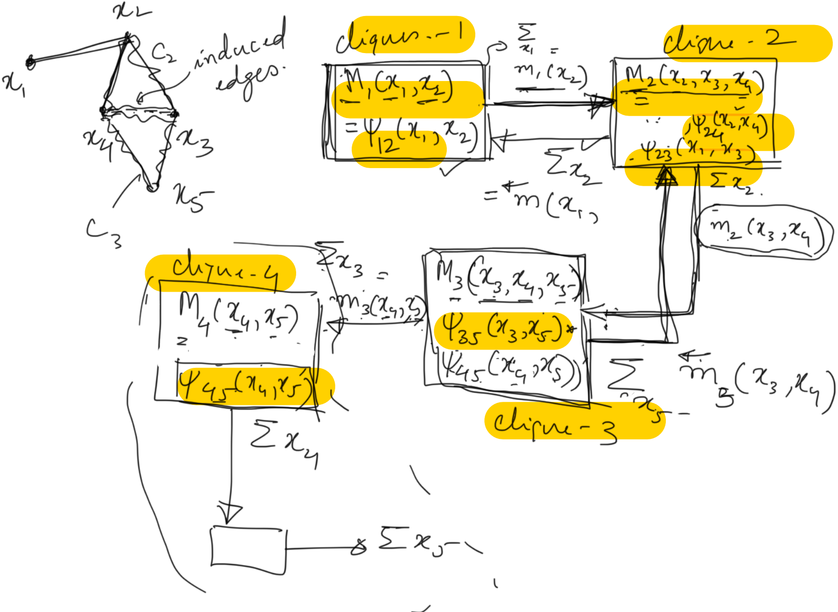
Computation reuse graph



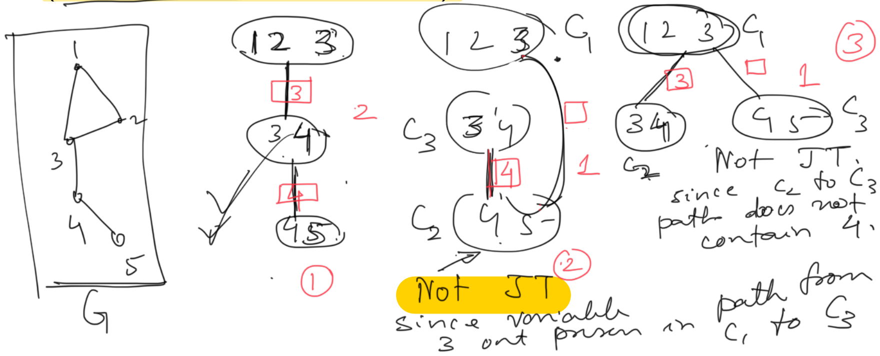
Junction tree algorithm

- An optimal general-purpose algorithm for exact marginal/MAP queries
- Simultaneous computation of many queries
- Efficient data structures
- Complexity: $O(m^w N)$ $w=$ size of the largest clique in (triangulated) graph, $m =$ number of values of each discrete variable in the clique. $\rightarrow$ linear for trees.
- Basis for many approximate algorithms.
- Many popular inference algorithms special cases of junction trees
  - Viterbi algorithm of HMMs
  - Forward-backward algorithm of Kalman filters

Junction tree

Junction tree JT of a triangulated graph G with nodes $x_1, \ldots, x_n$ is a tree where

- Nodes = maximal cliques of G
- Edges ensure that if any two nodes contain a variable x_i then x_i is present in every node in the unique path between them (Running intersection property).



Constructing a junction tree

Efficient polynomial time algorithms exist for creating a JT from a triangulated graph.

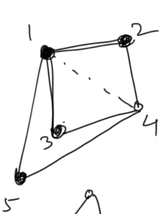
1. Enumerate a covering set of cliques
2. Connect cliques to get a tree that satisfies the running intersection property.

