. \tag{7.2}$$

Innovation (residual) or new nodes in C_i not encountered in the previous history, as:

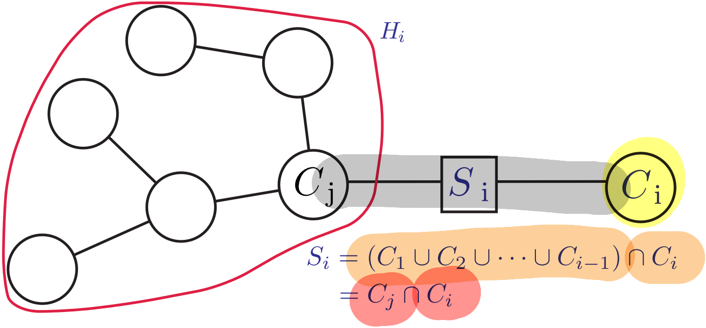
$$R_i = C_i \setminus H_{i-1}. \tag{7.3}$$

Lastly, define the non-innovation, commonality, or separation elements between new and previous history:

$$S_i = C_i \cap H_{i-1} \tag{7.4}$$

Note $C_i = R_i \cup S_i$, i^{th} cluster consists of the innovation R_i and the commonality S_i.

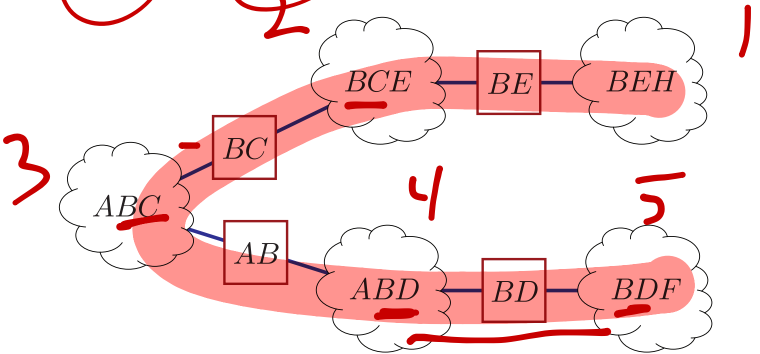
Running Intersection Property (r.i.p.)



Clusters are in r.i.p. order if the commonality S_i between new and history is fully contained in one element of history. I.e., there exists an $j < i$ such that $S_i \subseteq C_j$.

Example: c.i.p. and r.i.p.

2



Example of a set of node clusters (within the cloud-like shapes) arranged in a tree that satisfies the r.i.p. and also the cluster intersection property. The intersections between neighboring node clusters are shown in the figure as square boxes. Consider the path or

$$\{B, E, H\} \cap \{B, D, F\} = \{B\}.$$

First Two Properties: c.i.p. $\equiv$ r.i.p

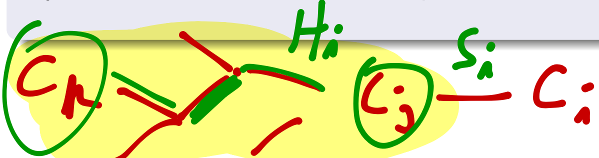
Lemma 7.3.1

The cluster intersection and running intersection properties are identical.

Proof.

Starting with clusters in r.i.p. order, construct cluster tree by connecting each i to its corresponding j node. This is a tree. Also, take any pair C_k, C_i and assume w.l.o.g. that $k < i$ and hence $C_k \subseteq H_{i-1}$. Then $C_i \cap C_k \subseteq C_i \cap H_{i-1} = S_i \subseteq C_j$. Note that C_j is one node closer to C_k on the path. Repeat this process, but with pair C_k, C_j (if $k < j$) or C_j, C_k (if $j < k$) which decreases the path by one edge, until we get adjacent clusters. This shows c.i.p.

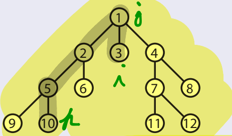
...

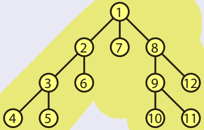


First Two Properties: c.i.p. $\equiv$ r.i.p

. . . proof of Theorem 7.3.1.

Conversely, perform a tree traversal (depth or breadth first search) on cluster tree to produce node ordering.





Then by c.i.p., for any i in that order, and any $k < i$, $C_i \cap C_k \subseteq C_j$ for any j on the unique path between k and i. In particular, $C_i \cap C_k \subseteq C_j$ for $j < i$ being i's neighbor in the tree. Then $\bigcup_{k<i}(C_i \cap C_k) \subseteq C_j$ implying $C_i \cap \bigcup_{k<i} C_k \subseteq C_j$ and so $C_i \cap \bigcup_{k<i} C_k \subseteq C_i \cap C_j$. On the other hand, we always have that $C_i \cap C_j \subseteq C_i \cap \bigcup_{k<i} C_k$, and the two together give us r.i.p.

$j < i$

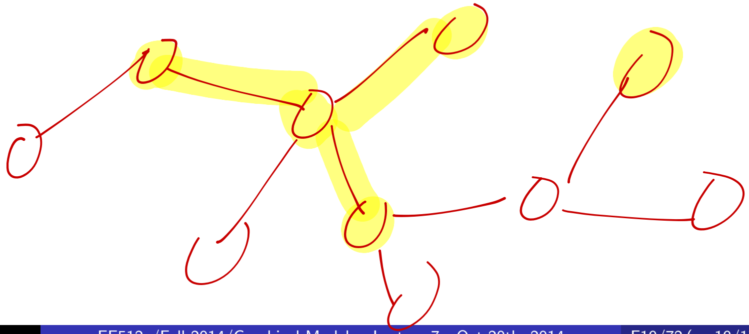
□

Induced sub-tree property (i.s.p.)

Definition 7.3.2 (Induced Sub-tree Property)

Given a cluster tree $\mathcal{T}$ for graph G, the *induced sub-tree property* holds for $\mathcal{T}$ if for all $v \in V$, the set of clusters $C \in \mathcal{C}$ such that $v \in C$ induces a sub-tree $\mathcal{T}(v)$ of $\mathcal{T}$.

Note, by definition the sub-tree is necessarily connected.



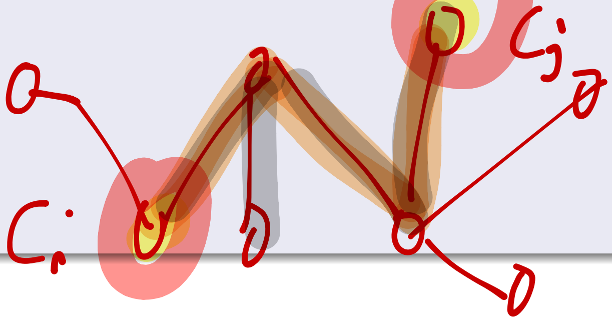
Three properties

Lemma 7.3.3

Induced sub-tree property holds iff cluster intersection property holds

Proof.

Assume induced subtree holds. For any pair C_i, C_j, every $v \in C_i \cap C_j$ induces a sub-tree of $\mathcal{T}$, and all of these sub-trees overlap on the unique path between C_i and C_j in $\mathcal{T}$. □



Three properties

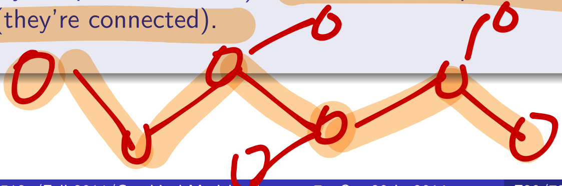
Lemma 7.3.3

Induced sub-tree property holds iff cluster intersection property holds

Proof.

Assume induced subtree holds. For any pair C_i, C_j, every $v \in C_i \cap C_j$ induces a sub-tree of $\mathcal{T}$, and all of these sub-trees overlap on the unique path between C_i and C_j in $\mathcal{T}$.

Conversely, assume c.i.p. holds. For a $v \in V$, consider all clusters that contain v, $\mathcal{C}(v) = \{C \in \mathcal{C} : v \in C\}$. For any pair $C_i, C_j \in \mathcal{C}(v)$, we have that $v \in C_i \cap C_j \subseteq C_k$ for any C_k on the unique path between C_i and C_j. Hence, v always exists on each of these paths. These paths, unioned together, cannot form a cycle (since they are paths on a tree). Moreover, these paths unioned together form a tree (they're connected). □



Three properties

Lemma 7.3.3

Induced sub-tree property holds iff cluster intersection property holds

Proof.

Assume induced subtree holds. For any pair C_i, C_j, every $v \in C_i \cap C_j$ induces a sub-tree of $\mathcal{T}$, and all of these sub-trees overlap on the unique path between C_i and C_j in $\mathcal{T}$.

Conversely, assume c.i.p. holds. For a $v \in V$, consider all clusters that contain v, $\mathcal{C}(v) = \{C \in \mathcal{C} : v \in C\}$. For any pair $C_i, C_j \in \mathcal{C}(v)$, we have that $v \in C_i \cap C_j \subseteq C_k$ for any C_k on the unique path between C_i and C_j. Hence, v always exists on each of these paths. These paths, unioned together, cannot form a cycle (since they are paths on a tree). Moreover, these paths unioned together form a tree (they're connected). □

Thus, 1) c.i.p., 2) r.i.p., and 3) the induced sub-tree property are all identical. We'll henceforth refer them collectively as r.i.p.

Tree decomposition and r.i.p.

Recall the definition of tree decomposition from the previous lecture, repeated again on the next slide.

Tree decomposition (definition)

Definition 7.3.3 (tree decomposition)

Given a graph $G = (V, E)$, a tree-decomposition of a graph is a pair $(\{C_i : i \in I\}, T)$ where $T = (I, F)$ is a tree with node index set I, edge set F, and $\{C_i\}_i$ (one for each $i \in I$) is a collection of subsets of $V(G)$ such that:

1. $\cup_{i \in I} C_i = V$
2. for any $(u, v) \in E(G)$, there exists $i \in I$ with $u, v \in C_i$
3. for any $v \in V$, the set $\{i \in I : v \in C_i\}$ forms a connected subtree of T

Tree decomposition and r.i.p.

Hence, we see that we see that a tree decomposition (when it exists) is just a cluster tree that satisfies (what we now know to be the) induced sub-tree property (e.g., r.i.p. and c.i.p. as well). I.e., property (3) is r.i.p.

Recap

- We want all original graph (o.g.) clique marginals. Why?

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.
- Def: decomposition of a graph, and factorization implication.

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.
- Def: decomposition of a graph, and factorization implication.
- Def: decomposable graph, and decomposition tree

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.
- Def: decomposition of a graph, and factorization implication.
- Def: decomposable graph, and decomposition tree
- Thm: triangulated graph $\equiv$ decomposable graph

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.
- Def: decomposition of a graph, and factorization implication.
- Def: decomposable graph, and decomposition tree
- Thm: triangulated graph $\equiv$ decomposable graph
- Def: tree decomposition (vertex and edge cover, and induced sub-tree).

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.
- Def: decomposition of a graph, and factorization implication.
- Def: decomposable graph, and decomposition tree
- Thm: triangulated graph $\equiv$ decomposable graph
- Def: tree decomposition (vertex and edge cover, and induced sub-tree).
- Def: cluster graph, cluster tree, based only on o.g. nodes, not o.g. edges. Edges in cluster graph cluster tree via cluster intersection.

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.
- Def: decomposition of a graph, and factorization implication.
- Def: decomposable graph, and decomposition tree
- Thm: triangulated graph $\equiv$ decomposable graph
- Def: tree decomposition (vertex and edge cover, and induced sub-tree).
- Def: cluster graph, cluster tree, based only on o.g. nodes, not o.g. edges. Edges in cluster graph cluster tree via cluster intersection.
- Def: cluster intersection property, running intersection property, induced sub-tree property, r.i.p.

Recap

- We want all original graph (o.g.) clique marginals. Why?
- Finding optimal elimination order is optimal for all o.g. clique marginals.
- Def: decomposition of a graph, and factorization implication.
- Def: decomposable graph, and decomposition tree
- Thm: triangulated graph $\equiv$ decomposable graph
- Def: tree decomposition (vertex and edge cover, and induced sub-tree).
- Def: cluster graph, cluster tree, based only on o.g. nodes, not o.g. edges. Edges in cluster graph cluster tree via cluster intersection.
- Def: cluster intersection property, running intersection property, induced sub-tree property, r.i.p.
- Next def: Junction tree, cluster tree with r.i.p. and edge cover.

Junction Tree

Definition 7.3.4

Given a graph $G = (V, E)$, a junction tree corresponding to G (if it exists) is a cluster tree $\mathcal{T} = (\mathcal{C}, E_T)$ having the r.i.p. over the clusters, and where any nodes u, v adjacent via edge $(u, v) \in E(G)$ are together in at least one cluster.

Junction Tree

Definition 7.3.4

Given a graph $G = (V, E)$, a junction tree corresponding to G (if it exists) is a cluster tree $\mathcal{T} = (\mathcal{C}, E_T)$ having the r.i.p. over the clusters, and where any nodes u, v adjacent via edge $(u, v) \in E(G)$ are together in at least one cluster.

