Cluster trees and message propagation

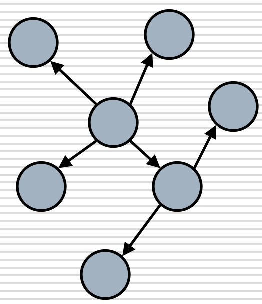
CS3710 Advanced AI

Tomas Singliar

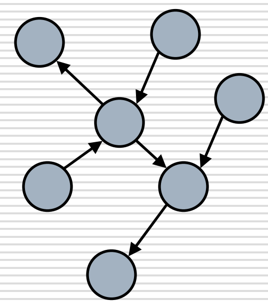
Outline

- Simple graphs: trees and polytrees
- Cluster graphs and clique trees
  - running intersection, sepsetsMessage propagation (= VE)
- Message passing VE in detail
- Caching, out-of-clique queries, DP
- Incremental updating
- Constructing clique trees
  - variable elimination
- VE and BP: Pros & cons and tradeoffs

Trees and polytrees



Tree: one directed path
from the root to each node



PolyTree: one undirected trail
from the root to each node

Undirected representation: a tree graph, treewidth=1

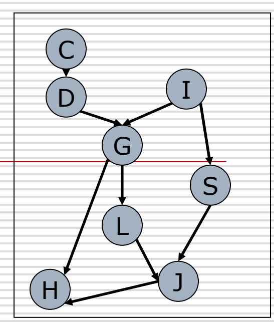
Clique trees

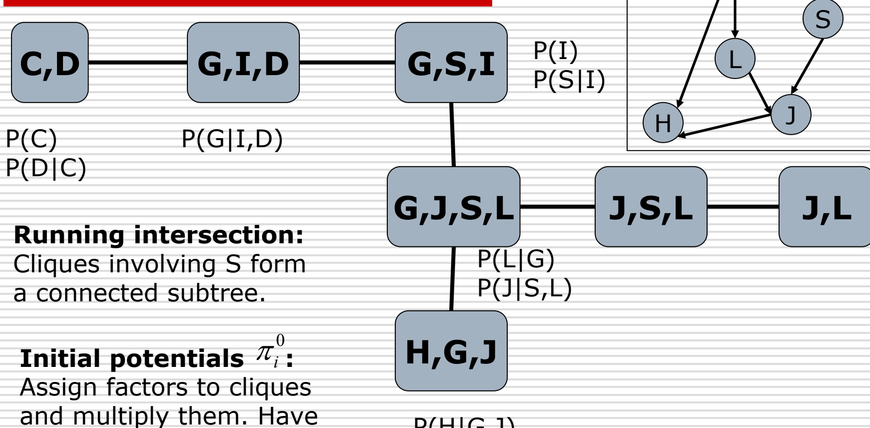
- VE works on *factors*
- Make factor a data structure
  - Sends and receives messages
- Cluster graph for set of factors □, each node *i* is associated with a subset (**cluster**) C_i of □.
  - Family-preserving: each factor’s variables are completely embedded in a cluster

Clique tree properties

- Sepset $S_{ij} = C_i \cap C_j$
  - **separation** set: Variables **X** on one side of sepset are separated from the variables **Y** on the other side in the factor graph given variables in **S**
- Running intersection
  - if C_i and C_j both contain X, then all cliques on the unique path between them do

Clique trees



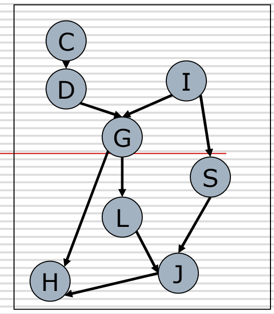


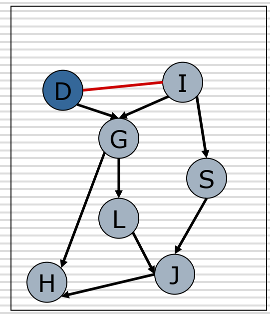
Running intersection: Cliques involving S form a connected subtree.

