COMP538: Introduction to Bayesian Networks

Lecture 5: Inference as Message Propagation

Nevin L. Zhang
lzhang@cse.ust.hk

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

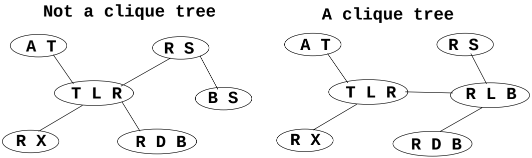
Fall 2008

Objectives

- Discusses another commonly used inference algorithm called clique tree propagation that
  - Is based on the same principle as VE except with a sophisticated caching strategy that
    - Enables one to compute the posterior probability distributions of all variables in twice the time it takes to compute that of one single variable.
    - Works in an intuitively appealing fashion, namely message propagation
- Readings: Zhang and Guo, Chapter 5
- Zhang (1998), Jensen *et al* 1990, Shafer and Shenoy (1990).

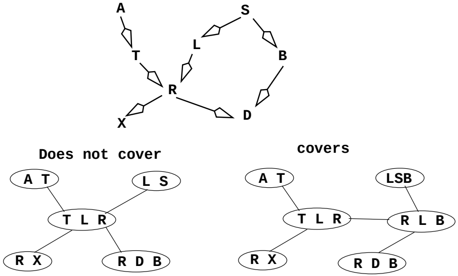
Outline

Clique trees



- A **clique tree** is an undirected tree,
  - Where each node represents a set of variables, which is called a **clique**.
  - That is **Variable-connected**:
    - If a variable appear in two cliques, it must appear in all cliques on the path between those two cliques.
    - That is, subgraph of cliques containing a given variable is connected.

Clique Trees



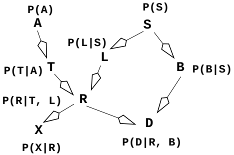
A clique tree **covers** a Bayesian network if

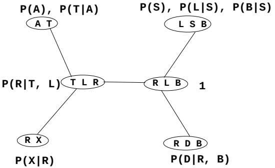
- The union of the cliques is the set of variables in the Bayesian network, and
- For any variable X in the Bayesian network, there is a clique that contains the variable and all its parents.
  - That clique is called the family cover clique of X.

Outline

Idea and Initialization

- Suppose we have a clique that covers a Bayesian network (BN).
- Idea: Use the clique tree to organize inference.
  - 5 steps.
- Step 1: Initialization:
  - For each variable X in BN,
    - Find a family cover clique **C** of X
    - Attach $P(X|pa(X))$ to the clique **C**.
  - If a clique is not attached with any function, attach the identity function to it.
  - Example:



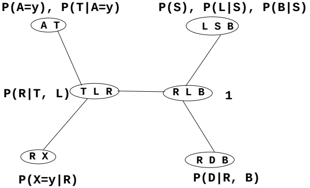


  - Multiplication of all functions on clique tree = joint distribution of all variables.

Step 2: Evidence absorption

Same as in VE.

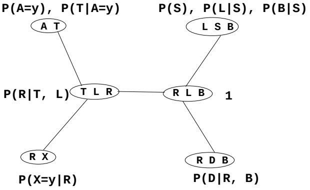
- Instantiate observed variables **E** in all functions.
- Example: Suppose $A{=}y$, $X{=}y$,



- Let **X** be the set of all unobserved variables:
  - Multiplication of all functions on clique tree $= P(\mathbf{X}, \mathbf{E} = \mathbf{e})$.
- This corresponds to Step 1 of VE.

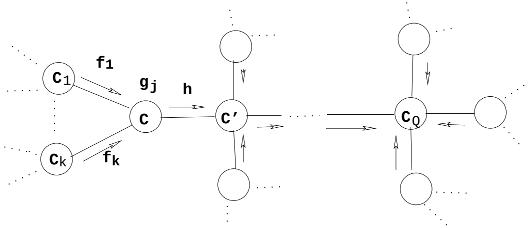
Step 3: Choose pivot for inference

- Suppose there is only one query variable Q.
- Find a clique $\mathbf{C}_Q$ that contains Q and use it as a **pivot** of inference.



