

# Reasoning with Bayesian Networks

## Lecture 4: Models for Graph Decomposition, Most Likely Instantiations

Jinbo Huang

NICTA and ANU

## Models for Graph Decomposition

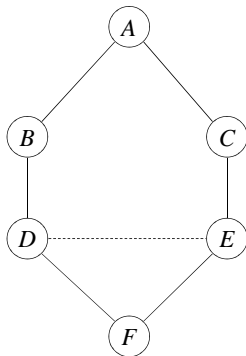
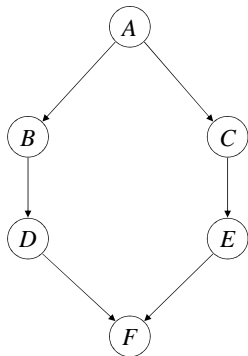
- ▶ Elimination orders for variable elimination
- ▶ Elimination trees / jointrees for factor elimination
- ▶ Dtrees for recursive conditioning
- ▶ All have notion of *width*, corresponding to treewidth

## Models for Graph Decomposition

- ▶ Elimination orders for variable elimination
- ▶ Elimination trees / jointrees for factor elimination
- ▶ Dtrees for recursive conditioning
- ▶ All have notion of *width*, corresponding to treewidth
- ▶ Width-preserving transformations
- ▶ Connection to graph triangulation

# Moral Graphs

Make each family of DAG a clique, drop directionality



# Elimination Orders

Generating low-width elimination orders by heuristic and optimal search

Width-preserving conversion of jointrees and dtrees into elimination orders

## Elimination Orders

Elimination with respect to graph (as opposed to Bayesian network)

- ▶ Eliminating node: connect neighbors into clique, remove node
- ▶ Elimination of all nodes induces sequence of *clusters*: cluster is node together with neighbors before it's removed
- ▶ *Width*:  $\max |cluster| - 1$
- ▶ *Treewidth* of graph: width of best elimination order (NP-hard to find)

## Heuristics for Good Elimination Orders

Min-degree: Eliminate node with fewest neighbors

- ▶ Optimal if treewidth  $\leq 2$

Min-fill: Eliminate node leading to fewest fill-in edges

Min-fill better in practice

- ▶ Can combine the two by using one to break ties
- ▶ Can use randomness: ties broken randomly, random choice of heuristic

## Optimal Elimination Prefixes

- ▶ Sequence of nodes that can be extended to optimal elimination order (i.e., “safe” nodes to eliminate)
- ▶ Width: Consider clusters generated by prefix
- ▶ How to identify “safe nodes” to eliminate?

## Simplicial Rule

*Simplicial* node: Neighbors form clique

Eliminate any simplicial node with degree  $d$

Update lowerbound on treewidth

$$low \leftarrow \max(low, d)$$

Special cases

- ▶ *Islet Rule*: Eliminate node with degree 0
- ▶ *Twig Rule*: Eliminate node with degree 1

## Almost Simplicial Rule

*Almost simplicial* node: Neighbors form clique except one

Eliminate any almost simplicial node with degree  $d$  as long as  $low \geq d$

Special cases

- ▶ *Series Rule*: Eliminate node with degree 2 if  $low \geq 2$
- ▶ *Triangle Rule*: Eliminate node with degree 3 if  $low \geq 3$  & at least one edge between neighbors

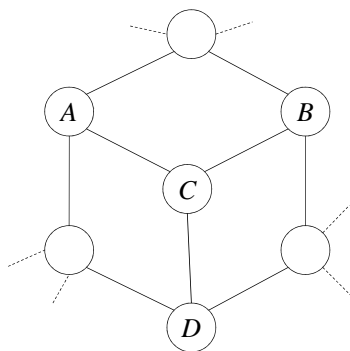
## Buddy Rule

*Buddies*:  $X$  and  $Y$  each having degree 3 and sharing same set of neighbors

If  $low \geq 3$ , eliminate any pair of buddies

## Cube Rule

If  $low \geq 3$ , eliminate any four nodes forming a *cube*



## Benefits of Preprocessing Rules

- ▶ Complete if treewidth  $\leq 3$  (repeated applications will eliminate all nodes)
- ▶ Can simplify graph considerably, improving efficiency of both optimal and heuristic algorithm for good elimination orders

