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
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.
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

1 Clique Trees

2 VE on Clique Trees

3 Correctness of VE on Clique Tree

4 The Clique Tree Propagation Algorithm

5 Constructing Clique Trees
   - Correctness

6 Link to Graph Theory
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 $\mathbf{E}$ in all functions.

- Example: Suppose $A=y$, $X=y$,

\[ P(A=y), P(T|A=y), P(S), P(L|S), P(B|S), P(R|T, L), P(D|R, B) \]

- Let $\mathbf{X}$ be the set of all unobserved variables:

  - Multiplication of all functions on clique tree $= P(\mathbf{X}, E = e)$.

- This corresponds to Step 1 of VE.
Step 3: Choose pivot for inference

- Suppose there is only one query variable $Q$.
- Find a clique $C_Q$ 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 $C$ passes a message to the neighbor $C'$ in the direction of the pivot after receiving messages from all other neighbors $C_1, \ldots, C_k$.
- Suppose $f_i$ is the message from $C_i$ to $C$ and $g_j$ are the functions attached to $C$.
- The message from $C$ to $C'$ is the following function:

$$h(C \cap C' - E) = \sum_{C \setminus (C' \cup E)} \prod_i f_i \prod_j g_j.$$
Step 5: Answer Extraction

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

$$P(Q|E = e) = \frac{\sum_X h(Q, X)}{\sum_{Q,X} h(Q, X)}$$
An Example

Messages passed toward the pivot:

\( f_1(T) = P(A=y)P(T|A=y) \)
\( f_2(R) = P(X=y|R) \)
\( f_3(L, B) = \sum_S P(S)P(L|S)P(B|S) \)
\( f_4(R, B) = \sum_D P(D|R, B) \)
\( f_5(L, R) = \sum_T f_1(T)f_2(R)P(R|T, L) \)

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

1 Clique Trees

2 VE on Clique Trees

3 Correctness of VE on Clique Tree

4 The Clique Tree Propagation Algorithm

5 Constructing Clique Trees
   • Correctness

6 Link to Graph Theory
A clique is **un-activated** if 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 = \( P(\mathbf{X}, \mathbf{E} = e) \),
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 $C'$.

  ![Diagram](image)

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

  $$ h = \sum_{C\setminus(C'\cup E)} \prod_{i} f_i \prod_{j} g_j $$

  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 $r(X')$.

Another description of $r(X')$: 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

\[ r(X') \prod_i f_i \prod_j g_j = P(X, E = e) \]

Now consider the set of variables \( C \setminus (C' \cup E) \).

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

\[
r(X').h = r(X') \sum_Z \prod_i f_i \prod_j g_j
\]

\[
= \sum_Z r(X') \prod_i f_i \prod_j g_j \quad \text{(Fact } Z \notin X' \text{ used here)}
\]

\[
= \sum_Z P(X, E = e) = \sum_Z P(Z, X', E = e) = P(X', E = e)
\]

The proposition is proved. Q.E.D
Theorem (5.1)

Let

- \( X \) stands for the set of unobserved variables in \( C_Q \) except \( Q \),
- \( h(Q, X) = \) product of all functions attached or sent to \( C_Q \) at the end of message passing,

Then

\[
h(Q, X) = P(Q, X, E = e)
\]

Consequently

\[
P(Q|E = e) = \frac{\sum_X h(Q, X)}{\sum_{Q,X} h(Q, X)}
\]

Proof: The Theorem follows readily from Proposition 5.1.
Outline

1. Clique Trees
2. VE on Clique Trees
3. Correctness of VE on Clique Tree
4. The Clique Tree Propagation Algorithm
5. Constructing Clique Trees
   - Correctness
6. Link to Graph Theory
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.

  \[ f_6(R, L) = \sum_B f_4(R, B)f_3(L, B)1 \]

  \[ f_7(R, B) = \sum_L f_5(R, L)f_3(L, B)1 \]

  \[ f_8(L, B) = \sum_R f_4(R, B)f_5(R, L)1 \]

  \[ f_9(R) = \sum_{T,L} f_6(R, L)f_1(T)P(R|T, L) \]

  \[ f_{10}(T) = \sum_{L,R} f_6(R, L)f_2(R)P(R|T, L) \]

**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

\[ CTP(T, E, e) \]

**Input:** \( T \) — Clique, initialized, evidence absorbed

**Output** \( P(X|E = e) \) of every non-observed variable \( X \)

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

CollectMessage($C, C'$)  //  $C \leftarrow C'$
1:  for (each neighbor $C''$ of $C'$ except $C$)
2:        CollectMessage($C', C''$)
3:  end for
4:  SendMessage($C', C$)

DistributeMessage($C, C'$)  //  $C \rightarrow C'$
1:  SendMessage($C, C'$)
2:  for (each neighbor $C''$ of $C'$)
3:        DistributeMessage($C', C''$)
4:  end for
Clique Tree Propagation

\textit{SendMessage}(C', C) // C' \rightarrow C

1: Suppose \( C_1', C_2', \cdots, C_k' \) are all the neighbors of \( C' \) except \( C \)
2: For \( i = 1, 2, \cdots, k \), \( g_i \leftarrow \text{RetrieveMessage}(C_i', C', ) \);
3: Let \( f_1, f_2, \cdots, f_l \) be the function stored at \( C' \) during initialization and \( Z = C' \setminus C \cup E \)
4: \( \psi \leftarrow \sum_Z \prod_{i=1}^l f_i \prod_{j=1}^k g_j \)
5: \text{SaveMessage}(C', C, \psi)
Clique Tree Propagation