- Example: Compute $P(L|A=y, X=y)$
  - Clique [RLB] can be used as a pivot.
  - So can [TLR] and [LSB]

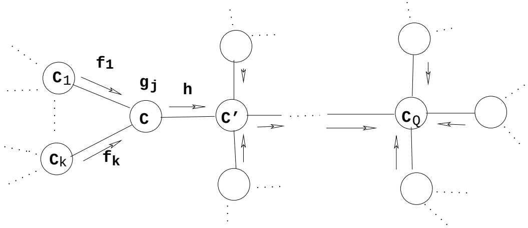
Step 4: Message Passing



- Messages are passed from the leaves toward the pivot.
- A clique $\mathbf{C}$ passes a message to the neighbor $\mathbf{C}'$ in the direction of the pivot after receiving messages from all other neighbors $\mathbf{C}_1, \ldots, \mathbf{C}_k$.
- Suppose f_i is the message from $\mathbf{C}_i$ to $\mathbf{C}$ and g_j are the functions attached to $\mathbf{C}$.
- The message from $\mathbf{C}$ to $\mathbf{C}'$ is the following function:

$$h(\mathbf{C}\cap\mathbf{C}' - \mathbf{E}) = \sum_{\mathbf{C}\setminus(\mathbf{C}'\cup\mathbf{E})} \prod_i f_i \prod_j g_j.$$

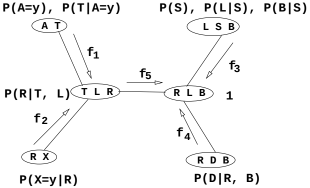
Step 5: Answer Extraction



- $h(Q, \mathbf{X})$ = product of all functions attached or sent to $\mathbf{C}_Q$
  - $\mathbf{X}$: set of unobserved variables in C_Q other than Q.
- Posterior probability of Q:

$$P(Q|\mathbf{E}=\mathbf{e}) = \sum_{\mathbf{X}} h(Q, \mathbf{X}) / \sum_{Q, \mathbf{X}} h(Q, \mathbf{X})$$

An Example



- Query: $P(L|A{=}y, X{=}y)$
- Clique [RLB] as pivot.

- Messages passed toward the pivot:

$$
\begin{aligned}
f_1(T) &= P(A{=}y)P(T|A{=}y) \\
f_2(R) &= P(X{=}y|R) \\
f_3(L,B) &= \sum_S P(S)P(L|S)P(B|S) \\
f_4(R,B) &= \sum_D P(D|R,B) \\
f_5(L,R) &= \sum_T f_1(T)f_2(R)P(R|T,L)
\end{aligned}
$$

- Extract answer:
  $h(R,L,B) = f_3(L,B)f_4(R,B)f_5(L,R)1$
  $(= P(R,L,B,A{=}y,X{=}y))$
  $P(L|A{=}y,X{=}y) =$
  $\sum_{R,B} h(R,L,B)/\sum_{R,L,B} h(R,L,B)$.

Complexity

- To send out a message, a clique C needs to compute the product of its attached functions and functions it has received.
- When there are no observations, variables in the product are all in the clique.
- If clique tree constructed using `buildCliqueTree` (to be given later), all variables in the clique are also in the product.
- Complexity of message-passing out of C can be measure by:

$$\prod_{X \in C} |X|$$

- Complexity of entire process can be measured by:

$$\sum_{C} \prod_{X \in C} |X|$$

- This is sometime dominated by the largest term.
- The complexity is exponential in the **maximum clique size**.

Outline

Program Invariants

A clique is **un-activated** is it has not sent out message.

