Scribe Notes: 2/13/2013
Presenter: Dr. Berthe Y. Choueiry
Scribe: Nate Stender
Topic: Directional Constency (Ch. 4 of Dechter book)

Dechter's Slides[1] for Chapter 4 (Slides 31-37)

Slide 31 - Adaptive-consistency, bucket-elimination

Adaptive-consistency algorithm

- Input: a constraint network *R* and an elimination ordering *d.*
- Output: a backtrack-free network $E_d(R)$ along *d,* if the empty constant was not generated. If empty constraint was generated, problem is inconsistent.

ADAPTIVE-CONSISTENCY (AC)

Input: a constraint network $\mathcal{R}$, an ordering $d = (x_1, \ldots, x_n)$
output: A backtrack-free network, denoted $E_d(\mathcal{R})$, along d, if the empty constraint was not generated. Else, the problem is inconsistent

1. Partition constraints into $bucket_1, \ldots, bucket_n$ as follows:
   for $i \leftarrow n$ **downto** 1, put in $bucket_i$ all unplaced constraints mentioning x_i.
2. **for** $p \leftarrow n$ **downto** 1 **do**
3.   **for** all the constraints $R_{S_1}, \ldots, R_{S_j}$ in $bucket_p$ **do**
4. $A \leftarrow \bigcup_{i=1}^{j} S_i - \{x_p\}$
5. $R_A \leftarrow \Pi_A(\bowtie_{i=1}^{j} R_{S_i})$
6.   **if** R_A is not the empty relation **then** add R_A to the bucket of the latest variable in scope A,
7.   **else** exit and return the empty network
8. **return** $E_d(\mathcal{R}) = (X, D, bucket_1 \cup bucket_2 \cup \cdots \cup bucket_n)$

Figure 1: Dechter's adaptive-consistency algorithm

Comments:

- This algorithm is an example of dynamic programming:
- Problem is broken into sub-problems, and information is passed between the sub-problems
- Usually the goal of dynamic programming is to optimize a function
- We can apply this algorithm to Bayesian networks to find an optimal solution (the most probable explanation, MPE).
- Chapter 13 describes bucket elimination for solving optimization problems. Chapter 14 describes the application of bucket elimination for solving probabilistic networks

Slide 32 - Adaptive-consistency, bucket-elimination (example)

Contrasting DPC, DiC and Adaptive Consistency (ADC)

- In DPC, given an ordering, generate binary constraints between all possible pairs of parents of the node(moralize the graph).
- In DiC, given an ordering, generate all possible constraints of arity (i-1) over the parents of the node.
- In ADC, given an ordering, we take all constraints between a node and its parents, join them, ad project them over the parents, generating a new constraint involving all of the parents of (arity same as number of parents). **Guarantees tractability.**

Tony and Daniel D. participated in a board example of bucket elimination

Complexity:

- Time complexity is determined by the largest arity constraint generated: $O(n(exp)^{w^*(d)})$ => $O(n(k)^{w^*(d)+1})$
- Space complexity is determined by the largest arity constraint stored.
- Both complexities are improved by selecting an ordering with the smallest induced width (which will generate smaller maximum for arity of constraints).

Slide 33 - Properties of bucket-elimination (adaptive consistency)

- Generates a **backtrack-free** constraint network.
- Time complexity: $O(n(2k)^{w^*+1})$ (this is a refinement of time complexity mentioned before)
- Space complexity: $O(n(k)^{w^*+1})$
- Special cases: ***trees*** (w*=1), ***series-parallel networks*** (w*=2), and in general ***k-trees*** (w*=k).

Partial answer to question by Daniel Dobos (https://piazza.com/class#spring2013/csce921/20) about k-trees:

Question: Section 4.1.3 covers *k-trees*, then the book never mentions them again. Is there anything else we should know about these, or was the section just included for completeness?

Answer: k-trees are mentioned because they can be used with ACD to provide a tractable solution.

Slides 34 & 35 (moved over quickly because we have already covered them)

Slide 36 - Summary: directional i-consistency

- Adaptive-consistency - creates constraint between all parents
- Directional path-consistency - creates a binary constraint between all pairs of parents
- Directional arc-consistency - filters domains of all parents

Slide 37 - Variable Elimination

Full example of Bucket-elimination. Note that after one pass through the elimination order, the network can now be solved in a backtrack-free manner in the order. Now a solution can be created by assigning values to the variables in the instantiation ordering, which is the reverse of the elimination ordering.

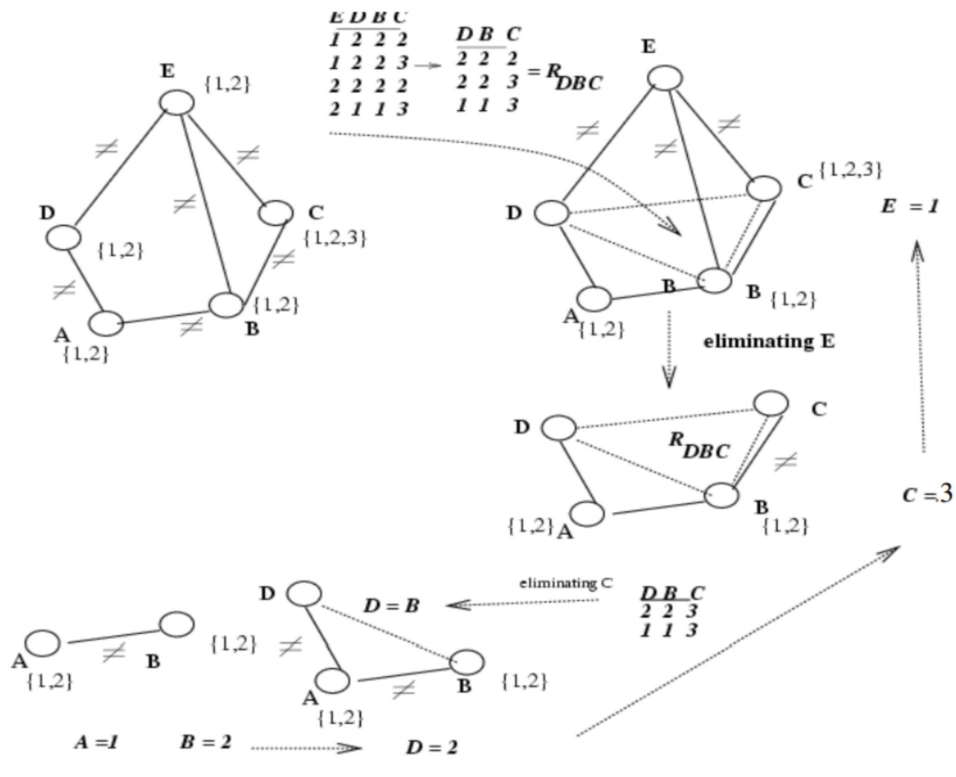

Figure 2: Full example of Bucket-elimination, followed by variable instantiation