## Lower Bounds on Treewidth

Can empower preprocessing rules

Critical for optimal search algorithms

- ▶ If graph has clique of size  $n$ , treewidth  $\geq n - 1$
- ▶ Degree of graph (min degree of any node)

## Lower Bounds on Treewidth

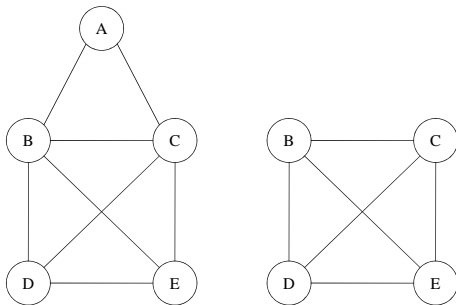
Can empower preprocessing rules

Critical for optimal search algorithms

- ▶ If graph has clique of size  $n$ , treewidth  $\geq n - 1$
- ▶ Degree of graph (min degree of any node)  $\leq$  treewidth

## Lower Bounds on Treewidth: Degeneracy

- ▶ Treewidth of subgraph cannot  $>$  that of original
- ▶ Degree of subgraph can  $>$  that of original

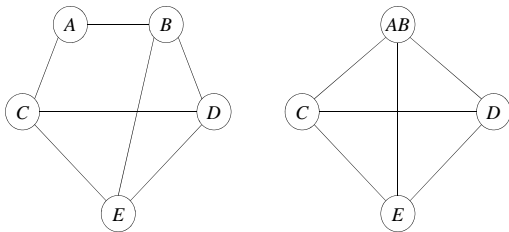


## Lower Bounds on Treewidth: Degeneracy

- ▶ *Degeneracy*: max degree of any subgraph
- ▶ Can be computed by considering only  $n$  subgraphs, created by successively removing min-degree node

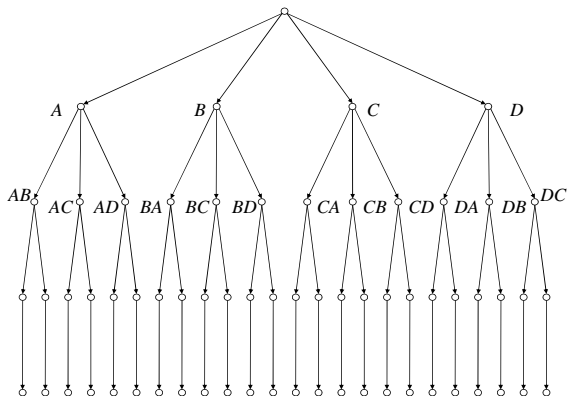
## Lower Bounds on Treewidth: Contraction Degeneracy

- ▶ *Graph minor*: graph obtained by node/edge removal and edge *contraction*
- ▶ *Contraction degeneracy*: max degree of any minor (NP-hard to compute)



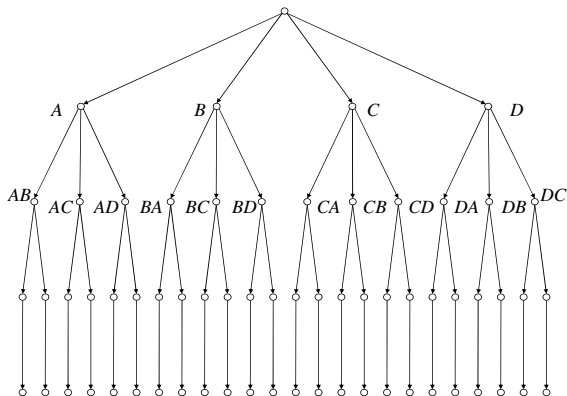
# Optimal Elimination Orders: Depth-first Search

- ▶ Use lower bounds to prune
- ▶ Linear space, but  $O(n!)$  time in worst case



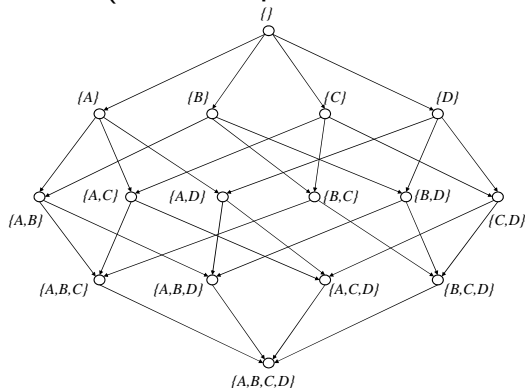
# Optimal Elimination Orders: Depth-first Search