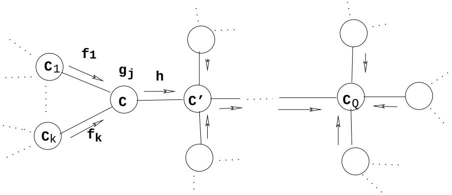
Proposition (5.1)

Program invariant: *The following two properties are preserved during message passing:*

1. *The arguments of the functions attached or sent to a clique all are variables in the clique.*
2. *Product of functions attached or sent to all un-activated cliques =* $P(\mathbf{X}, \mathbf{E}=\mathbf{e})$, *where* $\mathbf{X}$ *stands for all unobserved variables in those cliques.*

Proof of Proposition 5.1

- The properties are true before message passing starts.
- **Induction hypothesis**:
  - Suppose the properties hold before **C** sending message to **C**′.

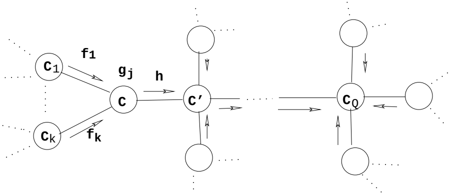


- **Induction**: After **C** sending message to **C**′:
  - It is clear that the message from **C** to **C**′:

$$h = \sum_{\mathbf{C} \setminus (\mathbf{C}' \cup \mathbf{E})} \prod_i f_i \prod_j g_j$$

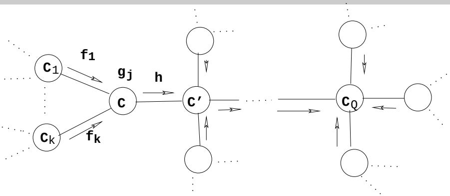
involves only variables in **C**′. Hence the first property remains true.

Proof of Proposition 5.1 (cont'd)



- **X**: set of unobserved variables in all un-activated cliques **before** **C** sending message to **C**$'$.
- **X**$'$: set of unobserved variables in all un-activated cliques **after** **C** sending message to **C**$'$.
- r: product of functions attached or sent to all un-activated cliques **after** **C** sending message to **C**$'$, **except** h.
- By induction hypothesis, r is a function of **X**$'$. Write it as $r(\mathbf{X}')$.
- Another description of $r(\mathbf{X}')$: the product of functions attached or sent to all un-activated cliques **before** **C** sending message to **C**$'$, **except** the f_i's and the g_j's.

Proof of Proposition 5.1 (cont'd)



- By the induction hypothesis, we have

$$r(\mathbf{X}') \prod_i f_i \prod_j g_j = P(\mathbf{X}, \mathbf{E} = \mathbf{e})$$

- Now consider the set of variables $\mathbf{C} \setminus (\mathbf{C}' \cup \mathbf{E})$.
  - For simplicity, assume the set contains only one variable Z.
- Because of variable-connectedness,
  - Z cannot appear in any cliques separated from $\mathbf{C}$ by $\mathbf{C}'$, which
  - include all cliques un-activated right after $\mathbf{C}$ sending message to $\mathbf{C}'$.
- Hence Z is not in the set $\mathbf{X}'$. So, $\mathbf{X} = \mathbf{X}' \cup \{Z\}$

Proof of Proposition 5.1 (cont'd)

- Hence

$$
\begin{aligned}
r(\mathbf{X}').h &= r(\mathbf{X}')\sum_{Z}\prod_{i} f_i \prod_{j} g_j \\
&= \sum_{Z} r(\mathbf{X}')\prod_{i} f_i \prod_{j} g_j \quad \text{(Fact } Z \notin \mathbf{X}' \text{ used here)} \\
&= \sum_{Z} P(\mathbf{X}, \mathbf{E}=\mathbf{e}) = \sum_{Z} P(Z, \mathbf{X}', \mathbf{E}=\mathbf{e}) = P(\mathbf{X}', \mathbf{E}=\mathbf{e})
\end{aligned}
$$

The proposition is proved. Q.E.D

Correctness of VE on Clique Trees

Theorem (5.1)

