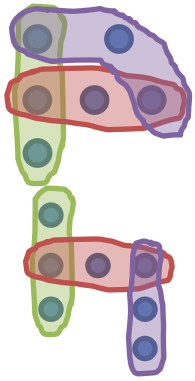
Scribe Notes: 2/4/13
Presenter: Dr. Berthe Choueiry
Scribe: Daniel Geschwender
Reading: Chapter 9 of Dechter's textbook

Tree-Decomposition Methods

The class period was spent discussing questions raised from the reading of Chapter 9 on Tree Decomposition Methods. The full questions can be found on the class piazza as well as responses to many of them.

Tony: Acyclical Recognition

On page 251, the book discusses primal-based recognition of acyclical networks. To quote the book, "...since different hypergraphs can map to the same primal graph, we need to further ensure another property called conformality that the maximal cliques of the chordal primal graph coincide with the scopes of the original constraints". I'm just not sure why this is necessary (and sufficient) for recognizing acyclicity.

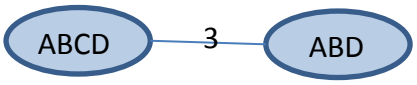


- These relations form a cycle
- They must be joined into one relation so that we can generate solutions in a backtrack-free manner.

- There is no cycle present in these relations
- Directional consistency can be applied here to generate all solutions in a backtrack-free manner

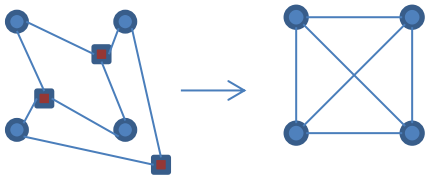
Methods of detecting acyclicity in a network:

1. Using the dual Graph
   - Each edge of the dual graph has a weight equal to the number of variables in common between the relations



   a. Find the maximum spanning tree of the weighted dual graph
   b. Verify that the connectedness property holds
2. Using Primal Graph
   - Find all max cliques and build a join tree with the simplicial ordering
   - A clique can be formed from one or more relations

Robert's Example



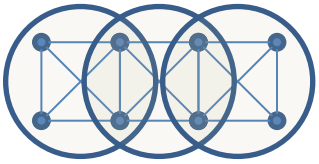
Tony: Ordering for JTC

The book mentions that max cardinality ordering is a good choice for JTC because, if the graph is chordal, no extra edges are need to moralize the ordering. Are there other advantages to using different orderings? Is max-cardinality typically the best, or just in the special case of chordal graphs?

- Max-cardinality ordering
  - Arbitrarily chose first vertex
  - Then choose vertex connected to the most already ordered vertices
  - The order of choosing the vertices is the instantiation ordering
- *If the graph is triangulated*, then the reverse of the instantiation ordering (elimination ordering) is a perfect elimination ordering (i.e., the graph is moralized, using this ordering for elimination will not add any new edges).
- Other orderings could be used as a refinement, to break ties. Least domain may be useful.

Robert: Primal Graph of Hypertree

On page 251, the book says: "... a primal graph of a hypertree must be chordal (Beeri et al. 1983)" and Theorem 9.3 states "(Maier 1983) A hypergraph has a join-tree iff its primal graph is chordal and conformal." Can you give a sketch of why this must be true?



- Vertices of the join tree are formed from cliques of constraints. In order to ensure that we have no cycles, we need to ensure that each clique in the primal graph originate from a constraint (conformality condition), otherwise we have a cycle (see Robert's example above).
- Zion's explanation:
  - The cliques that form the join tree vertices are chordal because they are completely connected
  - The join tree itself is trivially chordal because there are no cycles

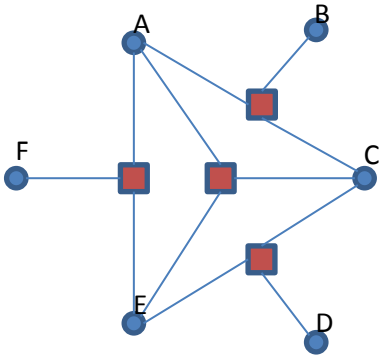
Robert: Use of Primal Graphs

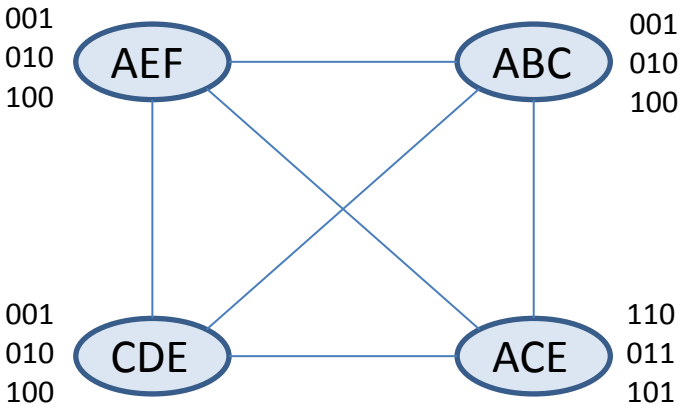
Page 254 describes the Join-Tree-Clustering (JTC) algorithm. This algorithm uses the primal graph, however, the dual graph could have also been used used to generate a (different) tree. The book states that the primal approach is more popular, why isn't the dual approach as popular?