- ▶ Contains many duplicate nodes: Prefixes over same variables result in same subgraph



# Optimal Elimination Orders: Best-fist Search

- ▶ Merge duplicates using open/closed lists
- ▶ Expand node with best estimate:  
 $\max(\text{width of prefix, lower bound for subgraph})$

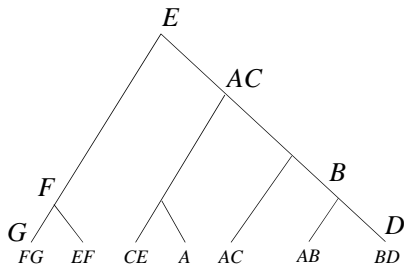
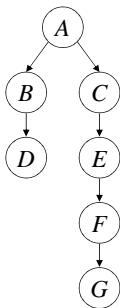


## Jointrees to Elimination Orders

- ▶ Simulate factor elimination
- ▶ Eliminate node, record set of variables “summed out”
- ▶ Sequence of sets of variables gives a class of elimination orders, with width  $\leq$  that of jointree

## Dtrees to Elimination Orders

- ▶ Partial order among cutsets gives a class of elimination orders, with width  $\leq$  that of dtree
  - ▶ Parents after child
  - ▶ Cutset of leaf  $\stackrel{def}{=} \text{variables} \setminus \text{context}$
  - ▶ Fact: All cutsets are disjoint



# Review of Jointrees

Jointree for DAG  $G$  is a tree where each node  $i$  is labeled with *cluster*  $\mathbf{C}_i$  such that:

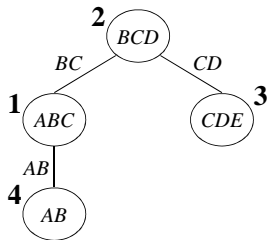
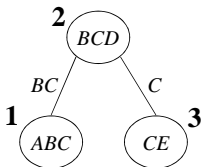
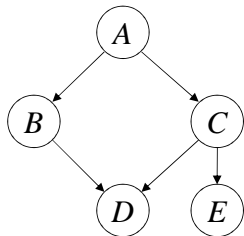
- ▶ Each  $\mathbf{C}_i$  is a set of nodes of  $G$
- ▶ Each family of  $G$  is contained in some  $\mathbf{C}_i$
- ▶  $\mathbf{C}_i \cap \mathbf{C}_j$  is contained in every cluster on the path between nodes  $i$  and  $j$  (*jointree property, running intersection property*)

Minimal jointree: ceases to be jointree with any variable removed from any cluster

*Width* of jointree  $\stackrel{\text{def}}{=} \max |\mathbf{C}_i| - 1$

For edge  $i-j$ , *separator*  $\mathbf{S}_{ij} \stackrel{\text{def}}{=} \mathbf{C}_i \cap \mathbf{C}_j$

# Jointrees



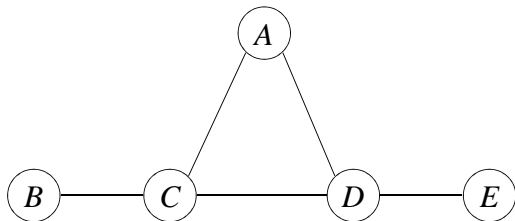
## Transformations That Preserve Jointree

- ▶ Add variable  $X$  to cluster as long as  $X \in$  neighbor
- ▶ Merge two neighboring clusters
- ▶ Add cluster  $C_j$ , connect it to  $C_i$ , as long as  $C_j \subseteq C_i$
- ▶ Remove cluster  $C_j$  as long as it has single neighbor  $C_i$  and  $C_j \subseteq C_i$

## Elimination Orders to Jointrees

Set of clusters induced by elimination ordered can be connected into jointree

$$\mathbf{C}_1 = ACD, \mathbf{C}_2 = BC, \mathbf{C}_3 = CD, \mathbf{C}_4 = DE, \mathbf{C}_5 = E$$



## Elimination Orders to Jointrees

Set of clusters induced by elimination ordered can be connected into jointree

$$\mathbf{C}_1 = ACD, \mathbf{C}_2 = BC, \mathbf{C}_3 = CD, \mathbf{C}_4 = DE, \mathbf{C}_5 = E$$