Let

- $\mathbf{X}$ *stands for the set of unobserved variables in* $\mathbf{C}_Q$ *except* Q*,*
- $h(Q, \mathbf{X})$ = *product of all functions attached or sent to* $\mathbf{C}_Q$ *at the end of message passing,*

Then

$$h(Q, \mathbf{X}) = P(Q, \mathbf{X}, \mathbf{E} = \mathbf{e})$$

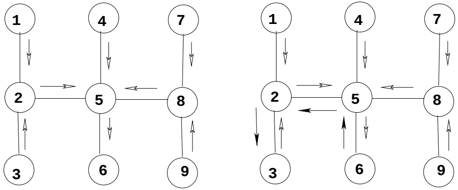
Consequently

$$P(Q|\mathbf{E} = \mathbf{e}) = \sum_{\mathbf{X}} h(Q, \mathbf{X}) / \sum_{Q,\mathbf{X}} h(Q, \mathbf{X})$$

Proof: The Theorem follows readily from Proposition 5.1.

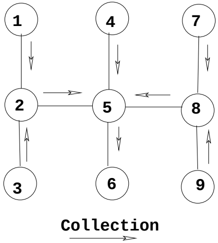
Outline

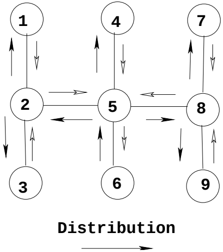
Computation sharing in clique tree



- Suppose messages have been propagated toward clique 6.
- Now consider propagating messages toward clique 3.
  - The following message passing steps from the first propagation can be reused:
    - $1 \rightarrow 2$, $9 \rightarrow 8$, $7 \rightarrow 8$, $8 \rightarrow 5$, $4 \rightarrow 5$
  - Only need to do:
    - $6 \rightarrow 5$, $5 \rightarrow 2$, $2 \rightarrow 3$
- Computation sharing opportunities exist between any two queries.

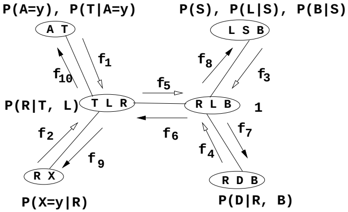
The Clique Tree Propagation Algorithm





- Characteristics:
  - Exploits computation sharing opportunities.
  - Computes posterior probabilities of **all** unobserved variables.
- Several variations. More or less equivalent.
- The algorithm: Two sweep message passing.
  - In the first sweep, called **collection**:
    - Messages are passed from leaves toward a pivot clique.
    - Exactly the same as VE on clique trees.
  - In the second sweep, called **distribution**:
    - Messages are passed from the pivot clique toward the leaves
  - Answer extraction: The same as in VE on clique trees and applied to very unobserved variables (or multiple query variables).

Example



- Collection: Messages propagated from leaves to [RLB]. (Done before)
- Distribution: Message propagated from [RLB] to leaves.

$$
\begin{aligned}
f_6(R,L) &= \sum_B f_4(R,B) f_3(L,B) 1 \\
f_7(R,B) &= \sum_L f_5(R,L) f_3(L,B) 1 \\
f_8(L,B) &= \sum_R f_4(R,B) f_5(R,L) 1 \\
f_9(R) &= \sum_{T,L} f_6(R,L) f_1(T) P(R|T,L) \\
f_{10}(T) &= \sum_{L,R} f_6(R,L) f_2(R) P(R|T,L)
\end{aligned}
$$

- **Note**: When computing the message from [RLB] to [TLR], we combine only two of the messages received by [RLB], f_3 and f_4. f_5 is not included.