- So, junction tree (JT), for a given graph G, is a cluster tree that: 1) satisfies r.i.p. over the clusters, and 2) includes all edges (edge cover). Not all r.i.p.-satisfying cluster trees need be an edge cover.

Junction Tree

Definition 7.3.4

Given a graph $G = (V, E)$, a junction tree corresponding to G (if it exists) is a cluster tree $\mathcal{T} = (\mathcal{C}, E_T)$ having the r.i.p. over the clusters, and where any nodes u, v adjacent via edge $(u, v) \in E(G)$ are together in at least one cluster.

- So, junction tree (JT), for a given graph G, is a cluster tree that: 1) satisfies r.i.p. over the clusters, and 2) includes all edges (edge cover). Not all r.i.p.-satisfying cluster trees need be an edge cover.
- Clusters in JT need not be original graph cliques!!

Junction Tree

Definition 7.3.4

Given a graph $G = (V, E)$, a junction tree corresponding to G (if it exists) is a cluster tree $\mathcal{T} = (\mathcal{C}, E_T)$ having the r.i.p. over the clusters, and where any nodes u, v adjacent via edge $(u, v) \in E(G)$ are together in at least one cluster.

- So, junction tree (JT), for a given graph G, is a cluster tree that: 1) satisfies r.i.p. over the clusters, and 2) includes all edges (edge cover). Not all r.i.p.-satisfying cluster trees need be an edge cover.
- Clusters in JT need not be original graph cliques!!
- JT could have clusters corresponding to cliques, maxcliques, or neither of the above.

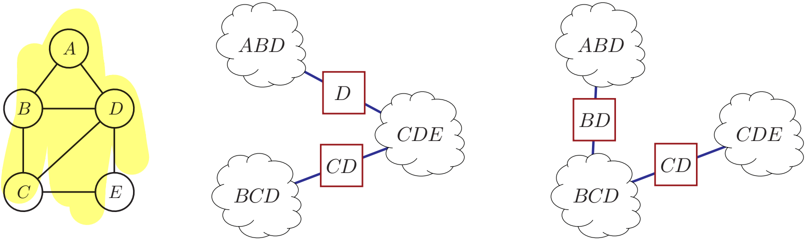
Junction Tree

Definition 7.3.4

Given a graph $G = (V, E)$, a junction tree corresponding to G (if it exists) is a cluster tree $\mathcal{T} = (\mathcal{C}, E_T)$ having the r.i.p. over the clusters, and where any nodes u, v adjacent via edge $(u, v) \in E(G)$ are together in at least one cluster.

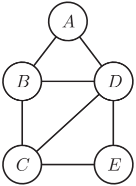
- So, junction tree (JT), for a given graph G, is a cluster tree that: 1) satisfies r.i.p. over the clusters, and 2) includes all edges (edge cover). Not all r.i.p.-satisfying cluster trees need be an edge cover.
- Clusters in JT need not be original graph cliques!!
- JT could have clusters corresponding to cliques, maxcliques, or neither of the above.
- If clusters correspond to the original graph cliques (resp. maxcliques) in G, it called a junction tree of cliques (resp. maxcliques).

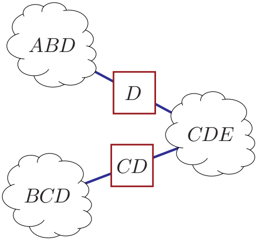
Examples junction trees and not

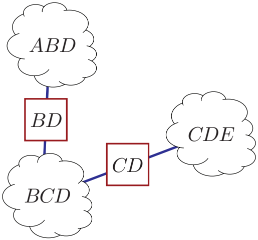


Questions to answer:

Examples junction trees and not







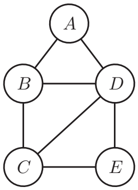
no

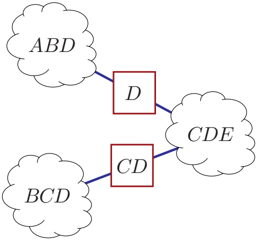
no

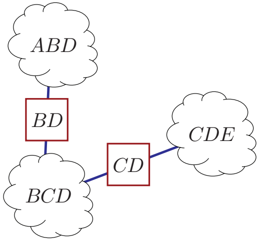
Questions to answer:

- cluster graph?

Examples junction trees and not







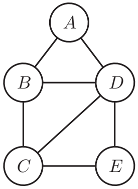
Questions to answer:

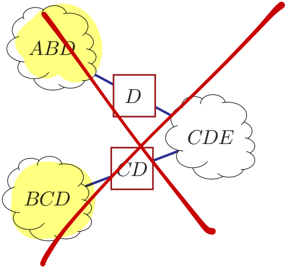
- cluster graph?
- cluster tree?

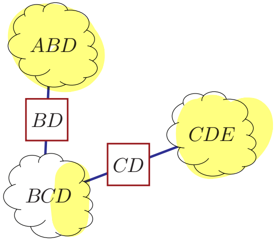
yes

yes

Examples junction trees and not







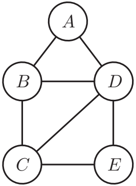
Questions to answer:

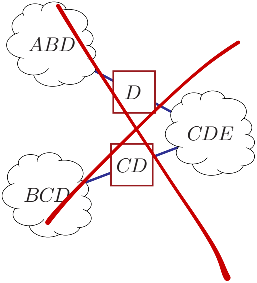
- cluster graph?
- cluster tree?
- Junction tree?

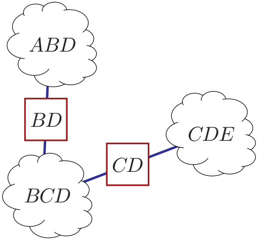
no

yes

Examples junction trees and not







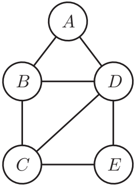
Questions to answer:

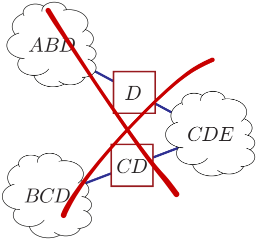
- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques?

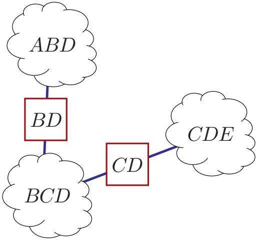
no

yes

Examples junction trees and not







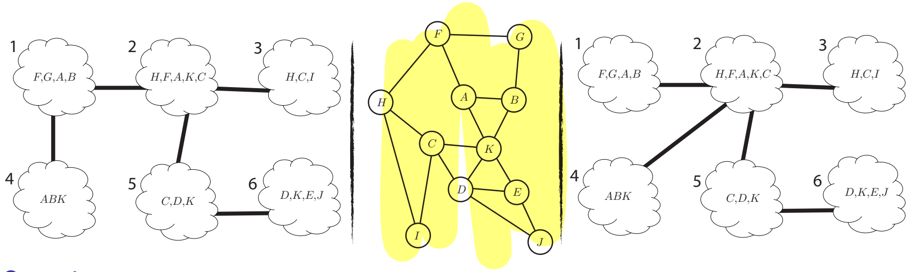
Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques?
- Junction tree of maxcliques?

no

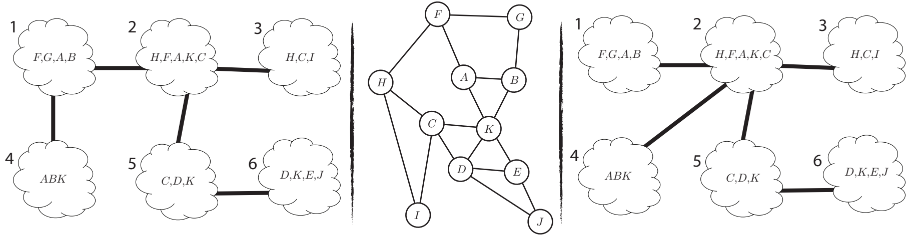
yes.

Examples junction trees and not



Questions to answer:

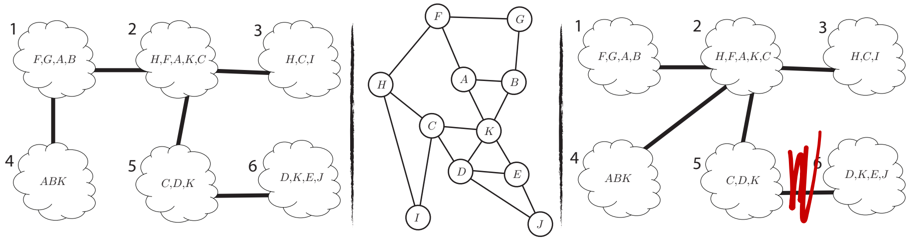
Examples junction trees and not



Questions to answer:

- cluster graph? no

Examples junction trees and not

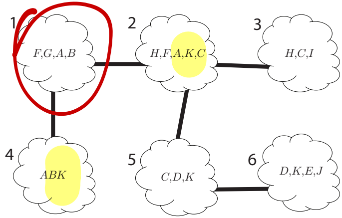


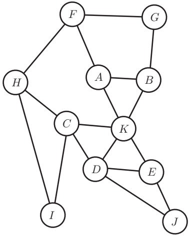
Questions to answer:

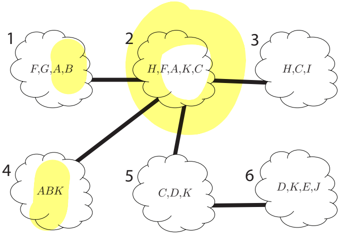
- cluster graph?
- cluster tree? Yes

Yes

Examples junction trees and not





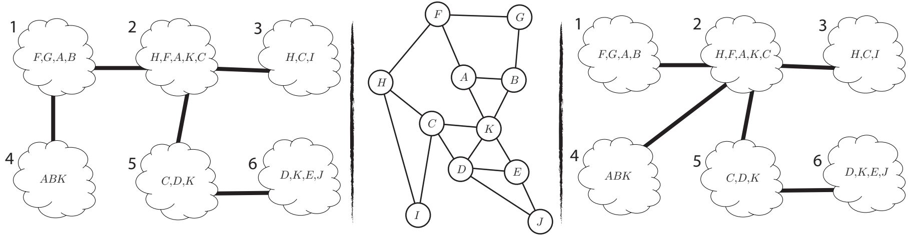


Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree? no

no

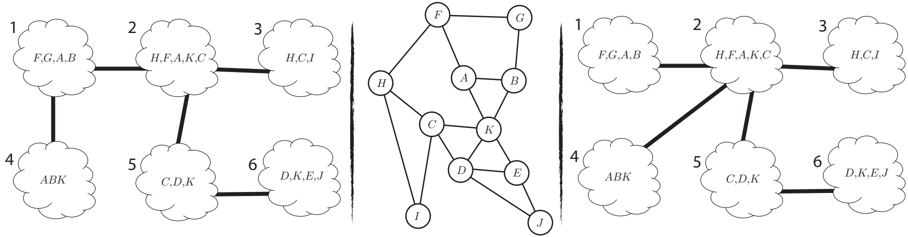
Examples junction trees and not



Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques?

Examples junction trees and not

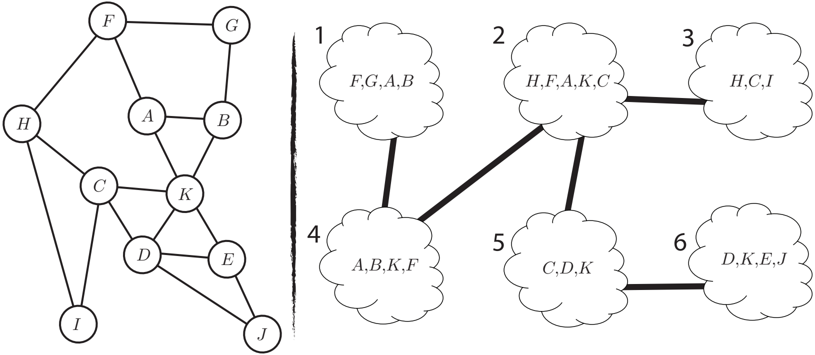


Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques?
- Junction tree of maxcliques?