Example: CTP on the clique tree shown on Slide 22

- 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)
The Clique Tree Propagation Algorithm

Clique Tree Propagation

- CTP: Lines 5-7, for-loop
  - DM(5, 8):
    - SM(5, 8)
      Combine \( M(2\rightarrow5) \), \( M(4\rightarrow5) \), \( M(6\rightarrow5) \), with functions at 5
      Compute and save \( M(5\rightarrow8) \) 
    - For-loop
      - DM(8, 7): Save and compute \( M(8\rightarrow7) \)
      - DM(8, 9): compute and save \( M(9\rightarrow7) \)
  - DM(5, 2): Compute and save \( M(5\rightarrow2) \), \( M(2\rightarrow1) \), \( M(2\rightarrow3) \)
  - DM(5, 4), DM(5, 6): Compute and save \( M(4\rightarrow4) \), \( M(5\rightarrow6) \)

- 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.

- 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

1. Clique Trees
2. VE on Clique Trees
3. Correctness of VE on Clique Tree
4. The Clique Tree Propagation Algorithm
5. Constructing Clique Trees
   - Correctness
6. Link to Graph Theory
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.
Constructing Clique Trees

An Algorithm

Let $G$ be the moral graph of a BN and $\rho$ be an elimination ordering.

Procedure buildCliqueTree($G, \rho$)

1. Remove the first node $Z$ from $\rho$. Set $S = \text{adj}(Z)$
2. Create clique $C = \{Z\} \cup S$
3. If $C$ contains all nodes in $G$, return the clique tree that consists of only one clique $C$.

4. Else
   1. Add edges to $G$ so that nodes in $S$ are pairwise connected.
   2. Remove $Z$ from $G$.
   3. Recursive call: $T = \text{buildCliqueTree}(G, \rho)$

5. In $T$, find clique $C'$ s.t. $S \subseteq C'$ (*we will show that such clique must exist*).
6. Add $C$ to $T$ by connecting it to $C'$.
7. Return $T$. 
Constructing Clique Trees

An Example

- BN and moral graph:
An Example (cont’d)

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

Eliminate: A
Clique: \{A, T\}

Eliminate: X
Clique: \{R, X\}

Eliminate: D
Clique: \{R, B, D\}

Eliminate: S
Clique: \{L, S, B\}

Eliminate: B
Clique: \{R, L, B\}

Eliminate: L
Clique: \{T, L, R\}

Could connect to RLB or RBD
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 $G$ be the moral graph of a BN and $T$ be the tree constructed by buildCliqueTree for $G$. Then for any node $X$ of the BN, there exists a $t$-clique $C$ in $T$ such that

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

Proof:

- $\{X\} \cup pa(X)$ is a g-clique in the moral graph $G$.
- 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 $Z$,

- $S$ is a $g$-clique in $G$.

- Let $T$ 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 $C'$ in $T$ s.t. $S \subseteq C'$. Q.E.D.
Variable-Connectedness

Proposition (5.3)

*The tree $T$ constructed by buildCliqueTree from undirected graph $G$ is variable-connected.*

**Proof:**

- Induction on the number $n$ of nodes in $G$.
- 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 $C_1$ and $C_2$ in $T$ and suppose $X \in C_1 \cap C_2$.
- Need to show: $X$ appears in all cliques on the path between $C_1$ and $C_2$. 
Variable-Connectedness

- Let $Z$ be the first variable eliminated.
- Let $T'$ be the tree return by the first recursive call.
- According to the induction hypothesis, $T'$ is variable-connected.
- $T$ is $T'$ plus the clique created when eliminating $Z$.
- If neither $C_1$ nor $C_2$ is the clique created when eliminating $Z$,
  - Then they are both in $T'$.
  - Since $T'$ is variable-connected, $X$ appears in all cliques between $C_1$ and $C_2$. 
Proof of Proposition 5.3 (cont’d)

- Now assume $C_1$ is the clique created when eliminating $Z$.
- $X$ cannot be $Z$ because $X$ is in $C_2$.
- Then $X$ must be in the set $S$, i.e. $adj(Z)$ in the graph $G$.
- Let $C'$ be the only neighbor of $C_1$ (determined at step 5).
- Then, $S \subseteq C'$.
- Hence $X$ must be in $C'$, which is in $T'$.
- Since $T'$ is variable-connected, $X$ appears in all cliques between $C'$ and $C_2$.
- Hence $X$ appears in all cliques between $C_1$ and $C_2$. Q.E.D.
Correctness of buildCliqueTree

Theorem (5.2)

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

Proof:

- According to Proposition 5.3, $G$ is a clique tree.
- According to Corollary 5.2, $G$ 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 minimal.
  - Example: Elimination ordering: E, D, C, B, A

As shown, can be easily made minimal.
Outline

1. Clique Trees
2. VE on Clique Trees
3. Correctness of VE on Clique Tree
4. The Clique Tree Propagation Algorithm
5. Constructing Clique Trees
6. Link to Graph Theory
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.
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**: Convert a graph that is not triangulated into one that is by adding edges.

**Example:**

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:

  ![Graph G1](image1)
  ![Graph G2](image2)

  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 \( \rho \)).
- Find all maximal g-cliques in triangulated graph.
- Arrange them into a tree.

The result is the same as that given by \texttt{buildCliqueTree} after minimization.