Scribe Notes: 2/25/13
Presenter: Dr. Berthe Choueiry
Scribe: Daniel Geschwender

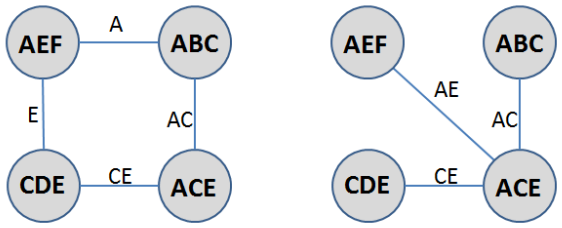
Tree Decomposition Methods – Rina's slides for chapter 9

The class period was spent going over Rina's slides for Chapter 9. The middle portion of the slides was covered starting at "Recognizing acyclic networks" and ending after the "Distributed Arc-Consistency" example.

Recognizing acyclic networks

By removing redundant edges in the dual graph, a tree can be formed. If it satisfies the connectedness property, the problem is acyclic and thus tractable. This tree is a *join tree*. Redundant edges cannot be removed using any technique because cycles may be left. There are two methods of identifying an acyclic network: dual-based and primal-based

1. Dual-based method
    - We replace the labelings of the edges of the dual graph by the number of variables in the labeling.
    - We compute a maximal spanning tree based on the weights of the edges
    - If the connectedness property holds, then the tree is a join tree.

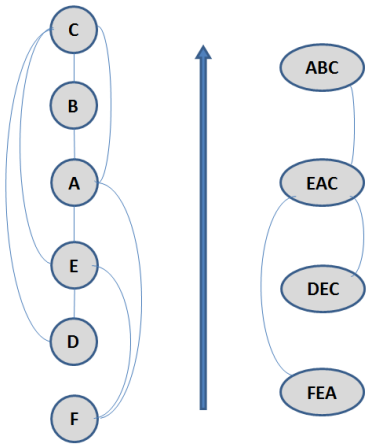


2. Primal-based method
    - According to a theorem in (Maier 83), a network is acyclic if and only if its primal graph is chordal and conformal.
    - We apply MaxCardinality ordering and moralize the graph to make it chordal. If it is not, then the network is not acyclic.
    - If every maximal clique corresponds to the scope of some constraint in the original hypergraph, then the primal graph is conformal.
    - The above guarantees the existence of a join tree, and a join tree can be built as shown below.

Primal-based recognition

- First check for chordality
- Test for conformality
- If a join tree exists, it must now be found

- Connect every clique such that it shares the maximum number of variables with another clique
- See figures for illustration
- Any node can be the root of the tree. By default, choose a balanced tree.



Tree-based clustering

Tree-based clustering builds a tree embedding of an arbitrary constraint network, making look like as it is acyclic even when it is not initially.

- It groups constraints together to form clusters until the problem is acyclic
- We must ensure that every constraint is still represented or the problem will become relaxed
- This technique is the basis for the join-tree clustering algorithm

Join-tree clustering

- The algorithm takes as input a constraint problem and its primal graph
- The algorithm outputs an equivalent acyclic problem and its join tree
- Steps:
    1. Pick an ordering
    2. Moralize
    3. Find maximal cliques, connect cliques in a tree
    4. Place original constraints into the generated cliques
    5. Solve subproblems and generate set of solutions
    6. Return the join tree and the new constraints

Input: A constraint problem R =(X,D,C) and its primal graph G = (X,E).
Output: An equivalent acyclic constraint problem and its join-tree: *T*= (X,D, {C '})
1. Select an d = (x_1,...,x_n)
2. **Triangulation**(create the induced graph along d and call it G^*:)
   for j=n to 1 by -1 do
   E ← E U {(i,k)| (i,j) in E,(k,j) in E }
3. **Create a join-tree of the induced graph G^*:**
   a. Identify all maximal cliques (each variable and its parents is a clique).
   Let C_1,...,C_t be all such cliques,
   b. Create a tree-structure T over the cliques:
   Connect each C_{i} to a C_{j} (j < I) with whom it shares largest subset of variables.
4. Place each input constraint in one clique containing its scope, and let
P_i be the constraint subproblem associated with C_i.
5. Solve P_i and let {R'}_i $ be its set of solutions.
6. Return C' = {R'}_1,..., {R'}_t
the new set of constraints and their join-tree, T.

Unifying Tree Decompositions

- Tree decompositions are studied in graph theory, theoretical computer science, databases as well as in CSPs. They are currently a hot topic.
- The nodes of a tree decomposition are *clusters* of variables and constraints.
- The variables common to two adjacent clusters form a *separator*.
- Any such clustering is a tree decomposition provided it satisfies the following two conditions:
    1. Each constraint in a problem must appear in at least one cluster that also has all the variables in the constraint's scope (otherwise, the problem is relaxed). Additional use of constraints is allowed and can serve to further tighten subproblems

2. The tree has the connectedness property, which requires that every two occurrences of a variable are connected by a path where the variable appears in the cluster.

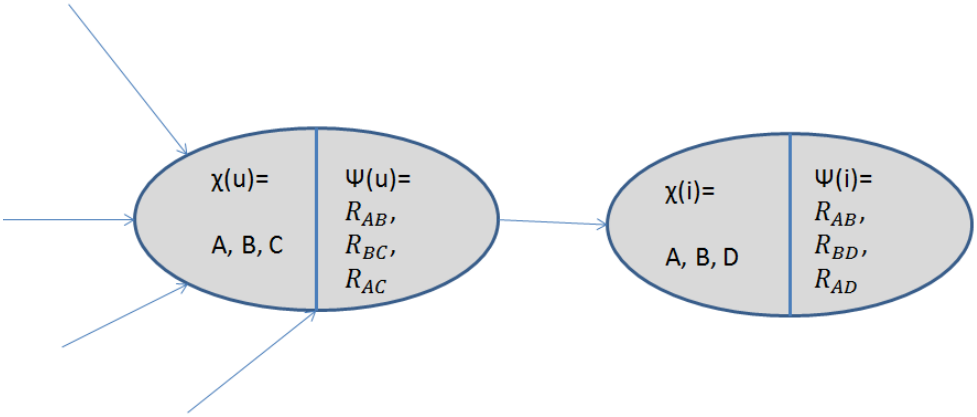
- Many techniques for generating tree decompositions exist: tree-clustering and bucket elimination are just two techniques.
- *Question* (Tony): Will it necessarily be a path connecting the occurrence of a given variable? *Answer*: The graph induced by a given variable must be a connected tree, which means a path between any two occurrences of a given variable

Cluster-Tree Elimination

- Uses message passing between clusters
- Makes two clusters agree on variables that are in their separator
- Each node sends a message to each of its neighbors
- The process is 'amorphous' and does not rely on any specific hierarchical structure or directionality, only that the condition for passing a message is satisfied
- A message may only be passed from cluster A to cluster B if A has received messages from all its neighbors besides B; A may or may not have received a message from B yet.
- *Observation* (Robert): the condition forces the leaves to initiate the message passing

Constraint Propagation

- To produce a message, all relations of a cluster are joined and then projected over the separator



- $m(u,i) = \pi_{AB}(R_{AB} \otimes R_{BC} \otimes R_{AC})$

Distributed relational arc-consistency example

- Uses message passing similar to cluster tree elimination but is not performed on a tree.
- Each cluster is a single relation, each separator is of size one (arc consistency)
- Message passing occurs between clusters simultaneously: each relation receives messages from all its neighbors, computes the join, and sends a message to each of its neighbors.
- Message passing continues until quiescence