Clique Tree Propagation

$CTP(\mathcal{T}, \mathbf{E}, \mathbf{e})$
Input: $\mathcal{T}$ — Clique, initialized, evidence absorbed
Output $P(X|\mathbf{E}=\mathbf{e})$ of every non-observed variable X

1: Pick one clique $\mathbf{C_P}$ as the pivot
2: **for** (each neighbor $\mathbf{C}$ of $\mathbf{C_P}$)
3: Call *CollectMessage*$(\mathbf{C_P}, \mathbf{C})$// $\mathbf{C_P} \leftarrow \mathbf{C}$
4: **end for**
5: **for** (each neighbor $\mathbf{C}$ of $\mathbf{C_P}$)
6: Call *DistributeMessage*$(\mathbf{C_P}, \mathbf{C})$// $\mathbf{C_P} \rightarrow \mathbf{C}$
7: **end for**
8: Extract posterior distribution of each non-observed variable.

Clique Tree Propagation

CollectMessage$(\mathbf{C}, \mathbf{C}')$ // $\mathbf{C} \leftarrow \mathbf{C}'$

1: **for**(each neighbor $\mathbf{C}''$ of $\mathbf{C}'$ except $\mathbf{C}$)
2: *CollectMessage*$(\mathbf{C}', \mathbf{C}'')$
3: **end for**
4: *SendMessage*$(\mathbf{C}', \mathbf{C})$

DistributeMessage$(\mathbf{C}, \mathbf{C}')$ // $\mathbf{C} \rightarrow \mathbf{C}'$

1: *SendMessage*$(\mathbf{C}, \mathbf{C}')$
2: **for**(each neighbor $\mathbf{C}''$ of $\mathbf{C}'$)
3: *DistributeMessage*$(\mathbf{C}', \mathbf{C}'')$
4: **end for**

Clique Tree Propagation

SendMessage$(\mathbf{C}', \mathbf{C})$ // $\mathbf{C}' \rightarrow \mathbf{C}$

1: Suppose $\mathbf{C_1}', \mathbf{C_2}', \cdots, \mathbf{C_k}'$ are all the neighbors of $\mathbf{C}'$ except $\mathbf{C}$

2: For $i = 1, 2, \cdots, k$, $g_i \leftarrow$ *RetrieveMessage*$(\mathbf{C_i}', \mathbf{C}',)$;

3: Let $f_1, f_2, \cdots, f_l$ be the function stored at $\mathbf{C}'$ during initializataionand $\mathbf{Z} = \mathbf{C}' \setminus \mathbf{C} \cup \mathbf{E}$

4: $\psi \leftarrow \sum_{\mathbf{Z}} \prod_{i=1}^{l} f_i \prod_{j=1}^{k} g_j$

5: *SaveMessage*$(\mathbf{C}', \mathbf{C}, \psi)$

Clique Tree Propagation

Example: CTP on the clique tree shown on Slide 22

- CTP: Line 1: Pick pivot, Say Clique 5.
- CTP: Lines 2-4: For loop
  - CM(5, 8): (CM – CollectMessage)
    - For-loop:
      CM(8, 7):
      SM(7, 8) (SM – SendMessage)
      Multiply functions stored at 8, Compute and save M(7->8)
      CM(8, 9): compute and save M(9->8)
    - SM(8, 5):
      M(7->8) and M(9->8) retrieved
      Combine with functions stored at 8
      Compute and save M(8->5)
  - CM(5, 2): Compute and save M(2->5), M(1->2), M(3->5)
  - CM(5, 4), CM(5, 6): Compute and save M(4->5), M(6->5)

Clique Tree Propagation

- CTP: Lines 5-7, for-loop
  - DM(5, 8):
    - SM(5, 8)
      Combine M(2->5), M(4->5), M(6->5), with functions at 5
      Compute and save M(5->8) "
    - For-loop
      DM(8, 7): Save and compute M(8->7)
      DM(8, 9): compute and save M(9->7)
  - DM(5, 2): Compute and save M(5->2), M(2->1), M(2->3)
  - DM(5, 4), DM(5, 6): Compute and save M(4->4), M(5->6)
- CTP Line 8: Every clique has received message from all neighbors. So we can extract posterior probability of any variable X in a clique that contains X.