- ▶ Remove non-maximal cluster ( $\mathbf{C}_3$ ), replace by closest superset ( $\mathbf{C}_1$ )
  - ▶ Superset must be ahead of it (why?)
- ▶ New sequence:  $\mathbf{C}_2, \mathbf{C}_1, \mathbf{C}_4, \mathbf{C}_5$
- ▶ Two ways to connect them into jointree

## Elimination Orders to Jointrees: Method One

Given cluster sequence  $\mathbf{C}_1, \dots, \mathbf{C}_n$ , for  $i = n - 1, \dots, 1$ , connect  $\mathbf{C}_i$  to some  $\mathbf{C}_j, j > i$ , that contains  $\mathbf{C}_i \cap (\mathbf{C}_{i+1} \cup \dots \cup \mathbf{C}_n)$

- ▶  $\mathbf{C}_j$  guaranteed to exist for cluster sequence induced by elimination order
- ▶ Not affected by removal of non-maximal clusters as described

Can be implemented in  $O(n^2)$  time and  $O(n)$  space

## Elimination Orders to Jointrees: Method Two

Connect clusters into complete graph

Define edge cost to be  $|\mathbf{C}_i \cap \mathbf{C}_j|$

Maximum spanning tree is jointree

- ▶ Kruskal's or Prim's algorithm

Slightly less efficient than Method One

## Dtrees to Jointrees

Clusters of dtree form a jointree

- ▶ Each cluster has at most three neighbors
- ▶ Exactly  $2n - 1$  clusters
- ▶ Useful for Shenoy-Shafer to attain  $O(n \exp(w))$  complexity

## Jointrees as Factorization Tool

### Independence information

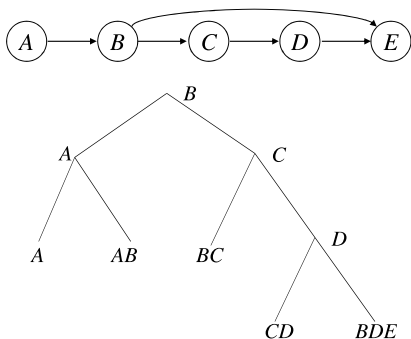
- ▶ Each edge partitions jointree into two
- ▶ Private variables on either side of edge d-separated by separator

### Jointree as factorization

- ▶  $\Pr(\mathbf{X}) = \frac{\prod \Pr(\mathbf{C}_i)}{\prod \Pr(\mathbf{S}_{ij})}$
- ▶ Useful in approximate inference

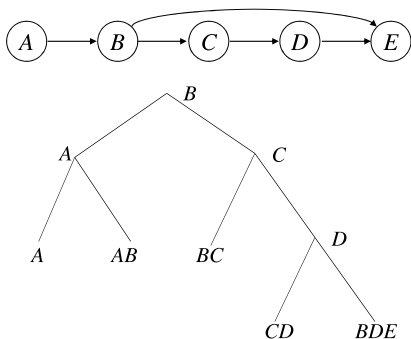
## Review of Dtrees

Full binary tree whose leaves correspond to families of DAG



## Review of Dtrees

*Cutset*:  $\cap$  of variables on either side minus *acutset*  
( $\cup$  of ancestor cutsets)



## Review of Dtrees

*Cutset*:  $\cap$  of variables on either side minus *acutset*  
( $\cup$  of ancestor cutsets)

*Context*: variables  $\cap$  acutset

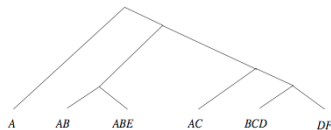
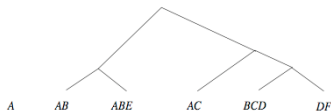
*Cluster*: cutset  $\cup$  context (for leaf, variables)

## Elimination Orders to Dtrees

Start with DAG families as single-node trees, connect them per elimination order

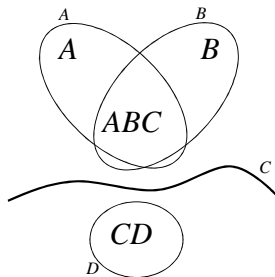
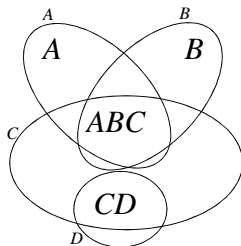
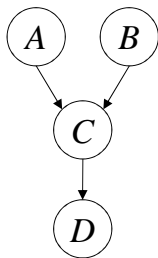
- ▶ When considering  $X$ , connect all trees mentioning  $X$