Initial potentials π_i^0: Assign factors to cliques and multiply them. Have respect for families!

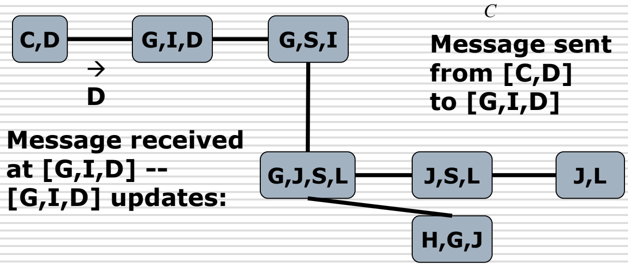
(Tree need not be minimal)

Message Passing VE



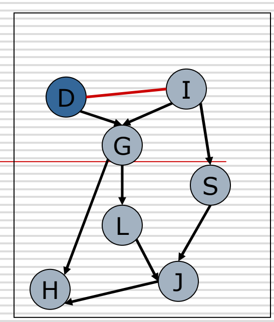


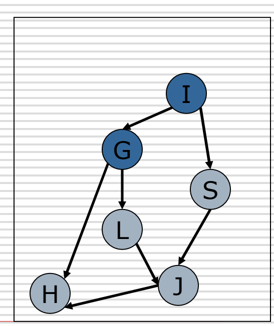
- Query for P(J)
  - Eliminate C: $\tau_1(D) = \sum_C \pi_1^0[C,D]$



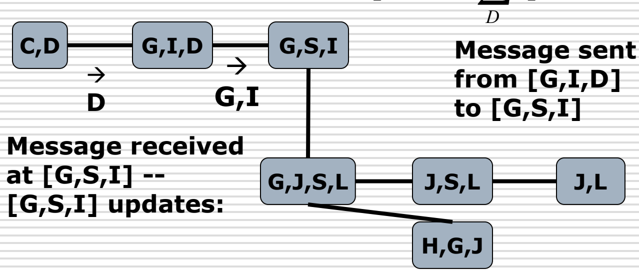
$$\pi_2[G,I,D] = \tau_1(D) \times \pi_2^0[G,I,D]$$

Message Passing VE





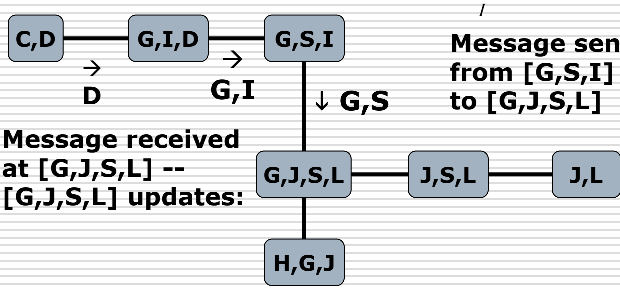
- Query for P(J)
  - Eliminate D: $\tau_2(G,I) = \sum_D \pi_2[G,I,D]$



$$\pi_3[G,S,I] = \tau_2(G,I) \times \pi_3^0[G,S,I]$$

Message Passing VE

- Query for P(J)
  - Eliminate I: $\tau_3(G,S) = \sum_I \pi_3[G,S,I]$

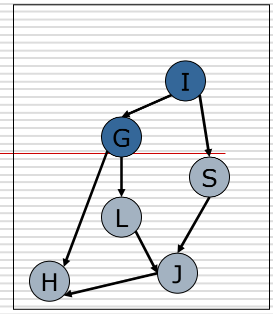


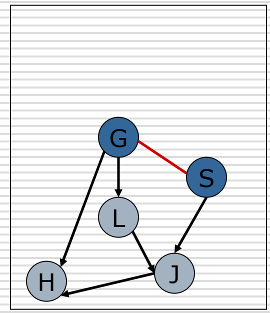
Message sent from [G,S,I] to [G,J,S,L]

Message received at [G,J,S,L] -- [G,J,S,L] updates:

$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \pi_4^0[G,J,S,L]$$

!