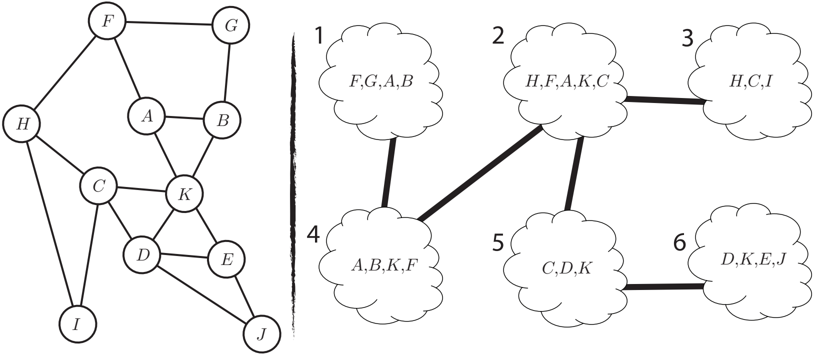
no

Examples junction trees and not



Questions to answer:

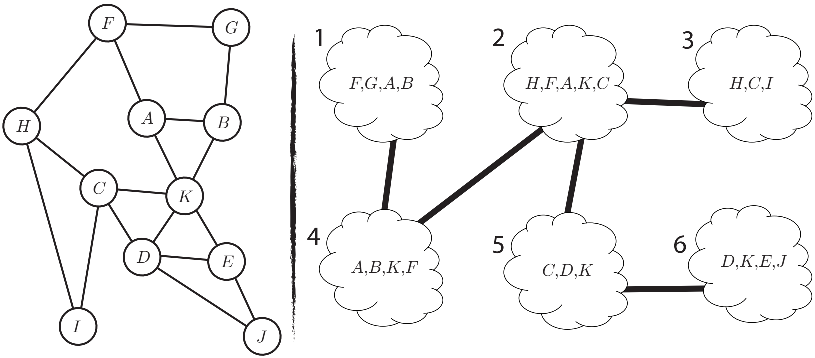
Examples junction trees and not



Questions to answer:

- cluster graph? no

Examples junction trees and not

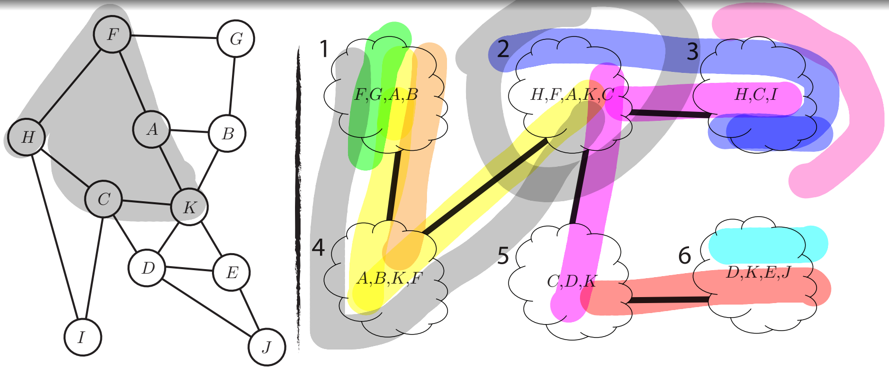


Questions to answer:

- cluster graph?
- cluster tree?

Yes

Examples junction trees and not

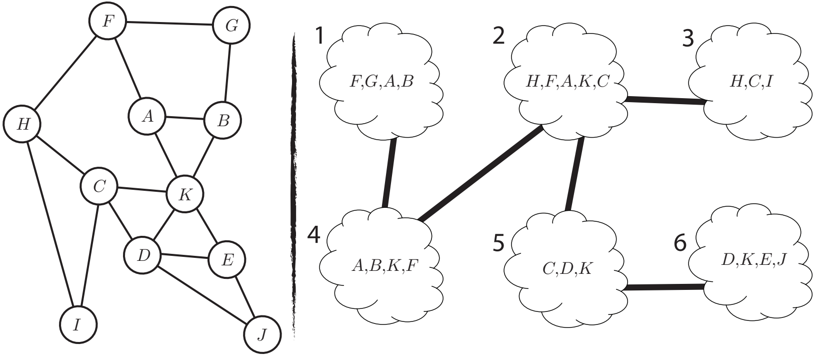


Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree?

yes

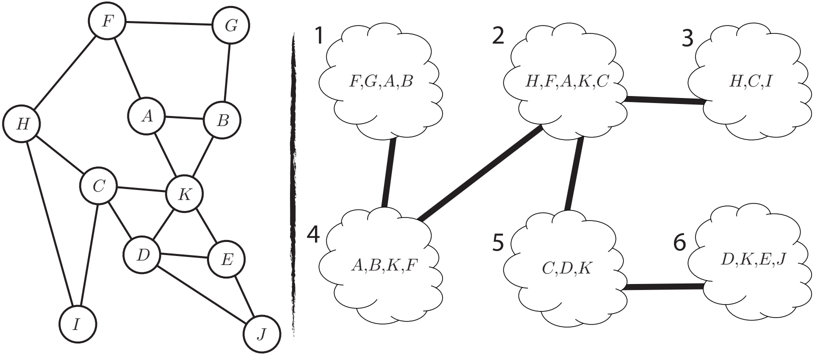
Examples junction trees and not



Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques?

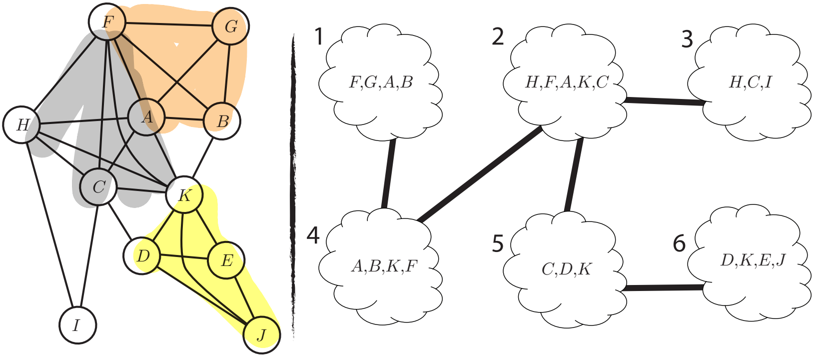
Examples junction trees and not



Questions to answer:

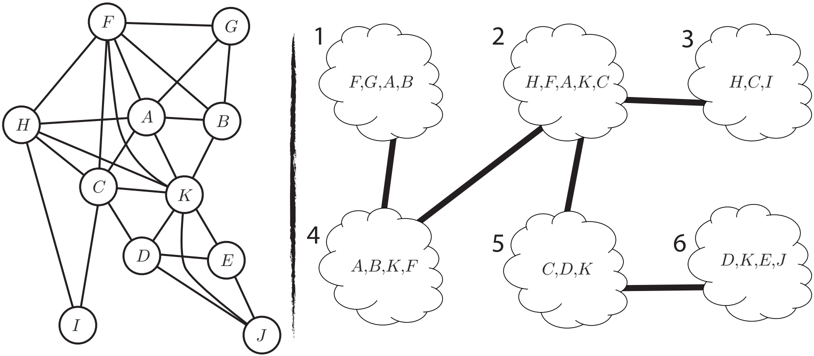
- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques?
- Junction tree of maxcliques?

Examples junction trees and not



Questions to answer:

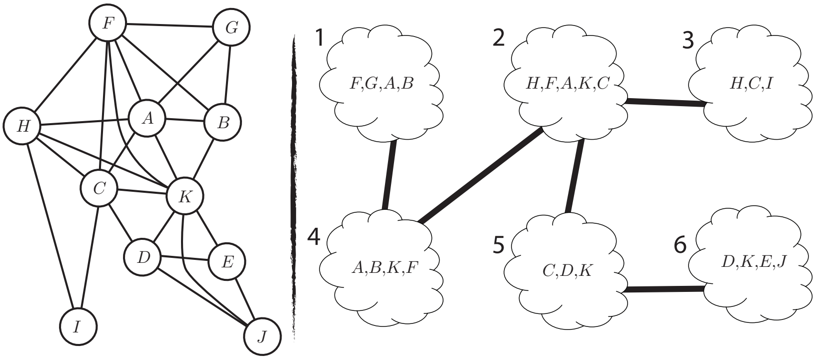
Examples junction trees and not



Questions to answer:

- cluster graph? no

Examples junction trees and not

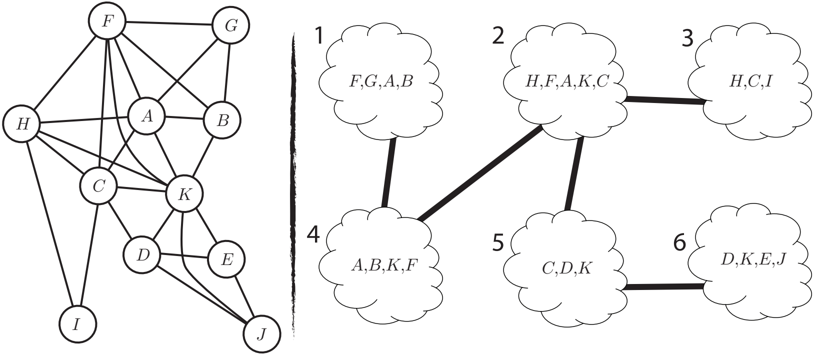


Questions to answer:

- cluster graph?
- cluster tree?

yes

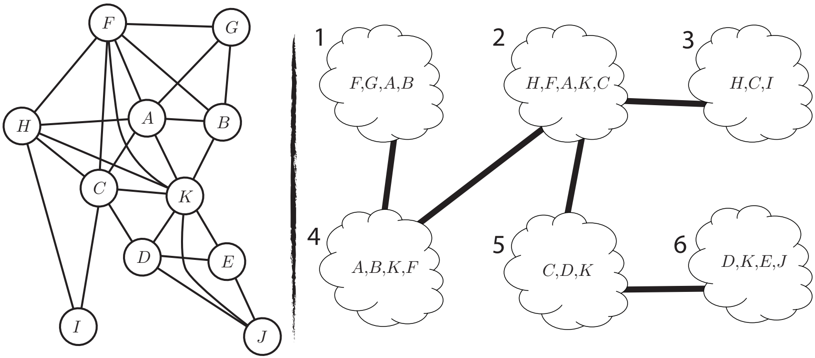
Examples junction trees and not



Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree?

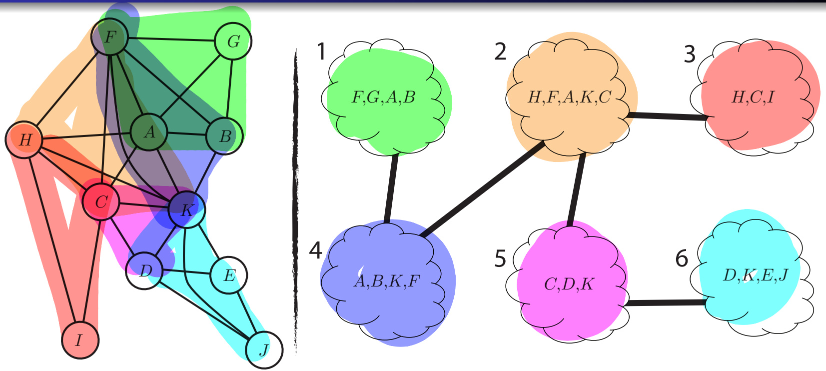
Examples junction trees and not



Questions to answer:

- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques? Yes

Examples junction trees and not

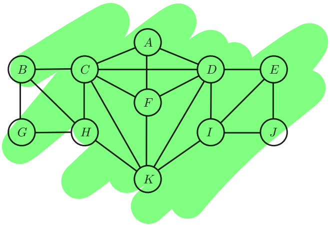


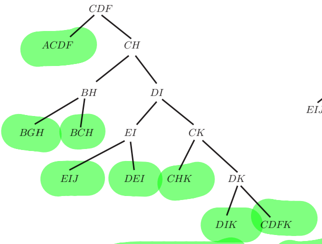
Questions to answer:

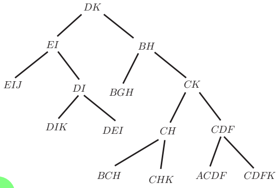
- cluster graph?
- cluster tree?
- Junction tree?
- Junction tree of cliques?
- Junction tree of maxcliques?

Yes

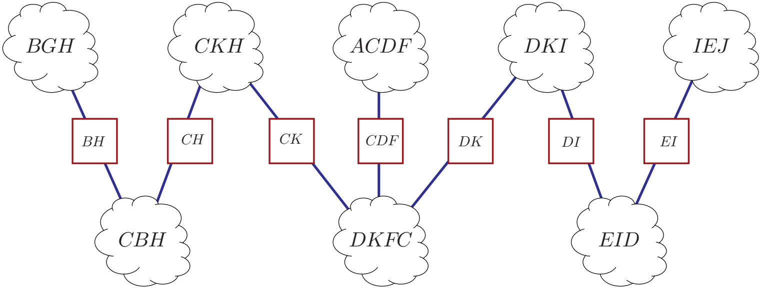
Examples junction trees and not







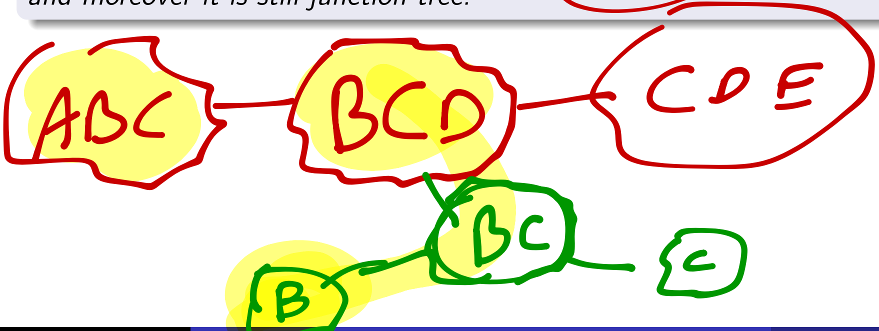
Tree of cliques for above graph. Does r.i.p. hold? JT? JT of cliques? JT of maxcliques?