$$\pi = D, F, E, C, B, A$$



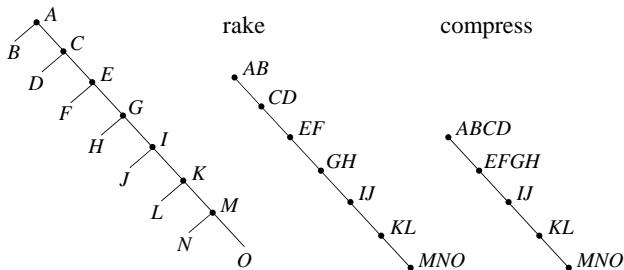
## Dtrees by Hypergraph Partitioning

- ▶ Family  $\rightarrow$  node, variable  $\rightarrow$  hyperedge
- ▶ Recursively partition roughly into halves, minimizing cut (tends to minimize width)



# Balancing Dtrees: Tree Contraction

- ▶ *Rake*: absorb each leaf into parent
- ▶ *Compress*: for maximal chain  $N_1, \dots, N_k$ , absorb  $N_i$  into  $N_{i+1}$  for odd  $i$ 
  - ▶  $N_k$  must have exactly one child, which is not leaf

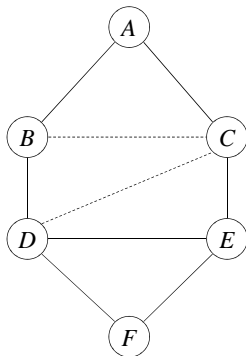
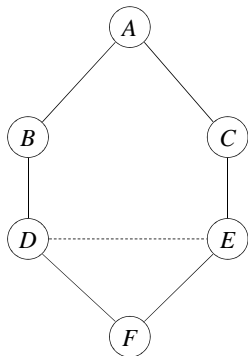


## Balancing Dtrees: Tree Contraction

- ▶ Can reduce tree to single node in  $O(\log n)$  steps
- ▶ Label non-leaf node with  $\emptyset$ , leaf with itself
- ▶ When absorbing, compose the dtrees
- ▶ When tree reduced to single node, new dtree will be balanced
  - ▶ Height  $O(\log n)$ , cutset width  $\leq w$ , context width  $\leq 2w$ , width  $\leq 3w - 1$ , where  $w$  is max context size of original dtree
- ▶ Useful for constructing balanced jointrees  
( $\exists$  root within  $O(\log n)$  distance of all nodes)

# Triangulated Graphs

Every cycle of length  $\leq 4$  has a chord



# Triangulated Graphs and Perfect Elimination Orders

Triangulated iff it admits a *perfect* elimination order

- ▶ One that leads to no fill-in edges
- ▶ Can be found in polytime, has optimal width

## Triangulation and Treewidth

Given triangulation  $G^*$  of  $G$ , can construct in polytime elimination order  $\pi$  with  $width(\pi, G) \leq treewidth(G^*)$

- ▶ Get perfect elimination order  $\pi$  for  $G^*$  in polytime
- ▶  $width(\pi, G)$  cannot be larger as  $G \subseteq G^*$

# Triangulation and Treewidth

Given elimination order  $\pi$  for  $G$ , can construct in polytime triangulation  $G^*$  of  $G$  with  $\text{treewidth}(G^*) \leq \text{width}(\pi, G)$

- ▶ Run elimination using  $\pi$ , the filled-in graph  $G^*$  admits  $\pi$  as perfect elimination order
- ▶ Hence  $G^*$  must be triangulated and have treewidth  $\text{width}(\pi, G)$

## Jointree Construction by Triangulation

Triangulate moral graph of Bayesian network,  
identify maximal cliques as jointree clusters

Equivalent to jointree from elimination order

- ▶ Maximal clusters in the cluster sequence induced by elimination order  $\pi$  are precisely maximal cliques of (triangulated) graph  $G^\pi$

## Models for Graph Decomposition: Summary

- ▶ Elimination orders, jointrees, dtrees, triangulations, all related through treewidth
- ▶ Algorithms for low-width decompositions
- ▶ Width-preserving conversions between decomposition models