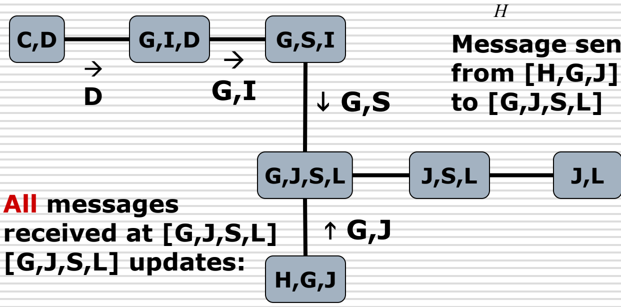




[G,J,S,L] is not **ready**!

Message Passing VE

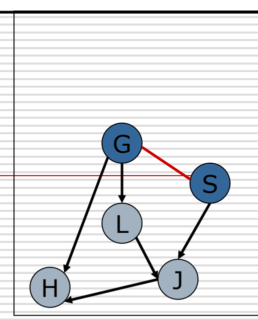
- Query for P(J)
  - Eliminate H: $\tau_4(G,J) = \sum_H \pi_5[H,G,J]$

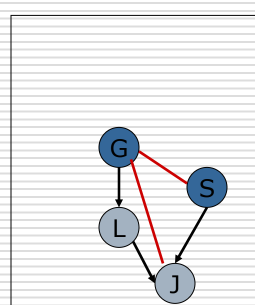


Message sent from [H,G,J] to [G,J,S,L]

All messages received at [G,J,S,L] [G,J,S,L] updates:

$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \pi_4^0[G,J,S,L]$$

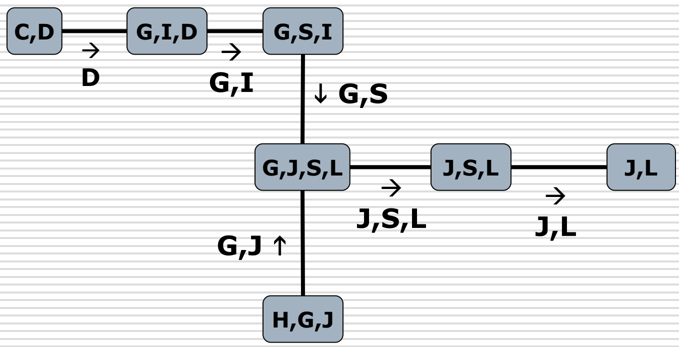




And so on…

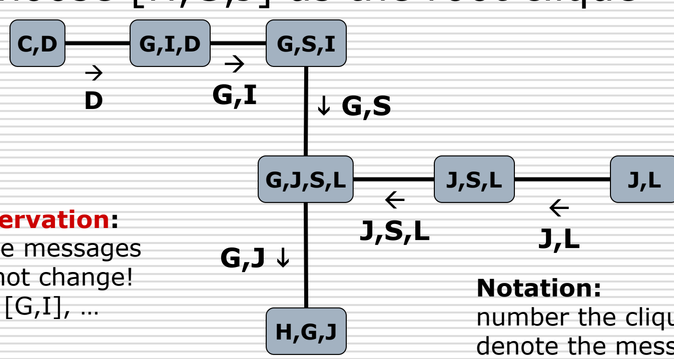
Message Passing VE

- Chose [J,L] as the root clique
- Could we have chosen otherwise?