VE versus Clique Tree Propagation

- VE:
  - Answers one query at a time.
  - Allows pruning of irrelevant variables.
  - No computation sharing among different queries.
- Clique tree propagation:
  - Computes posterior probabilities of all unobserved variables.
  - Does not allow pruning of irrelevant variables.
  - Allows computation sharing among different queries.
- See empirical comparisons in Zhang (1998).
- Empirical results suggest that one should use clique tree propagation only when we want posterior probabilities of many unobserved variables.
- Think: How to compute MPE and MAP in a clique tree?
- BN softwares support either VE, or clique tree propagation, or both. Check the software link on course page. (JavaBayes, Genie/Smile, Netica, Hugin, ...)

Outline

Constructing Clique Trees

- Given: A Bayesian network.
- Task: Construct a clique tree
  - That covers the Bayesian network,
  - whose cliques are as small as possible.
- Solution: Build clique tree via elimination in moral graph.

An Algorithm

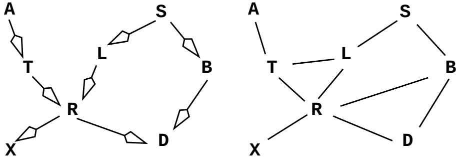
Let $\mathcal{G}$ be the moral graph of a BN and ρ be an elimination ordering.

Procedure `buildCliqueTree`$(\mathcal{G}, \rho)$

1. Remove the first node Z from ρ. Set $\mathbf{S} = adj(Z)$
2. Create clique $\mathbf{C} = \{Z\} \cup \mathbf{S}$
3. If $\mathbf{C}$ contains all nodes in $\mathcal{G}$, return the clique tree that consists of only one clique $\mathbf{C}$.
4. Else
    1. Add edges to $\mathcal{G}$ so that nodes in $\mathbf{S}$ are pairwise connected.
    2. Remove Z from $\mathcal{G}$.
    3. Recursive call: $\mathcal{T} = \texttt{buildCliqueTree}(\mathcal{G}, \rho)$
5. In $\mathcal{T}$, find clique $\mathbf{C}'$ s.t. $\mathbf{S} \subseteq \mathbf{C}'$ *(we will show that such clique must exist)*.
6. Add $\mathbf{C}$ to $\mathcal{T}$ by connecting it to $\mathbf{C}'$.
7. Return $\mathcal{T}$.

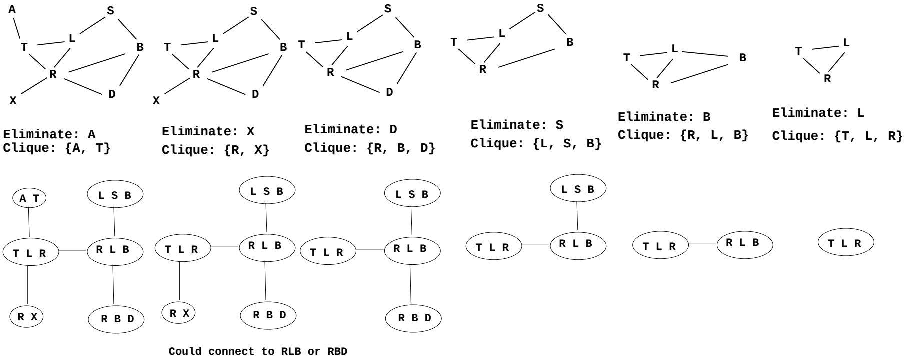
An Example

- BN and moral graph:



An Example (cont'd)

- Elimination ordering: A, X, D, S, B, L, T, R



g-Clique and t-cliques

- In an undirected graph, a set of nodes is a **clique** if vertices in the set are pairwise connected. (Standard graph-theoretic definition.)
- To avoid confusion:
  - Call such a clique a **g-clique**,
  - Call nodes in a clique tree **t-cliques**.