Junction Tree Preserving Operations

Lemma 7.3.5

Given a junction tree, form a new cluster tree as follows. For each cluster C in the JT, choose an order of nodes within C, say $c_1, c_2, \ldots, c_k$, and hang a chain of clusters off of C consisting of $C \setminus \{c_1\}$ hanging from C, $C \setminus \{c_1, c_2\}$ hanging from $C \setminus \{c_1\}$, $C \setminus \{c_1, c_2, c_3\}$ hanging from $C \setminus \{c_1, c_2\}$, and so on. Then the resulting cluster graph is a cluster tree, and moreover it is still junction tree.



Junction Tree Preserving Operations

Lemma 7.3.5

Given a junction tree, form a new cluster tree as follows. For each cluster C in the JT, choose an order of nodes within C, say $c_1, c_2, \ldots, c_k$, and hang a chain of clusters off of C consisting of $C \setminus \{c_1\}$ hanging from C, $C \setminus \{c_1, c_2\}$ hanging from $C \setminus \{c_1\}$, $C \setminus \{c_1, c_2, c_3\}$ hanging from $C \setminus \{c_1, c_2\}$, and so on. Then the resulting cluster graph is a cluster tree, and moreover it is still junction tree.

Lemma 7.3.6

Given a junction tree, where (C_i, C_j) are neighboring clusters in the tree, we can merge these two clusters forming a new cluster $C_{ij} = C_i \cup C_j$, and where the neighbors of C_{ij} are the set of neighbors of either C_i ~~and~~ C_j. Then the resulting structure is still junction tree.

or

Junction Tree Preserving Operations

Lemma 7.3.5

Given a junction tree, form a new cluster tree as follows. For each cluster C in the JT, choose an order of nodes within C, say $c_1, c_2, \ldots, c_k$, and hang a chain of clusters off of C consisting of $C \setminus \{c_1\}$ hanging from C, $C \setminus \{c_1, c_2\}$ hanging from $C \setminus \{c_1\}$, $C \setminus \{c_1, c_2, c_3\}$ hanging from $C \setminus \{c_1, c_2\}$, and so on. Then the resulting cluster graph is a cluster tree, and moreover it is still junction tree.

Lemma 7.3.6

Given a junction tree, where (C_i, C_j) are neighboring clusters in the tree, we can merge these two clusters forming a new cluster $C_{ij} = C_i \cup C_j$, and where the neighbors of C_{ij} are the set of neighbors of either C_i and C_j. Then the resulting structure is still junction tree.

If we keep doing the latter, we'll end up with one complete graph.

Key theorem: JT of maxcliques $\equiv$ triangulated graphs

Theorem 7.3.7

A graph $G = (V, E)$ is decomposable iff a junction tree of maxcliques for G exists.

Proof.

a junction tree exists $\Leftrightarrow$ decomposable: Induction on the number of maxcliques. If G has one maxclique, it is both a junction tree and decomposable. Assume true for $\leq k$ maxcliques and show it for $k + 1$.

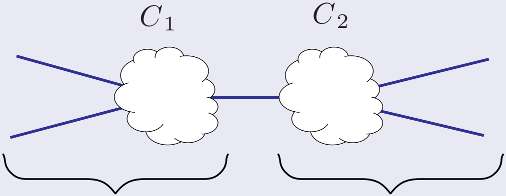
. . .

Junction tree of maxcliques ≡ triangulated graphs

JT of maxcliques implies Decomposable

... proof continued.

a junction tree exists ⇒ decomposable: Let $\mathcal{T}$ be a junction tree of maxcliques $\mathcal{C}$, and let C_1, C_2 be adjacent in $\mathcal{T}$. The edge C_1, C_2 in the tree separates $\mathcal{T}$ into two sub-trees $\mathcal{T}_1$ and $\mathcal{T}_2$, with V_i being the nodes in $\mathcal{T}_i$, $G_i = G[V_i]$ being the subgraph of G corresponding to $\mathcal{T}_i$, and $\mathcal{C}_i$ being the set of maxcliques in $\mathcal{T}_i$, for $i = 1, 2$. Thus $V(G) = V_1 \cup V_2$, and $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$. Note that $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$. We also let $S = V_1 \cap V_2$ which is the intersection of all the nodes in each of the two trees.



Tree $\mathcal{T}_1$ with nodes V_1 forming graph $G_1 = G[V_1]$ and max-cliques $\mathcal{C}_1$.

Tree $\mathcal{T}_2$ with nodes V_2 forming graph $G_2 = G[V_2]$ and max-cliques $\mathcal{C}_2$.

Junction tree of maxcliques ≡ triangulated graphs

JT of maxcliques implies Decomposable

... proof continued.

Also, the nodes in $\mathcal{T}_i$ are maxcliques in G_i and $\mathcal{T}_i$ is a junction tree for G_i since r.i.p. still holds in the subtrees of a junction tree. Therefore, by induction, G_i is decomposable. To show that G is decomposable, we need to show that: 1) $S = V_1 \cap V_2$ is complete, and 2) that S separates $G[V_1 \setminus S]$ from $G[V_2 \setminus S]$.

If $v \in S$, then for each G_i $(i = 1, 2)$, there exists a clique C'_i with $v \in C'_i$, and the path in $\mathcal{T}$ joining C'_1 and C'_2 passes through both C_1 and C_2. Because of the r.i.p., we thus have that $v \in C_1$ and $v \in C_2$ and so $v \in C_1 \cap C_2$. This means that $V_1 \cap V_2 \subseteq C_1 \cap C_2$. But $C_i \subseteq V_i$ since C_i is a clique in the corresponding tree $\mathcal{T}_i$. Therefore $C_1 \cap C_2 \subseteq V_1 \cap V_2 = S$, so that $S = C_1 \cap C_2$. This means that S contains all nodes that are common among the two subgraphs and moreover that S is complete as desired.

...

Junction tree of maxcliques ≡ triangulated graphs

JT of maxcliques implies Decomposable

… proof continued.

Next, to show that S is a separator, we take $u \in V_1 \setminus S$ and $v \in V_2 \setminus S$ (note that such choices mean $u \notin V_2$ and $v \notin V_1$ due to the commonality property of S). Suppose the contrary that S does not separate V_1 from V_2, which means there exists a path $u, w_1, w_2, \ldots, w_k, v$ for the given u, v with $w_i \notin S$ for all i. Therefore, there is a clique $C \in \mathcal{C}$ containing the set $\{u, w_1\}$. We must have $C \notin \mathcal{C}_2$ since $u \notin V_2$, which means $C \in \mathcal{C}_1$ or $C \subseteq V_1$ implying that $w_1 \in V_1$ and moreover that $w_1 \in V_1 \setminus S$. We repeat this argument with w_1 taking the place of u and w_2 taking the place of w_1 in the path, and so on until we end up with $v \in V_1 \setminus S$ which is a contradiction. Therefore, S must separate V_1 from V_2. We have thus formed a decomposition of G as $(V_1 \setminus S, V_2 \setminus S, S)$ and since G_i is decomposable (by induction), we have that G is decomposable.

…

Junction tree of maxcliques $\equiv$ triangulated graphs

Decomposable implies JT of maxcliques

... proof continued.

decomposable $\Rightarrow$ *a junction tree exists:* Since G is decomposable, let (W_1, W_2, S) be a proper decomposition of G into decomposable subsets $G_1 = G[V_1]$ and $G_2 = G[V_2]$ with $V_i = W_i \cup S$. By induction, since G_1 and G_2 are decomposable, there exits a junction tree $\mathcal{T}_1$ and $\mathcal{T}_2$ corresponding to maxcliques in G_1 and G_2. Since this is a decomposition, with separator S, we can form all maxcliques $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ with $\mathcal{C}_i$ maxcliques of V_i for tree $\mathcal{T}_i$. Choose $C_1 \in \mathcal{C}_1$ and $C_2 \in \mathcal{C}_2$ such that $S \subseteq C_1$ and $S \subseteq C_2$ which is possible since S is complete, and must be contained in some maxclique in both $\mathcal{T}_1$ and $\mathcal{T}_2$. We form a new tree $\mathcal{T}$ by linking $C_1 \in \mathcal{T}_1$ with $C_2 \in \mathcal{T}_2$. We need next to ensure that this new junction tree satisfies r.i.p.

...

Junction tree of maxcliques ≡ triangulated graphs

Decomposable implies JT of maxcliques

… proof continued.

Let $v \in V$. If $v \notin V_2$, then all cliques containing v are in $\mathcal{C}_1$ and those cliques form a connected tree by the junction tree property since $\mathcal{T}_1$ is a junction tree. The same is true if $v \notin V_1$. Otherwise, if $v \in S$ (meaning that $v \in V_1 \cap V_2$), then the cliques in $\mathcal{C}_i$ containing v are connected in $\mathcal{T}_i$ including C_i for $i = 1, 2$. But by forming $\mathcal{T}$ by connecting C_1 andC_2, and since v is arbitrary, we have retained the junction tree property. Thus, $\mathcal{T}$ is a junction tree.

□

Cliques or Maxcliques

Lemma 7.3.8

A junction tree of maxcliques for graph $G = (V, E)$ *exists iff a junction tree of cliques for graph* $G = (V, E)$ *exists.*

Cliques or Maxcliques

Lemma 7.3.8

A junction tree of maxcliques for graph $G = (V, E)$ exists iff a junction tree of cliques for graph $G = (V, E)$ exists.

- How can we get from one to the other?

Exercise

- Q: if you have a JT of cliques, can there be a missing maxclique?

Cliques or Maxcliques

Lemma 7.3.8

A junction tree of maxcliques for graph $G = (V, E)$ exists iff a junction tree of cliques for graph $G = (V, E)$ exists.

- How can we get from one to the other?

Since decomposable is same as triangulated:

Corollary 7.3.9

A graph G is triangulated iff a junction tree of cliques for G exists.

How to build a junction tree

- Maximum cardinality search algorithm can do this. If graph is triangulated, it produces a list of cliques in r.i.p. order.

Maximum Cardinality Search with maxclique order

Algorithm 1: Maximum Cardinality Search: Determines if a graph G is triangulated.

Input: An undirected graph $G = (V, E)$ with $n = |V|$.
Result: is triangulated?, if so MCS ordering $\sigma = (v_1, \ldots, v_n)$, and maxcliques in r.i.p. order.

$L \leftarrow \emptyset$; $i \leftarrow 1$; $\mathcal{C} \leftarrow \emptyset$;
while $|V \setminus L| > 0$ **do**
  Choose $v_i \in \text{argmax}_{u \in V \setminus L} |\delta(u) \cap L|$; /* v_i's previously labeled neighbors has max cardinality. */
  $c_i \leftarrow \delta(v_i) \cap L$; /* c_i is v_i's neighbors in the reverse elimination order. */
  if $\{v_i\} \cup c_i$ *is not complete in* G **then**
    return *"not triangulated"* ;
  if $|c_i| \leq |c_{i-1}|$ **then**
    $\mathcal{C} \leftarrow (\mathcal{C}, \{c_{i-1} \cup \{v_{i-1}\}\})$; /* Append the next maxclique to list $\mathcal{C}$. */
  if $i = n$ **then**
    $\mathcal{C} \leftarrow (\mathcal{C}, \{c_i \cup \{v_i\}\})$; ; /* Append the last maxclique to list $\mathcal{C}$. */
  $L \leftarrow L \cup \{v_i\}$ $i \leftarrow i + 1$;
return *"triangulated", the ordering* σ*, and the set of maxcliques* $\mathcal{C}$ *which are in r.i.p. order.*

Maximum Cardinality Search with maxclique order

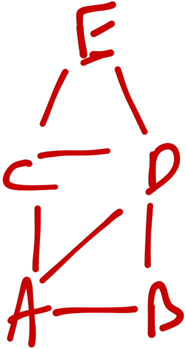
Algorithm 2: Maximum Cardinality Search: Determines if a graph G is triangulated.