Message Passing VE

- Choose [H,G,J] as the root clique



Observation:
Some messages did not change!
[D], [G,I], …

Notation:
number the cliques and denote the messages

$$\delta_{i \to j}$$

Correctness of VE on clique trees

- Message summarizes information in the part of tree it separates

$$\delta_{i\to j}(S_{ij}) = \sum_{V_{\prec(i\to j)}} \prod_{\phi\in F_{\prec(i\to j)}} \phi$$

- Proof is by induction from leaves
- Base case – leaf clique C_i

$$\delta_{i\to j}(C_i \cap C_j) = \sum_{C_i - S_{ij}} \pi_i^0(C_i) = \sum_{C_i - S_{ij}} \prod_{\phi\in F_i} \phi$$

Correctness of VE on clique trees

- Induction case: non-leaf clique C_i , sending to C_j, with children $C_{i_1},\ldots,C_{i_k}$

$$\sum_{V_{\prec,(i\to j)}} \prod_{\phi\in F_{\prec,(i\to j)}} \phi = \sum_{Y_i} \sum_{V_{\prec,(i_1\to i)}} \cdots \sum_{V_{\prec,(i_k\to i)}} \left(\prod_{\phi\in F_i} \phi\right) \times \left(\prod_{\phi\in F_{\prec,(i_1\to j)}} \phi\right) \cdots \times \left(\prod_{\phi\in F_{\prec,(i_k\to j)}} \phi\right)$$

- by intersection property, the unions are disjoint

$$V_{\prec,(i\to j)} = Y_i \overset{+}{\bigcup_{m=1,..k}} V_{\prec,(i_m\to i)} \qquad F_{\prec,(i\to j)} = F_i \overset{+}{\bigcup_{m=1,..k}} F_{\prec,(i_m\to i)}$$

- Then

$$\sum_{V_{\prec,(i\to j)}} \prod_{\phi\in F_{\prec,(i\to j)}} \phi = \sum_{Y_i} \left(\prod_{\phi\in F_i} \phi\right) \times \sum_{V_{\prec,(i_1\to i)}} \left(\prod_{\phi\in F_{\prec,(i_1\to j)}} \phi\right) \times \ldots \sum_{V_{\prec,(i_k\to i)}} \left(\prod_{\phi\in F_{\prec,(i_k\to j)}} \phi\right)$$

Correctness of VE on clique trees

By induction hypothesis $\delta_{i\to j}(S_{ij}) = \sum_{V\prec(i\to j)} \prod_{\phi\in F_{\prec(i\to j)}} \phi$

$$\sum_{V_{\prec,(i\to j)}} \prod_{\phi\in F_{\prec,(i\to j)}} \phi = \sum_{Y_i} \pi_i^0 \times \prod_{m=1,..,k} \delta_{i_m\to i} = \delta_{i\to j}$$

Then the root clique has the correct marginal:

$$\pi_r(C_r) = \sum_{X-C_r} \pi_r^0 \prod_{i\in Children(r)} \delta_{i\to r} = \sum_{X-C_r} \pi_r^0 \prod_{i\in Children(r)} \sum_{V\prec(i\to r)} \prod_{\phi\in F_{\prec(i\to r)}} \phi =$$

$$\sum_{X-C_r} \pi_r^0 \prod_{i\in Children(r)} \left[\pi_i^0(C_i) \prod_{j\in Children(i)} \delta_{j\to i} \right] = \ldots = \sum_{X-C_r} \pi_r^0 \prod_{k\in V\setminus C_r} \pi_k^0(C_k)$$

Message passing VE

- Message order is only partial
- Computes marginals for any node Y
  - Results in a *calibrated* clique tree
- Often, many marginals desired
  - Inefficient to re-run inference
  - One distinct message per edge & direction

- Recap: three kinds of factor objects
  - initial, final potentials and messages

Message Passing VE

- Shafer-Shenoy algorithm
  - asynchronous implementation of two passes: upward and downward
  - Asynchronously do:
    - node *i* ready to send *m* to node *j* when it has received a message from all other nodes
    - Send message $\delta_{i \to j} = \sum_{C_i - S_{ij}} \pi_i \times \prod_{k \in N(i) - j} \delta_{k \to i}$
  - Marginalize root clique's ancillary vars