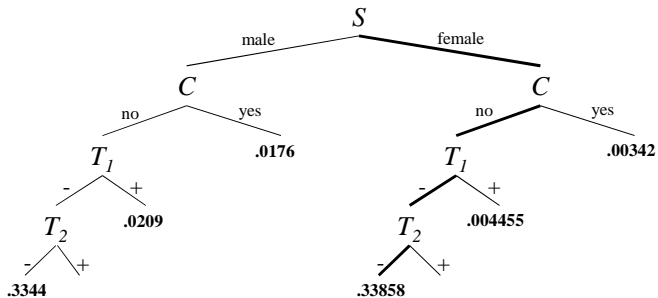
## Most Likely Instantiations

- ▶ MAP (Maximum a Posteriori Hypothesis):  
Maximize  $\Pr(\mathbf{m}|\mathbf{e})$  for variables  $\mathbf{M}$
- ▶ MPE (Most Probable Explanation): Special case  
where  $\mathbf{M}$  are all network variables (minus  $\mathbf{E}$ )

## MPE by Variable Elimination

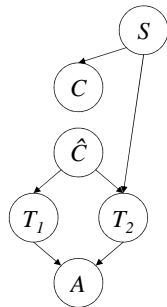
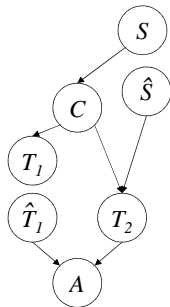
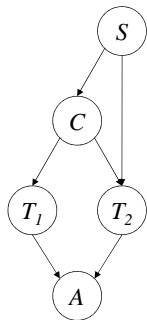
- ▶ VE computes MPE probability with summing out replaced by “maximizing out”
- ▶ MPE instantiation can be recovered by bookkeeping
- ▶ Complexity  $O(n \exp(w))$

# MPE by Systematic Search



# Upper Bounds by Node Splitting

- ▶ Can split according to all or subset of children
- ▶ Reduces treewidth, simplifying problem



## Upper Bounds by Node Splitting

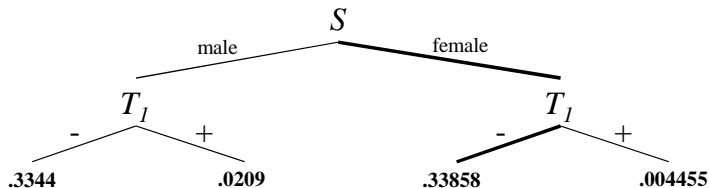
MPE of new problem is upper bound after normalizing by  $\beta$

- ▶ Variable and clone can differ in value, more opportunities for higher probability
- ▶  $\beta$  is # of instantiations of clone variables, as clones get uniform priors

# Upper Bounds by Node Splitting

Need only search over split variables

- ▶ Upper bound becomes exact solution once all split variables are set



## MAP by Variable Elimination

- ▶ Sum out non-MAP variables
- ▶ Remaining problem is MPE, solve with VE by maximizing out
- ▶ Overall, elimination order must put MAP variables **M** last
- ▶ Width of best order under this constraint is **M-constrained treewidth**, can be much higher than treewidth

## MAP by Systematic Search

- ▶ Depth-first search over MAP variables
- ▶ Upper bounds for pruning computed from *unconstrained* orders
  - ▶ Doing max too early (before sum) can only *increase* result
- ▶ Improve bounds by using orders close to constrained orders
  - ▶ Use elimination order obtained from jointree
  - ▶ Delay MAP variables by promoting them toward root, without increasing width

## MAP by Local Search

- ▶ Search in the space of instantiations of MAP variables
- ▶ Start with initial instantiation, try to improve by moving to neighbor
- ▶ *Stochastic hill climbing*: Always move to best neighbor if it leads to improvement; otherwise change value of variable at random

## MAP by Local Search: Initialization

- ▶ Random:  $O(m)$  time
- ▶ MPE based:  $O(n \exp(w))$
- ▶ Maximum individual marginal:  $O(n \exp(w))$
- ▶ Sequentially choose variable  $X_i$  that has highest  $\Pr(x_i | \mathbf{e}, \mathbf{y})$  for one of its values  $x_i$  and set it to  $x_i$  ( $\mathbf{y}$  is instantiations made so far):  $O(m n \exp(w))$ 
  - ▶ Most expensive, tends to be most effective

## Most Likely Instantiations: Summary

- ▶ MPE and MAP by variable elimination and search
- ▶ Upper bounds by node splitting and use of unconstrained elimination orders