Proposition (5.2)

Let $\mathcal{G}$ be an undirected graph and $\mathcal{T}$ be the tree constructed by `buildCliqueTree` *for $\mathcal{G}$. If a set of variables $\mathbf{X}$ is a g-clique in $\mathcal{G}$, then there exists a t-clique $\mathbf{C}$ in $\mathcal{T}$ such that*

$$\mathbf{X} \subseteq \mathbf{C}$$

Proof:

- Let $\mathbf{C}$ be the clique created when eliminating the first node in $\mathbf{X}$.
- Then $\mathbf{X} \subseteq \mathbf{C}$. Q.E.D

Families covered

Corollary (5.2)

Let $\mathcal{G}$ be the moral graph of a BN and $\mathcal{T}$ be the tree constructed by `buildCliqueTree` *for $\mathcal{G}$. Then for any node X of the BN, there exists a t-clique* $\mathbf{C}$ *in $\mathcal{T}$ such that*

$$\{X\} \cup pa(X) \subseteq \mathbf{C}$$

Proof:

- $\{X\} \cup pa(X)$ is a g-clique in the moral graph $\mathcal{G}$.
- The corollary follows from Proposition 5.2. Q.E.D

Step 5 of Algorithm

Corollary (5.1)

Step 5 of `buildCliqueTree` *is always successful.*

Proof: Right after eliminating Z,

- $\mathbf{S}$ is a g-clique in $\mathcal{G}$.
- Let $\mathcal{T}$ be the tree constructed by the recursive call to `buildCliqueTree` right after the removal of Z.
- According to Proposition 5.2, there must be a clique $\mathbf{C}'$ in $\mathcal{T}$ s.t. $\mathbf{S} \subseteq \mathbf{C}'$. Q.E.D

Variable-Connectedness

Proposition (5.3)

The tree $\mathcal{T}$ constructed by `buildCliqueTree` *from undirected graph $\mathcal{G}$ is variable-connected.*

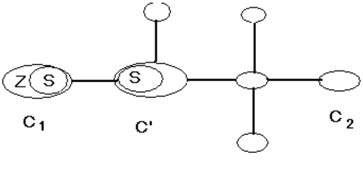
Proof:

- Induction on the number n of nodes in $\mathcal{G}$.
- When $n = 1$, the proposition is trivially true.
- **Induction hypothesis**: Assume the proposition is true when $n = k$.
- **Induction step**: Consider the case $n = k + 1$.
- Consider any two cliques $\mathbf{C}_1$ and $\mathbf{C}_2$ in $\mathcal{T}$ and suppose $X \in \mathbf{C}_1 \cap \mathbf{C}_2$.
- Need to show: X appears in all cliques on the path between $\mathbf{C}_1$ and $\mathbf{C}_2$.

Variable-Connectedness

- Let Z be the first variable eliminated.
- Let $\mathcal{T}'$ be the tree return by the first recursive call.
- According to the induction hypothesis, $\mathcal{T}'$ is variable-connected.
- $\mathcal{T}$ is $\mathcal{T}'$ plus the clique created when eliminating Z.
- If neither $\mathbf{C}_1$ nor $\mathbf{C}_2$ is the clique created when eliminating Z,
  - Then they are both in $\mathcal{T}'$.
  - Since $\mathcal{T}'$ is variable-connected, X appears in all cliques between $\mathbf{C}_1$ and $\mathbf{C}_2$.

Proof of Proposition 5.3 (cont'd)



- Now assume $\mathbf{C}_1$ is the clique created when eliminating Z.
- X cannot be Z because X is in $\mathbf{C}_2$.
- Then X must be in the set $\mathbf{S}$, i.e. $adj(Z)$ in the graph $\mathcal{G}$
- Let $\mathbf{C}'$ be the only neighbor of $\mathbf{C}_1$ (determined at step 5).
- Then, $\mathbf{S} \subseteq \mathbf{C}'$.
- Hence X must be in $\mathbf{C}'$, which is in $\mathcal{T}'$.
- Since $\mathcal{T}'$ is variable-connected, X appears in all cliques between $\mathbf{C}'$ and $\mathbf{C}_2$.
- Hence X appears in all cliques between $\mathbf{C}_1$ and $\mathbf{C}_2$. Q.E.D