Message Passing: BP

- Graphical model of a **distribution**
  - More edges = larger expressive power
  - Clique tree also a model of distribution
  - Message passing preserves model but changes parameterization
- Different but equivalent algorithm

Factor division

A=1	B=1	0.5
A=1	B=2	0.4
A=2	B=1	0.8
A=2	B=2	0.2
A=3	B=1	0.6
A=3	B=2	0.5

A=1	0.4
A=2	0.4
A=3	0.5

A=1	B=1	0.5/0.4=1.25
A=1	B=2	0.4/0.4=1.0
A=2	B=1	0.8/0.4=2.0
A=2	B=2	0.2/0.4=2.0
A=3	B=1	0.6/0.5=1.2
A=3	B=2	0.5/0.5=1.0

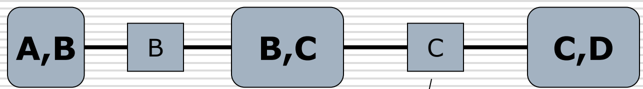
Inverse of factor product

Message Passing: BP

- Each node: multiply all the messages and divide by the one coming from node we send to
  - Clearly the same as VE

$$\delta_{i\to j} = \frac{\sum_{C_i - S_{ij}} \pi_i}{\delta_{j\to i}} = \frac{\sum_{C_i - S_{ij}} \prod_{k\in N(i)} \delta_{k\to i}}{\delta_{j\to i}} = \sum_{C_i - S_{ij}} \prod_{k\in N(i)\setminus j} \delta_{k\to i}$$

Message Passing: BP



$\pi_1^0(A,B)$ $\pi_2^0(B,C)$ $\pi_3^0(C,D)$

Store the last message on the edge and divide each passing message by the last stored.

$$\delta_{2\to 3} = \sum_B \pi_2^0(B,C)$$

$$\pi_3(C,D) = \pi_3^0(C,D) \sum_B \pi_2^0(B,C)$$

$$\delta_{3\to 2}(C) = \sum_D \pi_3(C,D)$$

$$\pi_2(B,C) = \frac{\pi_2^0(B,C)}{\delta_{2\to 3}(C)} \times \delta_{3\to 2}(C) = \frac{\pi_2^0(B,C)}{\delta_{2\to 3}(C)} \times \sum_D \pi_3^0(C,D) \times \delta_{2\to 3}(C) = \pi_2^0(B,C) \times \sum_D \pi_3^0(C,D)$$

Message Propagation: BP

- Lauritzen-Spiegelhalter algorithm
- Two kinds of objects
  - Initial potentials not kept
- Improved "stability" of asynchronous algorithm (repeated messages cancel out)
- Distribution representation – clique tree invariant

$$\pi_T = \frac{\prod_{C_i \in T} \pi_i(C_i)}{\prod_{(C_i \leftrightarrow C_j) \in T} \mu_{ij}(S_{ij})} = P_F(X)$$

Multiple queries

- Much caching possible over a clique tree
- Example: compute P(X,Y), for each $X, Y \in$ □
- Dynamic programming
  - Base case, X and Y are in neighbor cliques

$$P(C_i \mid C_j) = \frac{\pi_i(C_i)}{\mu_{ij}(C_i \cap C_j)} \qquad P(C_i) \propto \pi_i$$

  - Take advantage of conditional independence:

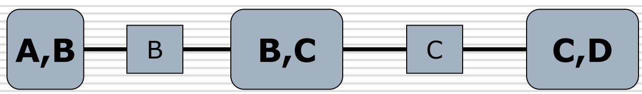
$$\exists l : C_i \perp C_j \mid C_l \qquad P(C_i, C_j) = \sum_{C_l - C_j} P(C_i, C_l) P(C_l \mid C_j)$$