If graph is non-triangulated, triangulate first using heuristics, optimal triangulation is NP-hard.

Optimal triangulation: A triangulation which gives rise to a JT where the size of the largest clique is smallest.

Triangulation heuristics

- Choose vertex with smallest degree and connect all its neighbors.
- Choose vertex which will require smallest additional edges to connect neighbors.

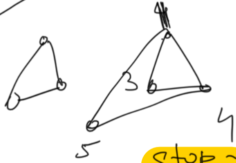


for $i = 1$ to n

π_i = choose the vertex i for which some score is minimum

connect all neighbors of chosen vertex π_i.

remove π_i from G.

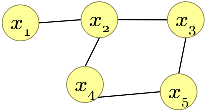


stop.

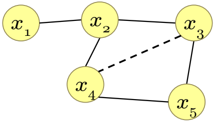
$(1, 3, 4, 5) \rightarrow$ not minimal. gets since $(1, 4)$ is an edge.

Creating a junction tree from a graphical model

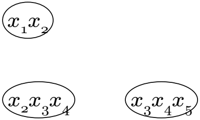
1. Starting graph



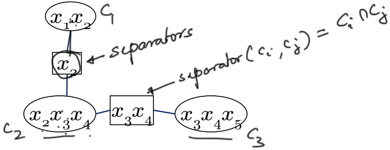
2. Triangulate graph



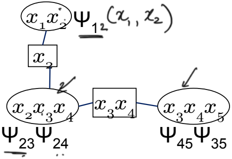
3. Create clique nodes



4. Create tree edges such that variables connected.



5) Assign potentials to exactly one subsumed clique node.



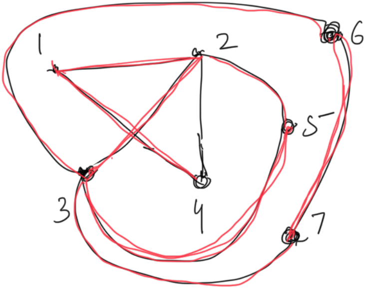
Finding cliques of a triangulated graph

Theorem

Every triangulated graph has a simplicial vertex, that is, a vertex whose neighbors form a complete set.

Input: Graph G. n = number of vertices of G
for $i = 1, \ldots, n$ **do**
 π_i = pick any simplicial vertex in G
 $C_i = \{\pi_i\} \cup Ne(\pi_i)$
 remove π_i from G
end for
Return the maximal cliques from $C_1, \ldots C_n$ as nodes of JT

Example

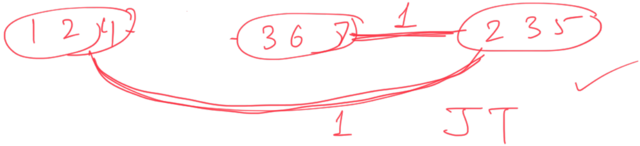


$\Pi_1 = 1$ $\quad \Pi_2 = 6$ $\quad \Pi_3 = 5$

$C_1 = \{1,2,4\}$ $\quad C_2 = \{3,6,7\}$ $\quad C_3 = \{2,3,5\}$

$\Pi_4 = 7$ $\quad \Pi_5 = 3$ $\quad \Pi_6 = 4$

$C_4 = \{3,7\}$ $\quad \{2,3\}$ $\quad \{2,4\}$



Connecting cliques to form junction tree

Separator variables = intersection of variables in the two cliques joined by an edge.

Theorem

A clique tree that satisfies the running intersection property maximizes the number of separator variables.

Proof: `https://people.eecs.berkeley.edu/~jordan/courses/281A-fall04/lectures/lec-11-16.pdf`

Input: Cliques: $C_1, \ldots C_k$
Form a complete weighted graph H with cliques as nodes and edge weights = size of the intersection of the two cliques it connects.
T = maximum weight spanning tree of H
Return T as the junction tree.