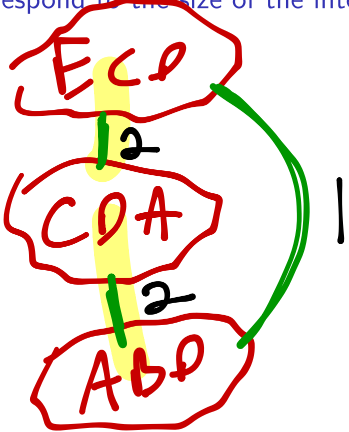
Input: An undirected graph $G = (V, E)$ with $n = |V|$.
Result: is triangulated?, if so MCS ordering $\sigma = (v_1, \ldots, v_n)$, and maxcliques in r.i.p. order.

```
L ← ∅ ; i ← 1 ; C ← ∅ ;
while |V \ L| > 0 do
    Choose v_i ∈ argmax_{u∈V\L} |δ(u) ∩ L| ; /* v_i's previously labeled neighbors has max cardinality. */
    c_i ← δ(v_i) ∩ L ;          /* c_i is v_i's neighbors in the reverse elimination order. */
    if {v_i} ∪ c_i is not complete in G then
        return "not triangulated" ;
    if |c_i| ≤ |c_{i-1}| then
        C ← (C, {c_{i-1} ∪ {v_{i-1}}}) ;          /* Append the next maxclique to list C. */
    if i = n then
        C ← (C, {c_i ∪ {v_i}}) ; ;          /* Append the last maxclique to list C. */
    L ← L ∪ {v_i} i ← i + 1 ;
return "triangulated", the ordering σ, and the set of maxcliques C which are in r.i.p. order.
```

How to build a junction tree

- Alternatively, we can construct the maxcliques in any form (say by running elimination) and find a maximal spanning tree over the edge-weighted cluster graph, where clusters correspond to maxcliques, and edge weights correspond to the size of the intersection of the two adjacent maxcliques.





How to build a junction tree

- Alternatively, we can construct the maxcliques in any form (say by running elimination) and find a maximal spanning tree over the edge-weighted cluster graph, where clusters correspond to maxcliques, and edge weights correspond to the size of the intersection of the two adjacent maxcliques.
- Prim's algorithm can run in $O(|E| + |V| \log |V|)$, much better than $|V|^2$ for sparse graphs.

How to build a junction tree

- Alternatively, we can construct the maxcliques in any form (say by running elimination) and find a maximal spanning tree over the edge-weighted cluster graph, where clusters correspond to maxcliques, and edge weights correspond to the size of the intersection of the two adjacent maxcliques.
- Prim's algorithm can run in $O(|E| + |V| \log |V|)$, much better than $|V|^2$ for sparse graphs.

Theorem 7.3.10

A tree of maxcliques $\mathcal{T}$ is a junction tree iff it is a maximum spanning tree on the maxclique graph, with edge weights set according to the cardinality of the separator between the two maxcliques.

How to build a junction tree

- Alternatively, we can construct the maxcliques in any form (say by running elimination) and find a maximal spanning tree over the edge-weighted cluster graph, where clusters correspond to maxcliques, and edge weights correspond to the size of the intersection of the two adjacent maxcliques.
- Prim's algorithm can run in $O(|E| + |V| \log |V|)$, much better than $|V|^2$ for sparse graphs.

Theorem 7.3.10

A tree of maxcliques $\mathcal{T}$ is a junction tree iff it is a maximum spanning tree on the maxclique graph, with edge weights set according to the cardinality of the separator between the two maxcliques.

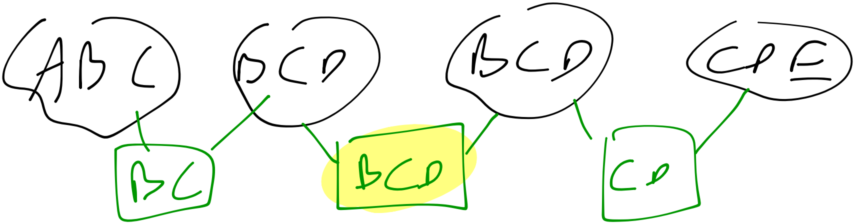
- Note: graph must be triangulated. I.e., maximum spanning tree of a cluster graph where the clusters are maxcliques but the graph is not triangulated will clearly not produce a junction tree.

Other aspects of JTs

- There can be multiple JTs for a given triangulated graph (e.g., consider any graph where $d(S) \geq 3$ for some separator S).

Other aspects of JTs

- There can be multiple JTs for a given triangulated graph (e.g., consider any graph where $d(S) \geq 3$ for some separator S).
- JTs are not binary decomposition trees (BDTs), but they are related. Leaf nodes of BDTs correspond to nodes in a JT of maxcliques. Non-leaf nodes in a BDTs may correspond to edges in a JT. Therefore, edges in a JT may correspond to all minimal separators in triangulated graph G' but also might not (e.g., $\{ABC\} - \{BCD\} - \{CDE\}$ with $\{BCD\}$ repeated).



Other aspects of JTs

- There can be multiple JTs for a given triangulated graph (e.g., consider any graph where $d(S) \geq 3$ for some separator S).
- JTs are not binary decomposition trees (BDTs), but they are related. Leaf nodes of BDTs correspond to nodes in a JT of maxcliques. Non-leaf nodes in a BDTs may correspond to edges in a JT. Therefore, edges in a JT may correspond to all minimal separators in triangulated graph G' but also might not (e.g., $\{ABC\} - \{BCD\} - \{CDE\}$ with $\{BCD\}$ repeated).
- Set of maxcliques is unique in a triangulated graph. Set of minimal separators is unique in a triangulated graph.

Other aspects of JTs

- There can be multiple JTs for a given triangulated graph (e.g., consider any graph where $d(S) \geq 3$ for some separator S).
- JTs are not binary decomposition trees (BDTs), but they are related. Leaf nodes of BDTs correspond to nodes in a JT of maxcliques. Non-leaf nodes in a BDTs may correspond to edges in a JT. Therefore, edges in a JT may correspond to all minimal separators in triangulated graph G' but also might not (e.g., $\{ABC\} - \{BCD\} - \{CDE\}$ with $\{BCD\}$ repeated).
- Set of maxcliques is unique in a triangulated graph. Set of minimal separators is unique in a triangulated graph.
- Again, JT can be over not just maxcliques. JT can exist over all cliques, or over some cliques (if they contain all maxcliques)

Other aspects of JTs

- There can be multiple JTs for a given triangulated graph (e.g., consider any graph where $d(S) \geq 3$ for some separator S).
- JTs are not binary decomposition trees (BDTs), but they are related. Leaf nodes of BDTs correspond to nodes in a JT of maxcliques. Non-leaf nodes in a BDTs may correspond to edges in a JT. Therefore, edges in a JT may correspond to all minimal separators in triangulated graph G' but also might not (e.g., $\{ABC\} - \{BCD\} - \{CDE\}$ with $\{BCD\}$ repeated).
- Set of maxcliques is unique in a triangulated graph. Set of minimal separators is unique in a triangulated graph.
- Again, JT can be over not just maxcliques. JT can exist over all cliques, or over some cliques (if they contain all maxcliques)
- Different JTs of maxcliques always has same set of nodes and separators, just different configurations.

Intersection Graphs

- We're next going to look at seemingly very different way to view triangulated graphs and junction trees, based on intersection graph theory.

Intersection Graphs

- We're next going to look at seemingly very different way to view triangulated graphs and junction trees, based on intersection graph theory.
- We'll see that triangulated graphs are identical to a type of intersection graph, where the underlying object is a tree (furthering our connection to trees).

Intersection Graphs

- We're next going to look at seemingly very different way to view triangulated graphs and junction trees, based on intersection graph theory.
- We'll see that triangulated graphs are identical to a type of intersection graph, where the underlying object is a tree (furthering our connection to trees).
- first, lets talk a bit about terminology.

Covers (in general) and Edge Clique Covers

- Set cover - sets must cover the ground/universal set (ground set cover)

Covers (in general) and Edge Clique Covers

- Set cover - sets must cover the ground/universal set (ground set cover)
- Vertex cover - vertices must cover the edges (edge vertex cover)

Covers (in general) and Edge Clique Covers

- Set cover - sets must cover the ground/universal set (ground set cover)
- Vertex cover - vertices must cover the edges (edge vertex cover)
- Edge cover - edges must cover the vertices (vertex edge cover)

Covers (in general) and Edge Clique Covers

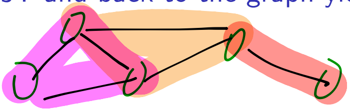
- Set cover - sets must cover the ground/universal set (ground set cover)
- Vertex cover - vertices must cover the edges (edge vertex cover)
- Edge cover - edges must cover the vertices (vertex edge cover)
- clique cover - cliques cover the edges (edge clique cover)

Covers (in general) and Edge Clique Covers

- Set cover - sets must cover the ground/universal set (ground set cover)
- Vertex cover - vertices must cover the edges (edge vertex cover)
- Edge cover - edges must cover the vertices (vertex edge cover)
- clique cover - cliques cover the edges (edge clique cover)
- The nodes of a junction tree of cliques (or maxcliques) constitute an edge clique cover for triangulated graph G' — start with set of nodes $V = \cup_{C \in \mathcal{C}} C$. Add edge between $u, v \in V$ if exists a $C \in \mathcal{C}$ such that $u, v \in C$.

Covers (in general) and Edge Clique Covers

- Set cover - sets must cover the ground/universal set (ground set cover)
- Vertex cover - vertices must cover the edges (edge vertex cover)
- Edge cover - edges must cover the vertices (vertex edge cover)
- clique cover - cliques cover the edges (edge clique cover)
- The nodes of a junction tree of cliques (or maxcliques) constitute an edge clique cover for triangulated graph G' — start with set of nodes $V = \cup_{C \in \mathcal{C}} C$. Add edge between $u, v \in V$ if exists a $C \in \mathcal{C}$ such that $u, v \in C$.
- Going from G' to JT and back to the graph yields the same graph.



Intersection Graphs

Definition 7.4.1 (Intersection Graph)

An intersection graph is a graph $G = (V, E)$ where each vertex $v \in V(G)$ corresponds to a set U_v and each edge $(u, v) \in E(G)$ exists only if $U_u \cap U_v \neq \emptyset$.

Intersection Graphs

Definition 7.4.1 (Intersection Graph)

An intersection graph is a graph $G = (V, E)$ where each vertex $v \in V(G)$ corresponds to a set U_v and each edge $(u, v) \in E(G)$ exists only if $U_u \cap U_v \neq \emptyset$.

- some underlying set of objects U and a **multiset** of subsets of U of the form $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ with $U_i \subseteq U$ — might have some i, j where $U_i = U_j$.

Intersection Graphs

Definition 7.4.1 (Intersection Graph)

An intersection graph is a graph $G = (V, E)$ where each vertex $v \in V(G)$ corresponds to a set U_v and each edge $(u, v) \in E(G)$ exists only if $U_u \cap U_v \neq \emptyset$.

- some underlying set of objects U and a **multiset** of subsets of U of the form $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ with $U_i \subseteq U$ — might have some i, j where $U_i = U_j$.

Theorem 7.4.2

Every graph is an intersection graph.

Intersection Graphs

Definition 7.4.1 (Intersection Graph)

An intersection graph is a graph $G = (V, E)$ where each vertex $v \in V(G)$ corresponds to a set U_v and each edge $(u, v) \in E(G)$ exists only if $U_u \cap U_v \neq \emptyset$.

- some underlying set of objects U and a **multiset** of subsets of U of the form $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ with $U_i \subseteq U$ — might have some i, j where $U_i = U_j$.

Theorem 7.4.2

Every graph is an intersection graph.

This can be seen informally by consider an arbitrary graph, create a U_i for every node, and construct the subsets so that the edges will exist when taking intersection.

Interval Graphs (a type of intersection graph)

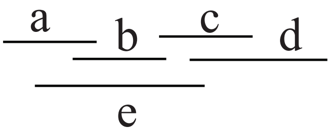
- Interval graphs are intersection graphs where the subsets are intervals/segments $[a, b]$ in $\mathbb{R}$

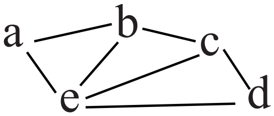
Interval Graphs (a type of intersection graph)

- Interval graphs are intersection graphs where the subsets are intervals/segments $[a, b]$ in $\mathbb{R}$
- Any graph that can be constructed this way is an interval graph

Interval Graphs (a type of intersection graph)

- Interval graphs are intersection graphs where the subsets are intervals/segments $[a, b]$ in $\mathbb{R}$
- Any graph that can be constructed this way is an interval graph





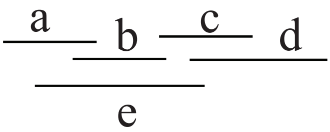
Interval Graphs (a type of intersection graph)

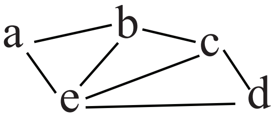
- Interval graphs are intersection graphs where the subsets are intervals/segments $[a, b]$ in $\mathbb{R}$
- Any graph that can be constructed this way is an interval graph

- Are all graphs interval graphs?

Interval Graphs (a type of intersection graph)

- Interval graphs are intersection graphs where the subsets are intervals/segments $[a, b]$ in $\mathbb{R}$
- Any graph that can be constructed this way is an interval graph





- Are all graphs interval graphs? 4-cycle

Interval Graphs

Theorem 7.4.3

All Interval Graphs are triangulated.

proof sketch.