Incremental updates

- Fully-informed: all neighbors have sent their messages
- Calibrated -- messages and cliques agree on marginals: $\sum_{C_i - S_{ij}} \pi_i = \sum_{C_j - S_{ij}} \pi_j = \mu_{ij}$
  - fixed point of MP
- Evidence available in pieces
  - Re-running inference inefficient
- Express evidence in indicator vector
  - multiply into some clique C_i
  - run one pass away from C_i to inform the rest
  - works for soft evidence as well

Out-of-clique queries

- I want P(B, D), no clique with both B and D!
  - Build a new clique tree – expensive, or
  - Do variable elimination over *calibrated* tree



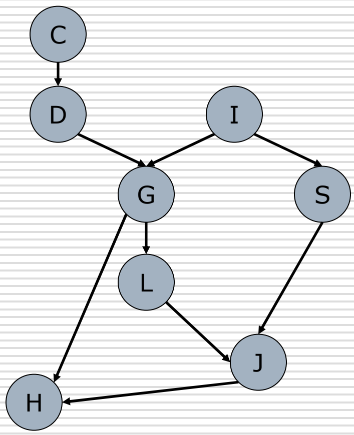
$$P(B,D)=\sum_C P(B,C,D)$$
$$=\sum_C \frac{\pi_2(B,C)\pi_3(C,D)}{\mu_{23}(C)}$$
$$=\sum_C P(B\mid C)P(C,D)$$

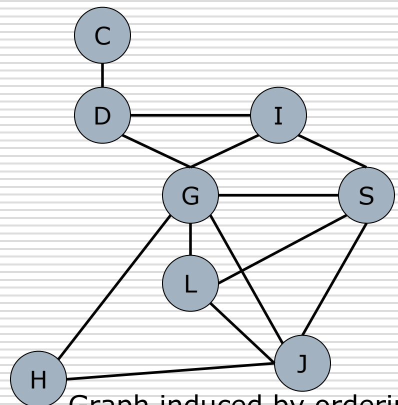
This is back to VE, we save if variables of interest are close in the clique tree.

Defining clique trees

- VE defines cliques
  - Each factor is subset of a clique of □ □
  - Every max clique in □ □ is a factor
  - Each clique in □ is a subclique in □ □
  - Each clique in □ □ is a clique in □
  - Non-maximal cliques can be eliminated
- Chordal graphs
  - Maximal cliques of any c.g. that is a superset of □ □ can be arranged into a clique tree for □
  - triangulation