- The dual approach is not widely discussed. Further, Shant found that we get orderings of lower treewidths using the primal graph than using the dual graph.
- Shant's explanation:
  - The best decompositions use triangulation
  - An edge added by triangulating a dual graph can map to many edges (between every two pairs of variables) in the primal graph
  - Resulting in larger cliques and tree decompositions of larger treewidths

Nate: Acyclic Algorithm Example

In example 9.3 on page 249, when the constraint R_{CDE} is considered, They state that R_{ACE} = (R_{ACE} x R_{CDE}) = {(0,1,1)}. However with this solution C=1 and D=1, and there is no support for this in the original constraint R_{CSE} = {(0,0,1),(0,1,0),(1,0,0)}. Is this a typo or an I missing something here?



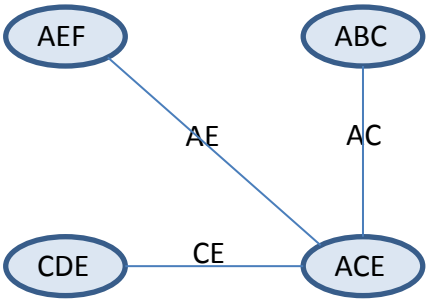


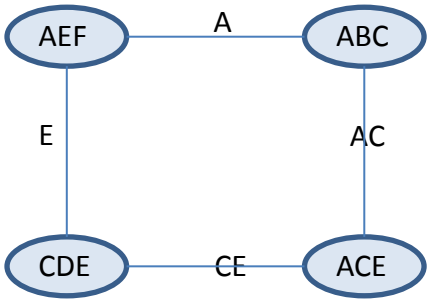
- After having revised R_{ACE} with both R_{AEF} and R_{ABC} only (0,1,1) remains
- Joining this with R_{CDE} yields an empty relation
- Typo in the book and in the slides, please report to wiki

Fikayo: Tree Decomposition

Page 246 talks about the alternate path of constraints for the enforcement of equality. I don't quite understand its relationship to the dual network.

- Edges of the dual graph enforce the values to agree (enforces equality)
- Redundant edges may be removed because equality is still maintained across a path; a cycle is unnecessary
- Minimality of dual graphs
    - Claim: all minimal graphs have the same number of edges
    - Robert demonstrated a graph of weight 6 with 4 edges as compared to a graph of weight 6 with 3 edges



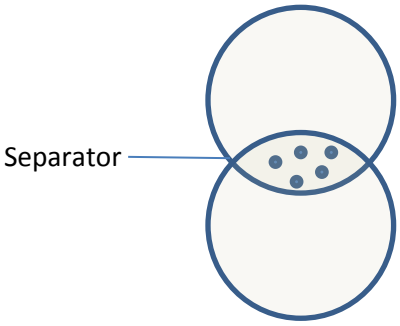


    - Shant: not minimal, minimality based on number of edges and not just weight

Daniel G.:Tree Width vs. Hyperwidth

Theorems 9.8 and 9.9 give two different bounds on the complexity of Cluster-Tree Elimination. One is based on the tree width and one is based on the hyperwidth. Are these two ways of representing the same bound or are different procedures being used to obtain different complexities?

- Complexity by tree width
  - Time: $O((r+N)\cdot deg\cdot k^{w^*})$
  - Space: $O(N\cdot k^{sep})$
- Complexity by hyperwidth
  - Time: $O(N\cdot t^{2hw})$
  - Space: $O(N\cdot t^{hw})$
- deg: maximum degree of tree decomposition
- sep: maximum separator size of tree decomposition
- N: # of clusters in tree decomposition
- k: domain size
- r: number of constraints
- w^*: tree width
- hw: hyperwidth
- The different complexity bounds measure the same thing in different ways and are used in different situations depending on what parameter is fixed (hw, w^*) and what parameter is given as input (e.g., domain size or number of tuples in relation).



Daniel G.: When to use JTC

Both JTC and CTE can be used to create a tree decomposition. When is it preferable to use one or the other? In what situations can JTC not be used?

- The two algorithms do different things and have different inputs and outputs
  - JTC
    - The first three steps of JTC create a join tree which is a tree decomposition
    - After the clusters are formed and the tree created, solutions are found for each cluster
    - The clusters then act as variables and the set of its solutions is its domain
  - CTE
    - Takes a tree decomposition as input, does not build one. Any techniques can be used to create the tree decomposition.
    - Performs message passing twice for each edge, once in each direction
    - Takes all relations in a cluster, joins them, and projects this onto the separator
    - For every node of the tree, the minimal subproblem is returned
- There are many tree decomposition algorithms: Biconnected Component (Freuder), Cycle Cutset Decomposition (Dechter), Tree Clustering (Dechter and Pearl), HyperEdge (Gyssens), Hyper Cutset (Gyssens?), Hypertree (Gottlob), etc.
- Tree clustering is widely used to solve CSPs and more generally graphical models. Hypertree is used by Gottlob's group. It is a more recent technique and operates by search (may be costly).