Given interval graph $G = (V, E)$, consider any cycle $u, w_1, w_2, \ldots, w_k, v, u \in V(G)$. Cycle must go (w.l.o.g.) forward and then backwards along the line in order to connect back to u, so there must be a chord between some non-adjacent nodes (since they will overlap). □

Are all triangulated graphs interval graphs?

Interval Graphs

Theorem 7.4.3

All Interval Graphs are triangulated.

proof sketch.

Given interval graph $G = (V, E)$, consider any cycle $u, w_1, w_2, \ldots, w_k, v, u \in V(G)$. Cycle must go (w.l.o.g.) forward and then backwards along the line in order to connect back to u, so there must be a chord between some non-adjacent nodes (since they will overlap). □

Are all triangulated graphs interval graphs? No, consider spider graph (elongated star graph).

Interval Graphs

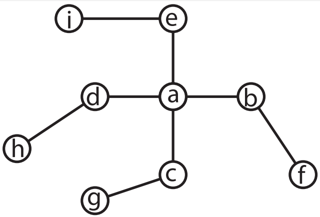
Theorem 7.4.3

All Interval Graphs are triangulated.

proof sketch.

Given interval graph $G = (V, E)$, consider any cycle $u, w_1, w_2, \ldots, w_k, v, u \in V(G)$. Cycle must go (w.l.o.g.) forward and then backwards along the line in order to connect back to u, so there must be a chord between some non-adjacent nodes (since they will overlap). $\square$

Are all triangulated graphs interval graphs? No, consider spider graph (elongated star graph).



Sub-tree intersection Graphs

- Given underlying tree, create intersection graph, where subsets are (nec. connected) subtrees of some "ground" tree.

Sub-tree intersection Graphs

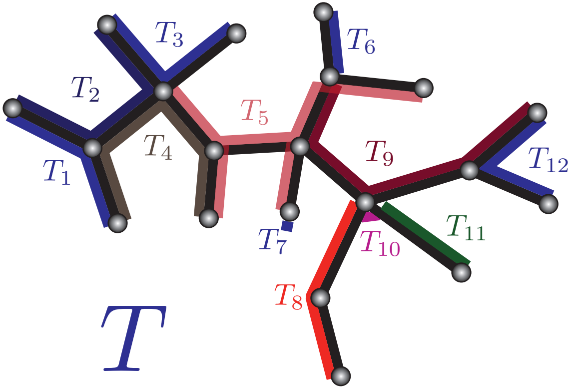
- Given underlying tree, create intersection graph, where subsets are (nec. connected) subtrees of some "ground" tree.
- Intersection exists if there are any nodes in common amongst the two corresponding trees.

Sub-tree intersection Graphs

- Given underlying tree, create intersection graph, where subsets are (nec. connected) subtrees of some "ground" tree.
- Intersection exists if there are any nodes in common amongst the two corresponding trees.

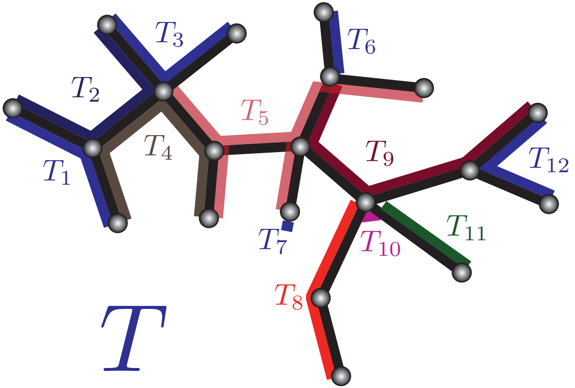
Sub-tree intersection Graphs

- Given underlying tree, create intersection graph, where subsets are (nec. connected) subtrees of some “ground” tree.
- Intersection exists if there are any nodes in common amongst the two corresponding trees.

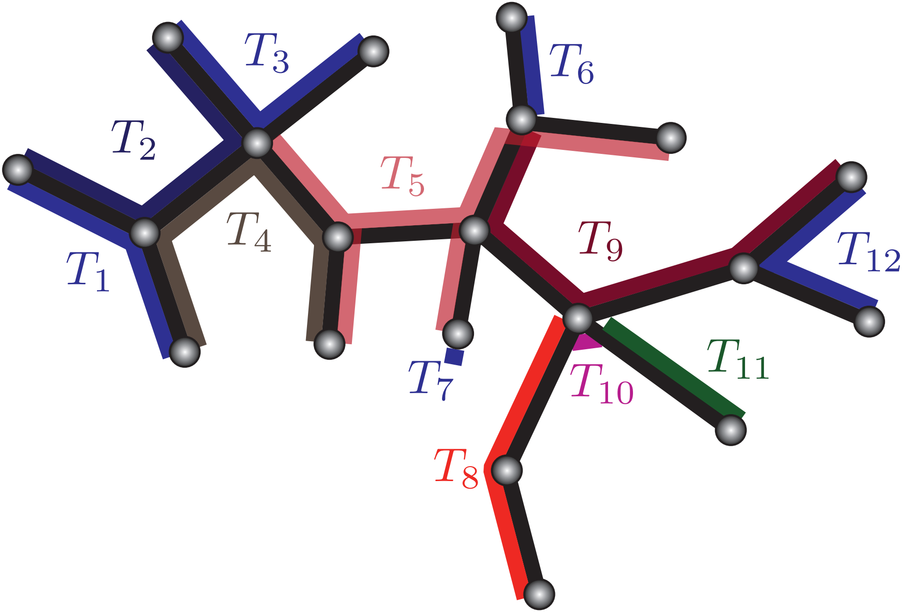


Sub-tree intersection Graphs

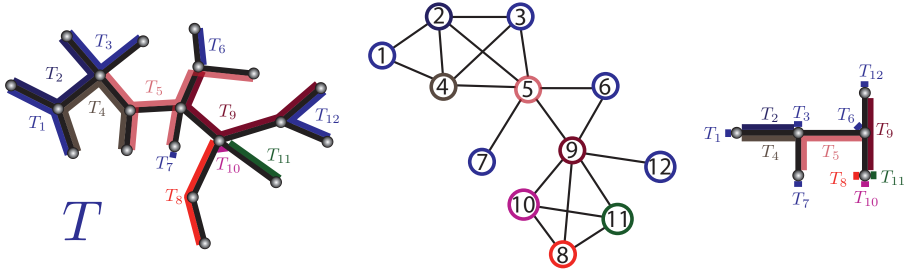
- Given underlying tree, create intersection graph, where subsets are (nec. connected) subtrees of some "ground" tree.
- Intersection exists if there are any nodes in common amongst the two corresponding trees.



Lets zoom in a little on this

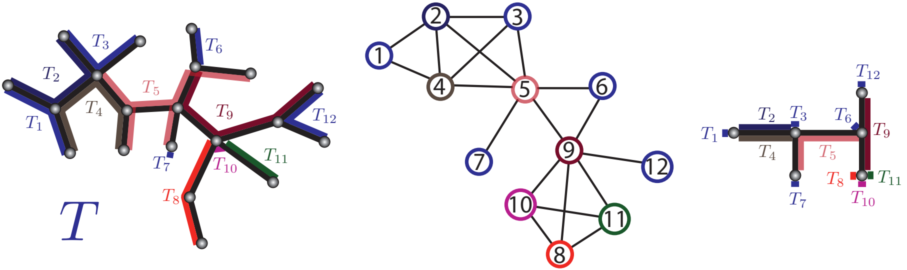

T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
T_{10}
T_{11}
T_{12}

Sub-tree intersection Graphs



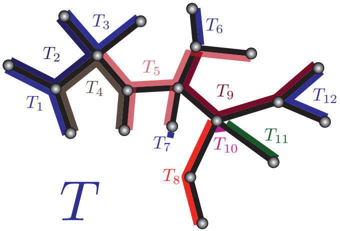
- Intersection exists if there are any nodes in common amongst the two corresponding trees.

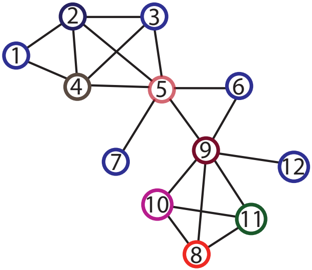
Sub-tree intersection Graphs

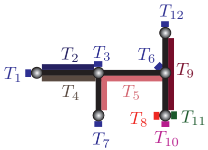


- Intersection exists if there are any nodes in common amongst the two corresponding trees.
- A sub-tree graph corresponds to more than one underlying tree (thus ground set and underlying subsets).

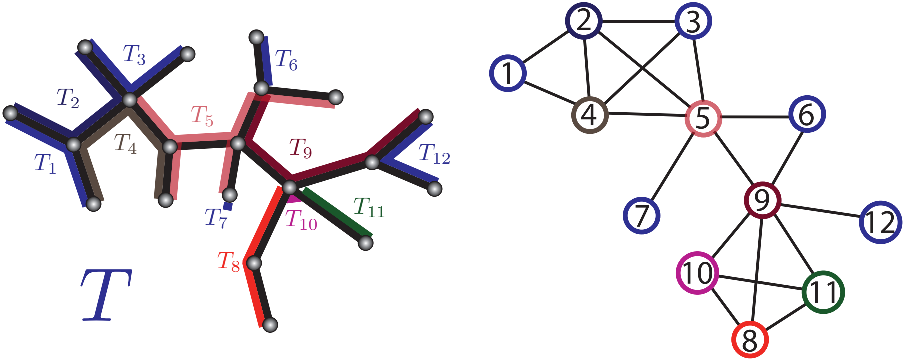
Sub-tree intersection Graphs

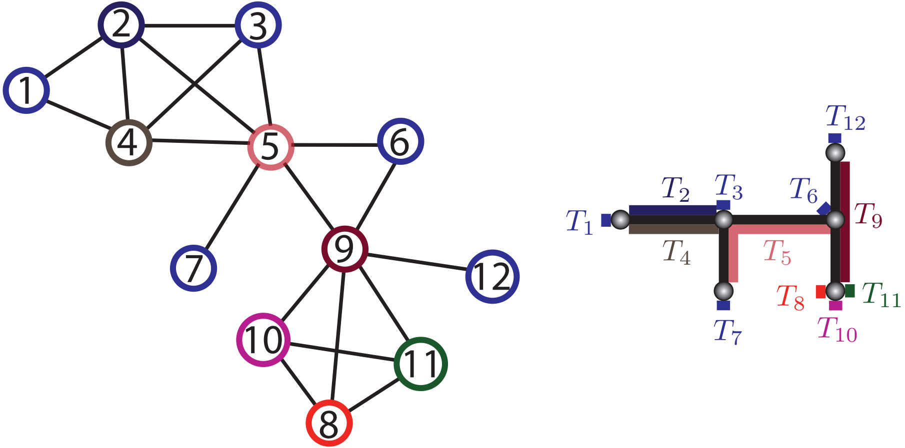




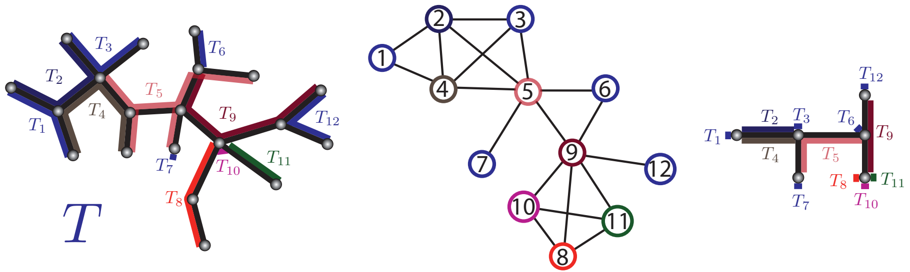


- Intersection exists if there are any nodes in common amongst the two corresponding trees.
- A sub-tree graph corresponds to more than one underlying tree (thus ground set and underlying subsets).
- What is the difference between left and right trees?


T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
T_{10}
T_{11}
T_{12}
T
1
2
3
4
5
6
7
8
9
10
11
12

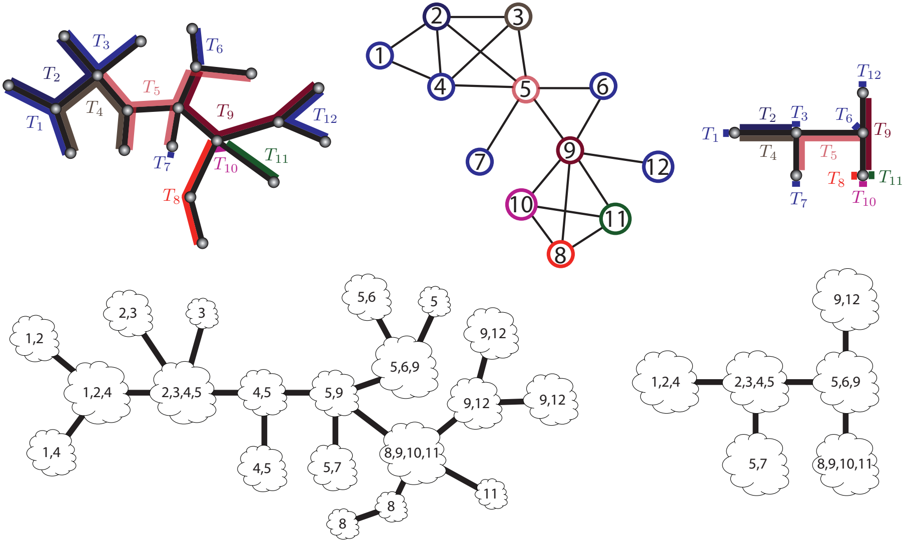

1
2
3
4
5
6
7
8
9
10
11
12
T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
T_{10}
T_{11}
T_{12}

Sub-tree intersection Graphs



- Intersection exists if there are any nodes in common amongst the two corresponding trees.
- A sub-tree graph corresponds to more than one underlying tree (thus ground set and underlying subsets).
- What is the difference between left and right trees?
- Junction tree of cliques and maxcliques vs. junction tree of just maxcliques.