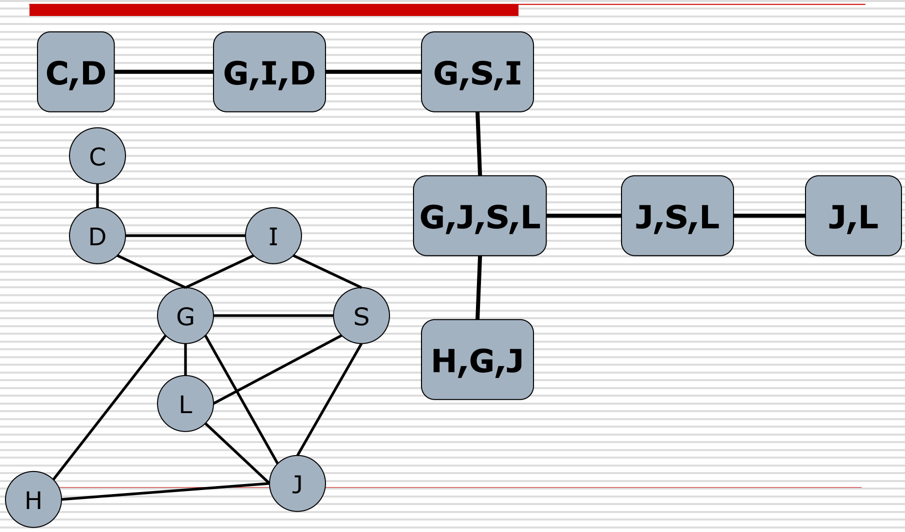
Clique trees generated by VE



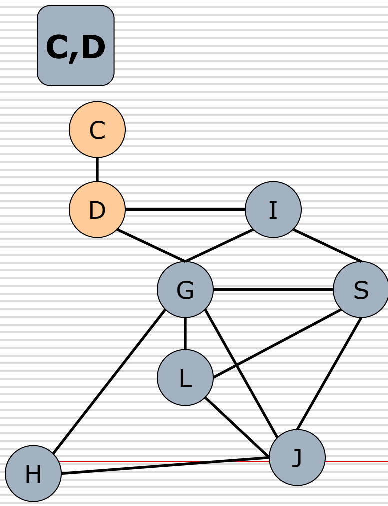


Graph induced by ordering:
C,D,I,H,G,S,L

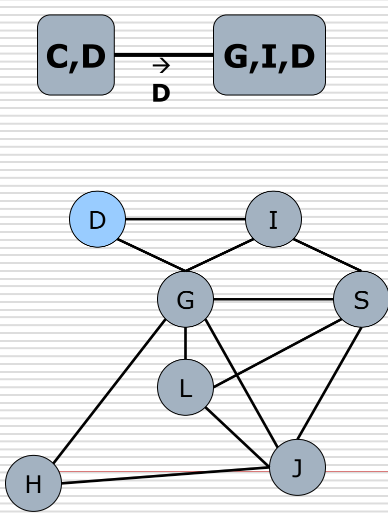
VE constructing a clique tree



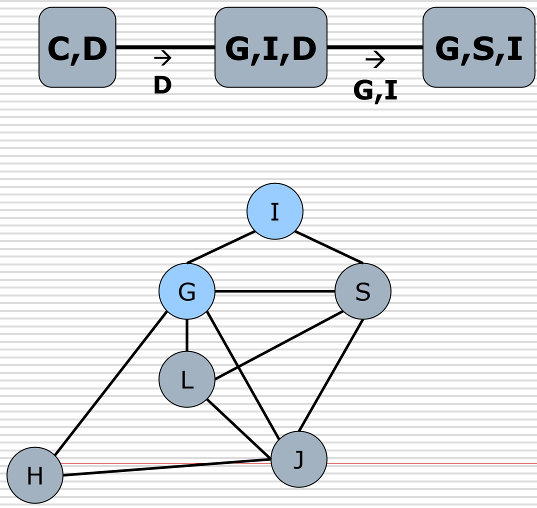
VE constructing a clique tree



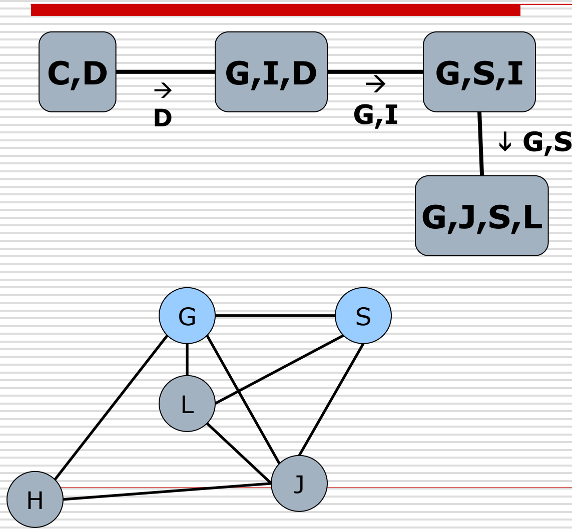
VE constructing a clique tree



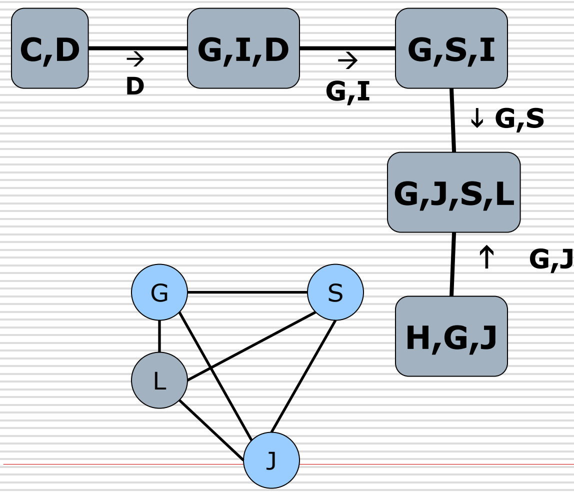
VE constructing a clique tree



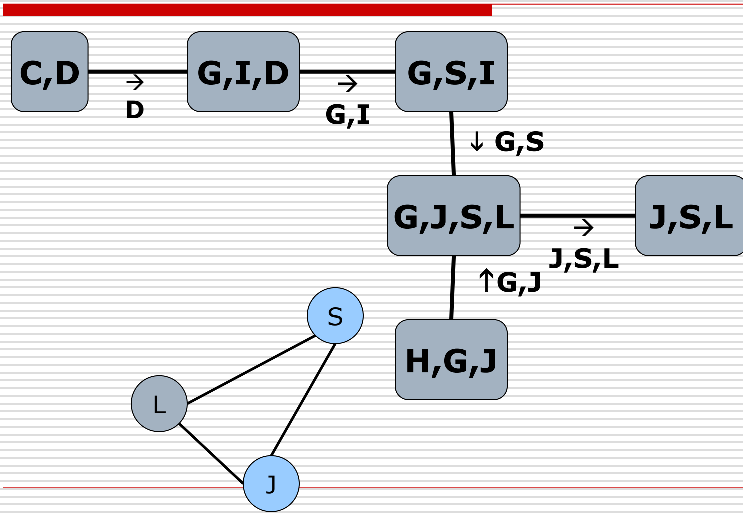