Correctness of `buildCliqueTree`

Theorem (5.2)

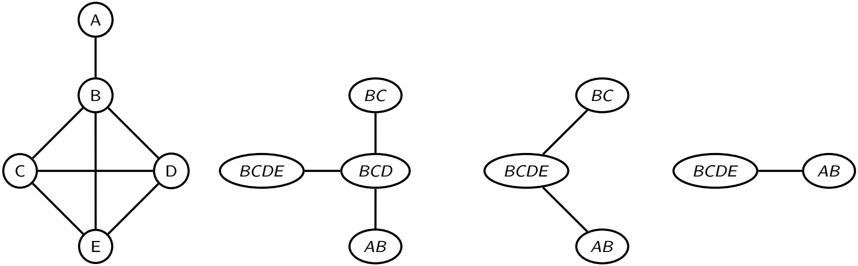
Let $\mathcal{G}$ be the moral graph of a BN and $\mathcal{T}$ be the tree constructed by `buildCliqueTree` *for $\mathcal{G}$. Then $\mathcal{T}$ is a clique tree that covers the Bayesian network.*

Proof:

- According to Proposition 5.3, $\mathcal{G}$ is a clique tree.
- According to Corollary 5.2, $\mathcal{G}$ covers the Bayesian network.

Minimal Clique Trees

- A clique tree is **minimal**: if none of the cliques are subsets of their neighbors.
- The tree obtained by `buildCliqueTree` might not be a minimal.
  - Example: Elimination ordering: E, D, C, B A



- As shown, can be easily made minimal.

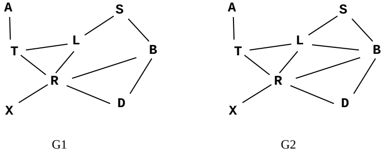
Outline

Why Link to Graph Theory

- I have explained the construction of clique trees in a way different from existing literature.
- Advantage: Easier to understand.
- Disadvantage: Intuition behind terminology (why clique tree) not clear.
- So it is necessary to explicate the link to graph theory.
- Also useful when reading papers.

Triangulated Graphs

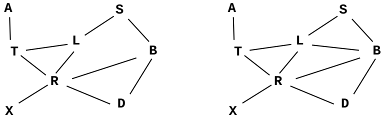
- An undirected graph is **triangulated (chordal)** if every cycle with four or more nodes contains a *chord* —An edge between two nonconsecutive nodes.
- Example:



  - G1 is not triangulated: Cycle S-L-R-B has no chords.
  - G2 is triangulated.

Triangulation

- **Triangulation**: Convert a graph that is not triangulated into one that is by adding edges.
- Example:

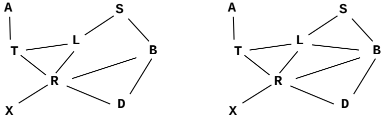


G1 G2

$G1$ is not triangulated.
Adding edge L-B, we get $G2$, which is triangulated.

Maximal Cliques

- A g-clique is **maximal** if none of its supersets are g-cliques.
- Example:



G1 G2

  - Maximal cliques of G1: [AT], [TLR], [XR], [RDB], [SL], [SB]
  - Maximal cliques of G2: [AT], [TLR], [XR], [RDB], [RLB], [SBL]
- Maximal cliques of a triangulated graph can be arranged into a clique tree.

Traditional Way to Build Clique Trees

$\mathcal{G}$: moral graph of a BN.

- Triangulate $\mathcal{G}$ by adding edges (equivalent to triangulation-via-elimination using an EO ρ) .
- Find all maximal g-cliques in triangulated graph.
- Arrange them into a tree.

The result is the same as that given by `buildCliqueTree` after minimization.