Sub-tree intersection Graphs w. Junction Trees



Sub-tree intersection graphs

Theorem 7.4.4

A graph $G = (V, E)$ is triangulated iff it corresponds to a sub-tree graph (i.e., an intersection graph on subtrees of some tree).

proof sketch.

We see that any sub-tree graph is such that nodes in the tree correspond to cliques in G, and by the nature of how the graph is constructed (subtrees of some underlying tree), the tree corresponds to a cluster tree that satisfies the induced subtree property. Therefore, any sub-tree graph corresponds to a junction tree, and any corresponding graph G is triangulated. □

Sub-tree intersection graphs

- All interval graphs are sub-tree intersection graphs (underlying tree is a chain, subtrees are sub-chains)

Sub-tree intersection graphs

- All interval graphs are sub-tree intersection graphs (underlying tree is a chain, subtrees are sub-chains)
- Are all sub-tree intersection graphs interval graphs?

Sub-tree intersection graphs

- All interval graphs are sub-tree intersection graphs (underlying tree is a chain, subtrees are sub-chains)
- Are all sub-tree intersection graphs interval graphs?
- So sub-tree intersection graphs capture the "tree-like" nature of triangulated graphs.

Sub-tree intersection graphs

- All interval graphs are sub-tree intersection graphs (underlying tree is a chain, subtrees are sub-chains)
- Are all sub-tree intersection graphs interval graphs?
- So sub-tree intersection graphs capture the “tree-like” nature of triangulated graphs.
- Triangulated graphs are also called hyper-trees (specific type of hyper-graph, where edges are generalized to be clusters of nodes rather than 2 nodes in a normal graph). In hyper-tree, the unique “max-edge” path between any two nodes property is generalized.

Inference on JTs.

- We can define an inference procedure on junction trees that corresponds to our inference procedure on trees.

Inference on JTs.

- We can define an inference procedure on junction trees that corresponds to our inference procedure on trees.
- We are given $p \in \mathcal{F}(G', \mathcal{M}^{(\mathrm{f})})$, where G' is triangulated. It might be naturally triangulated, might be an MRF for which we've found a good elimination order, or might even have come from a triangulated moralized Bayesian network. In either case, if we solve inference for the family $\mathcal{F}(G', \mathcal{M}^{(\mathrm{f})})$ we've solved it for the original graph.

Inference on JTs.

- We can define an inference procedure on junction trees that corresponds to our inference procedure on trees.
- We are given $p \in \mathcal{F}(G', \mathcal{M}^{(\mathrm{f})})$, where G' is triangulated. It might be naturally triangulated, might be an MRF for which we've found a good elimination order, or might even have come from a triangulated moralized Bayesian network. In either case, if we solve inference for the family $\mathcal{F}(G', \mathcal{M}^{(\mathrm{f})})$ we've solved it for the original graph.
- Let G be the original graph with cliques $\mathcal{C}(G)$, and let $\mathcal{C}(G')$ be the cliques of the triangulated graph.

Inference on JTs.

- We can define an inference procedure on junction trees that corresponds to our inference procedure on trees.
- We are given $p \in \mathcal{F}(G', \mathcal{M}^{(\mathrm{f})})$, where G' is triangulated. It might be naturally triangulated, might be an MRF for which we've found a good elimination order, or might even have come from a triangulated moralized Bayesian network. In either case, if we solve inference for the family $\mathcal{F}(G', \mathcal{M}^{(\mathrm{f})})$ we've solved it for the original graph.
- Let G be the original graph with cliques $\mathcal{C}(G)$, and let $\mathcal{C}(G')$ be the cliques of the triangulated graph.
- We know we have factorization:

$$p(x) = \prod_{C \in \mathcal{C}(G)} \psi_C(x_C) \tag{7.1}$$

Inference on JTs.

- Every clique $C \in \mathcal{C}(G)$ is contained in at least one clique $C' \in \mathcal{C}(G')$.

Inference on JTs.

- Every clique $C \in \mathcal{C}(G)$ is contained in at least one clique $C' \in \mathcal{C}(G')$.
- Therefore, each factor $\psi_C(x_C)$ for $C \in \mathcal{C}(G)$ can be assigned to a new factor $\psi_{C'}(x_{C'})$ for some $C' \in \mathcal{C}(G')$.

Inference on JTs.

- Every clique $C \in \mathcal{C}(G)$ is contained in at least one clique $C' \in \mathcal{C}(G')$.
- Therefore, each factor $\psi_C(x_C)$ for $C \in \mathcal{C}(G)$ can be assigned to a new factor $\psi_{C'}(x_{C'})$ for some $C' \in \mathcal{C}(G')$.
- Given that we have a junction tree of maxcliques, we are going to allocate "storage" for maxclique potentials $\psi_{C'}(x_{C'})$ for all $C' \in \mathcal{C}(G')$ (equivalently all nodes in the junction tree).

Inference on JTs.

- Every clique $C \in \mathcal{C}(G)$ is contained in at least one clique $C' \in \mathcal{C}(G')$.
- Therefore, each factor $\psi_C(x_C)$ for $C \in \mathcal{C}(G)$ can be assigned to a new factor $\psi_{C'}(x_{C'})$ for some $C' \in \mathcal{C}(G')$.
- Given that we have a junction tree of maxcliques, we are going to allocate "storage" for maxclique potentials $\psi_{C'}(x_{C'})$ for all $C' \in \mathcal{C}(G')$ (equivalently all nodes in the junction tree).
- We are also going to allocate storage for all separators in the junction tree. That is, we will have a function $\phi_S(x_S)$ for all $S \in \mathcal{S}(G')$ where $S(G')$ are the set of separators in the junction tree corresponding to triangulated graph G'.

Inference on JTs.

- Every clique $C \in \mathcal{C}(G)$ is contained in at least one clique $C' \in \mathcal{C}(G')$.
- Therefore, each factor $\psi_C(x_C)$ for $C \in \mathcal{C}(G)$ can be assigned to a new factor $\psi_{C'}(x_{C'})$ for some $C' \in \mathcal{C}(G')$.
- Given that we have a junction tree of maxcliques, we are going to allocate "storage" for maxclique potentials $\psi_{C'}(x_{C'})$ for all $C' \in \mathcal{C}(G')$ (equivalently all nodes in the junction tree).
- We are also going to allocate storage for all separators in the junction tree. That is, we will have a function $\phi_S(x_S)$ for all $S \in \mathcal{S}(G')$ where $S(G')$ are the set of separators in the junction tree corresponding to triangulated graph G'.
- We need to know how to initialize these separators.

Inference on JTs - table initialization

- Initialization Step: For each $C' \in \mathcal{C}(G')$, assign $\psi_{C'}(x_{C'}) = 1$.

Inference on JTs - table initialization

- Initialization Step: For each $C' \in \mathcal{C}(G')$, assign $\psi_{C'}(x_{C'}) = 1$.
- For each clique $C \in \mathcal{C}(G)$, find one $C' \in \mathcal{C}(G')$ such that $C \subseteq C'$, and update $\psi_{C'}(x_{C'})$ as follows:

$$\psi_{C'}(x_{C'}) \leftarrow \psi_{C'}(x_{C'})\psi_C(x_C) \tag{7.2}$$

Inference on JTs - table initialization

- Initialization Step: For each $C' \in \mathcal{C}(G')$, assign $\psi_{C'}(x_{C'}) = 1$.
- For each clique $C \in \mathcal{C}(G)$, find one $C' \in \mathcal{C}(G')$ such that $C \subseteq C'$, and update $\psi_{C'}(x_{C'})$ as follows:

$$\psi_{C'}(x_{C'}) \leftarrow \psi_{C'}(x_{C'})\psi_C(x_C) \tag{7.2}$$

- Crucial: Only do this once, otherwise, we'll be double counting the clique $\psi_C(x_C)$.

Inference on JTs - table initialization

- Initialization Step: For each $C' \in \mathcal{C}(G')$, assign $\psi_{C'}(x_{C'}) = 1$.
- For each clique $C \in \mathcal{C}(G)$, find one $C' \in \mathcal{C}(G')$ such that $C \subseteq C'$, and update $\psi_{C'}(x_{C'})$ as follows:

$$\psi_{C'}(x_{C'}) \leftarrow \psi_{C'}(x_{C'})\psi_C(x_C) \tag{7.2}$$

- Crucial: Only do this once, otherwise, we'll be double counting the clique $\psi_C(x_C)$.
- We now have the following representation of $p \in \mathcal{F}(G, \mathcal{M}^{(\mathrm{f})})$:

$$p(x) = \prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'}) \tag{7.3}$$

Inference on JTs - table initialization

- Initialization Step: For each $C' \in \mathcal{C}(G')$, assign $\psi_{C'}(x_{C'}) = 1$.
- For each clique $C \in \mathcal{C}(G)$, find one $C' \in \mathcal{C}(G')$ such that $C \subseteq C'$, and update $\psi_{C'}(x_{C'})$ as follows:

$$\psi_{C'}(x_{C'}) \leftarrow \psi_{C'}(x_{C'})\psi_C(x_C) \tag{7.2}$$

- Crucial: Only do this once, otherwise, we'll be double counting the clique $\psi_C(x_C)$.
- We now have the following representation of $p \in \mathcal{F}(G, \mathcal{M}^{(\mathrm{f})})$:

$$p(x) = \prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'}) \tag{7.3}$$

- We also initialize all separators by doing $\phi_S(x_S) = 1 \ \forall S$.

Inference on JTs - table initialization

- Initialization Step: For each $C' \in \mathcal{C}(G')$, assign $\psi_{C'}(x_{C'}) = 1$.
- For each clique $C \in \mathcal{C}(G)$, find one $C' \in \mathcal{C}(G')$ such that $C \subseteq C'$, and update $\psi_{C'}(x_{C'})$ as follows:

$$\psi_{C'}(x_{C'}) \leftarrow \psi_{C'}(x_{C'})\psi_C(x_C) \tag{7.2}$$

- Crucial: Only do this once, otherwise, we'll be double counting the clique $\psi_C(x_C)$.
- We now have the following representation of $p \in \mathcal{F}(G, \mathcal{M}^{(\mathrm{f})})$:

$$p(x) = \prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'}) \tag{7.3}$$

- We also initialize all separators by doing $\phi_S(x_S) = 1 \ \forall S$.
- Once this is done, we have

$$p(x) = \frac{\prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'})}{\prod_{S \in \mathcal{S}(G')} \phi_S(x_S)^{d(S)-1}} \tag{7.4}$$

Maxclique marginals as the goal

- Since G' is triangulated, and is decomposable, we know it is possible to represent p as:

$$p(x) = \prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'}) = \frac{\prod_{C \in \mathcal{C}'} p(x_{C'})}{\prod_{S \in \mathcal{S}(G')} p(x_S)^{d(S)-1}} \tag{7.5}$$

where $d(S)$ is the shattering coefficient of separator S.

Maxclique marginals as the goal

- Since G' is triangulated, and is decomposable, we know it is possible to represent p as:

$$p(x) = \prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'}) = \frac{\prod_{C \in \mathcal{C}'} p(x_{C'})}{\prod_{S \in \mathcal{S}(G')} p(x_S)^{d(S)-1}} \tag{7.5}$$

  where $d(S)$ is the shattering coefficient of separator S.
- If we set $\phi_S(x_S) = 1$ for all S, then

$$p(x) = \frac{\prod_{C \in \mathcal{C}(G')} \psi_C(x_C)}{\prod_{S \in \mathcal{S}(G')} \phi_S(x_S)^{d(S)-1}} \tag{7.6}$$

Maxclique marginals as the goal

- Since G' is triangulated, and is decomposable, we know it is possible to represent p as:

$$p(x) = \prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'}) = \frac{\prod_{C \in \mathcal{C}'} p(x_{C'})}{\prod_{S \in \mathcal{S}(G')} p(x_S)^{d(S)-1}} \tag{7.5}$$

  where $d(S)$ is the shattering coefficient of separator S.

- If we set $\phi_S(x_S) = 1$ for all S, then

$$p(x) = \frac{\prod_{C \in \mathcal{C}(G')} \psi_C(x_C)}{\prod_{S \in \mathcal{S}(G')} \phi_S(x_S)^{d(S)-1}} \tag{7.6}$$

- In Equation 7.8, we have the functions at each maxclique and at each separator equal to the **marginal distribution** over the corresponding nodes.