VE constructing a clique tree



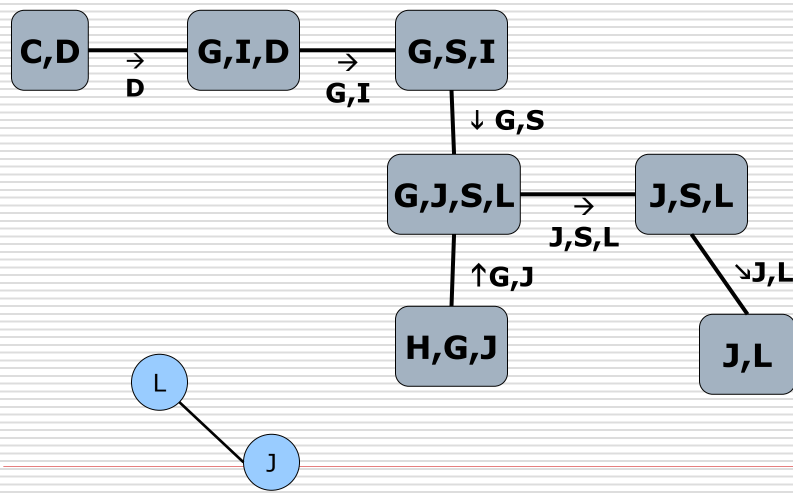
VE constructing a clique tree



VE constructing a clique tree



VE constructing a clique tree



Summary

- Clique trees
  - factors assigned to cliques many-to-one
  - running intersection
- Message passing on clique trees
  - Variable Elimination
  - Belief propagation
  - different views, algebraically the same
- VE defines cliques
- Time and space tradeoff spectrum

Thank you

- Questions solicited