Maxclique marginals as the goal

- With the marginals, we can easily compute any desired original-graph clique marginal for any $C \in \mathcal{C}(G)$.

Maxclique marginals as the goal

- With the marginals, we can easily compute any desired original-graph clique marginal for any $C \in \mathcal{C}(G)$.
- Our goal is to efficiently go from the representation at Equation 7.7 to the representation at the right of Equation 7.8.

Maxclique marginals as the goal

- With the marginals, we can easily compute any desired original-graph clique marginal for any $C \in \mathcal{C}(G)$.
- Our goal is to efficiently go from the representation at Equation 7.7 to the representation at the right of Equation 7.8.
- Can we do this using a similar message passing procedure to what we've already seen?

Maxclique marginals as the goal

- Start out (after initialization) with the expression

$$p(x) = \frac{\prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'})}{\prod_{S \in \mathcal{S}(G')} \phi_S(x_S)^{d(S)-1}} \tag{7.7}$$

- Do message passing, so that we end up with

$$p(x) = \frac{\prod_{C' \in \mathcal{C}(G')} \psi_{C'}(x_{C'})}{\prod_{S \in \mathcal{S}(G')} \phi_S(x_S)^{d(S)-1}} = \frac{\prod_{C \in \mathcal{C}'} p(x_{C'})}{\prod_{S \in \mathcal{S}(G)} p(x_S)^{d(S)-1}} \tag{7.8}$$

- Meaning, $\psi_{C'}(x_{C'}) = p(x_{C'})$ for all C' and $\phi_S(x_S) = p(x_S)$ for all S, marginals.

Maxclique marginals as the goal

- We do this using a junction tree (which we know to exist over the cliques and/or maxcliques of G'). So form a junction tree.

Maxclique marginals as the goal

- We do this using a junction tree (which we know to exist over the cliques and/or maxcliques of G'). So form a junction tree.
- Goal (again) is for the clique and separator functions to equal marginals.

Maxclique marginals as the goal

- We do this using a junction tree (which we know to exist over the cliques and/or maxcliques of G'). So form a junction tree.
- Goal (again) is for the clique and separator functions to equal marginals.
- What must be true of clique functions if they are marginals?

Maxclique marginals as the goal

- We do this using a junction tree (which we know to exist over the cliques and/or maxcliques of G'). So form a junction tree.
- Goal (again) is for the clique and separator functions to equal marginals.
- What must be true of clique functions if they are marginals? They must (at least) agree with what they have in common.

Maxclique marginals as the goal

- We do this using a junction tree (which we know to exist over the cliques and/or maxcliques of G'). So form a junction tree.
- Goal (again) is for the clique and separator functions to equal marginals.
- What must be true of clique functions if they are marginals? They must (at least) agree with what they have in common.
- Consider pair of neighboring cliques in a JT. Given maxclique C'_1 and C'_2 of $\mathcal{C}$, with $S = C'_1 \cap C'_2$, they must agree, i.e.,:

$$\sum_{x_{C'_1 \setminus S}} \psi_{C'_1}(x_{C'_1}) = \sum_{x_{C'_2 \setminus S}} \psi_{C'_2}(x_{C'_2}) \tag{7.9}$$

Maxclique marginals as the goal

- This is a necessary condition for the clique/separator functions to be marginals because

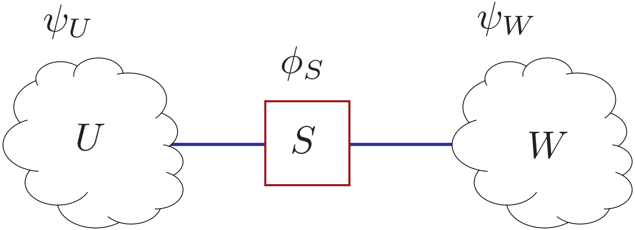
$$\sum_{x_{C_1'\setminus S}} \psi_{C_1'}(x_{C_1'}) = \sum_{x_{C_1'\setminus S}} p(x_{C_1'}) = \sum_{x_{C_2'\setminus S}} p(x_{C_2'}) = \sum_{x_{C_2'\setminus S}} \psi_{C_2'}(x_{C_2'}) \tag{7.10}$$

Maxclique marginals as the goal

- This is a necessary condition for the clique/separator functions to be marginals because

$$\sum_{x_{C_1'\setminus S}} \psi_{C_1'}(x_{C_1'}) = \sum_{x_{C_1'\setminus S}} p(x_{C_1'}) = \sum_{x_{C_2'\setminus S}} p(x_{C_2'}) = \sum_{x_{C_2'\setminus S}} \psi_{C_2'}(x_{C_2'}) \tag{7.10}$$

- Given two maxcliques U and W with separator $S = U \cap W$, and potential functions ψ_U, ψ_W, and ϕ_S, arranged in small JT as follows:



Maxclique marginals as the goal

- Shorthand notation: $\phi_S^* = \sum_{U \setminus S} \psi_U$ — represents new potential over separator S obtained from ψ_U where all but S has been marginalized away.

Maxclique marginals as the goal

- Shorthand notation: $\phi^*_S = \sum_{U \setminus S} \psi_U$ — represents new potential over separator S obtained from ψ_U where all but S has been marginalized away.
- Thus,

$$\sum_{U \setminus S} \psi_U \triangleq \sum_{x_{U \setminus S}} \psi_U(x_U) = \sum_{x_{U \setminus S}} \psi_U(x_{U \setminus S}, x_S) = \phi^*_S(x_S)$$

which is a function only of x_S.

Maxclique marginals as the goal

- More shorthand notation: table multiplication

$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W \tag{7.11}$$

Maxclique marginals as the goal

- More shorthand notation: table multiplication

$$\psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W \tag{7.11}$$

- Let $W_S = W \setminus S$, so that $W = S \cup W_S$. then

$$\psi_W = \psi_W(x_W) = \psi_W(x_S, x_{W_S}), \quad \phi_S = \phi_S(x_S) \tag{7.12}$$

Maxclique marginals as the goal

- More shorthand notation: table multiplication

$$\psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W \tag{7.11}$$

- Let $W_S = W \setminus S$, so that $W = S \cup W_S$. then

$$\psi_W = \psi_W(x_W) = \psi_W(x_S, x_{W_S}), \quad \phi_S = \phi_S(x_S) \tag{7.12}$$

and

$$\psi_W^* = \psi_W^*(x_W) = \psi_W^*(x_S, x_{W_S}), \quad \phi_S^* = \phi_S^*(x_S) \tag{7.13}$$

Maxclique marginals as the goal

- More shorthand notation: table multiplication

$$\psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W \tag{7.11}$$

- Let $W_S = W \setminus S$, so that $W = S \cup W_S$. then

$$\psi_W = \psi_W(x_W) = \psi_W(x_S, x_{W_S}), \quad \phi_S = \phi_S(x_S) \tag{7.12}$$

and

$$\psi_W^* = \psi_W^*(x_W) = \psi_W^*(x_S, x_{W_S}), \quad \phi_S^* = \phi_S^*(x_S) \tag{7.13}$$

so to expand everything out, we get

$$\psi_W^* = \psi_W^*(x_S, x_{W_S}) = \frac{\phi_S^*(x_S)}{\phi_S(x_S)}\psi_W(x_S, x_{W_S}) \tag{7.14}$$

Maxclique marginals as the goal

- Suppose, JT potentials start out inconsistent. i.e.,

$$\sum_{U \setminus S} \psi_U \neq \sum_{W \setminus S} \psi_W \quad \text{and} \quad \phi_S = 1 \tag{7.15}$$

but we still have that $p(x_U, x_W) = p(x_H, \bar{x}_E) = \psi_U \psi_W / \phi_S$.

Maxclique marginals as the goal

- Suppose, JT potentials start out inconsistent. i.e.,

$$\sum_{U \setminus S} \psi_U \neq \sum_{W \setminus S} \psi_W \quad \text{and} \quad \phi_S = 1 \tag{7.15}$$

  but we still have that $p(x_U, x_W) = p(x_H, \bar{x}_E) = \psi_U \psi_W / \phi_S$.
- Note (again) that we may treat evidence $\bar{x}_E$ as additional factors contained within a clique and that any summation would only sum over corresponding evidence value, so we can avoid mentioning evidence for now.

Maxclique marginals as the goal

- Suppose, JT potentials start out inconsistent. i.e.,

$$\sum_{U \setminus S} \psi_U \neq \sum_{W \setminus S} \psi_W \quad \text{and} \quad \phi_S = 1 \tag{7.15}$$

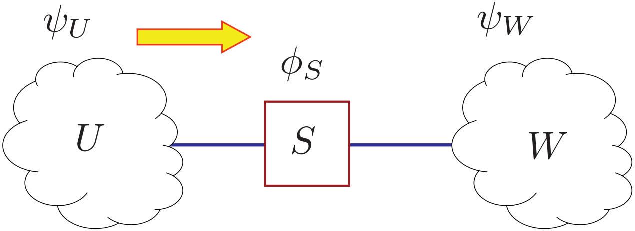
  but we still have that $p(x_U, x_W) = p(x_H, \bar{x}_E) = \psi_U \psi_W / \phi_S$.
- Note (again) that we may treat evidence $\bar{x}_E$ as additional factors contained within a clique and that any summation would only sum over corresponding evidence value, so we can avoid mentioning evidence for now.
- What we'll do: exchange information between cliques via separators to achieve consistency.

Maxclique marginals as the goal

- **Marginalize U:**

$$\phi_S^* = \sum_{U \setminus S} \psi_U \tag{7.16}$$

which leads to a new separator potential ϕ_S^* and can be seen as a partial message, as shown in the following figure

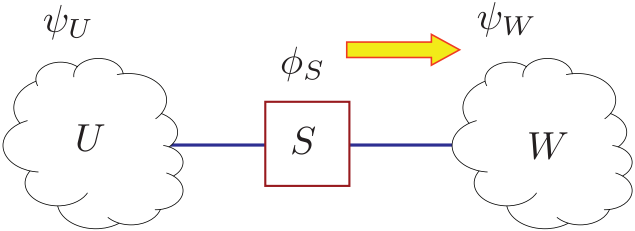


Maxclique marginals as the goal

- **Rescale W:**

$$\psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W \tag{7.17}$$

This produces a new potential on W based on the updated separator potential at S. This can also be seen as a partial message.



Maxclique marginals as the goal

- After these ops, joint has not changed: define $\psi_U^* = \psi_U$ for convenience, we get:

$$\frac{\psi_U^* \psi_W^*}{\phi_S^*} = \frac{\psi_U \psi_W \phi_S^*}{\phi_S \phi_S^*} = \frac{\psi_U \psi_W}{\phi_S} \tag{7.18}$$

Maxclique marginals as the goal

- After these ops, joint has not changed: define $\psi^*_U = \psi_U$ for convenience, we get:

$$\frac{\psi^*_U \psi^*_W}{\phi^*_S} = \frac{\psi_U \psi_W \phi^*_S}{\phi_S \phi^*_S} = \frac{\psi_U \psi_W}{\phi_S} \tag{7.18}$$

- Don't yet (nec.) have consistency since

$$\sum_{U \setminus S} \psi^*_U = \sum_{U \setminus S} \psi_U = \phi^*_S \neq \sum_{W \setminus S} \psi^*_W = \frac{\phi^*_S}{\phi_S} \sum_{W \setminus S} \psi_W \tag{7.19}$$

which follows because

$$\phi_S \neq \sum_{W \setminus S} \psi_W \tag{7.20}$$

Maxclique marginals as the goal

- We do at least have one marginal at ψ_W^*. This is because we started with:

$$p(x) = p(x_U, x_W) = \frac{\psi_U \psi_W}{\phi_S} \tag{7.21}$$

and

$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W = \psi_W \sum_{U \setminus S} \psi_U = \sum_{x_{U \setminus S}} p(x_H, \bar{x}_E) = p(x_W) \tag{7.22}$$

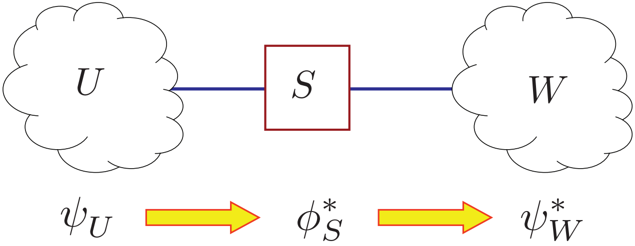
is one of the marginals that we desire.

Maxclique marginals as the goal

- We see this as a message passing procedure, passing a message between two nodes in a cluster tree.

Maxclique marginals as the goal

- We see this as a message passing procedure, passing a message between two nodes in a cluster tree.
- Message from cluster U through S and to W is the message directly from U to W (but done in two steps).



Sources for Today's Lecture

- Most of this material comes from the reading handout `tree_inference.pdf`