Scribe Notes: 2/11/2013
Speaker: Dr. Berthe Y. Choueiry
Scribe: Robert Woodward

Topic: Directional Consistency (Chapter 4 of Dechter book)
Slides: http://cse.unl.edu/~choueiry/S13-921/Slides/Dechter-chapter04.pdf

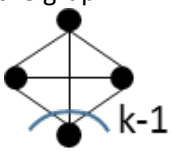
In these notes, I review three questions on Chapter 9 (Part A) then discuss slides from Chapter 4 (Part B).

Part A: Questions from Chapter 9 (Tree Decomposition)

Questions by Daniel Dobos (https://piazza.com/class#spring2013/csce921/38)

Q: Page 255 Dechter mentions that "in general, the maximal clique size of an induced ordered graph equals its induced width + 1." I'm not entirely sure why this is.

A: Consider a clique of size k that is in the graph.



In any ordering, there will be some variable in the clique with k-1 parents.

As a general note, when saying induced graph, what the graph "is induced by" needs to be specified. Detcher usually means by triangulation.

Q: From lecture (and cse421, if I'm not mistaken), we talk about instantiation order and elimination order, but I haven't been able to find these terms in our textbook. Did I just overlook them?

A: The *elimination order* is the order that the variables are processed for propagation by directional consistency algorithms of Dechter (bottom up). The *instantiation order* is the order that the variables are assigned during search (top down). See the example in Fikayo's question below. The textbook is not always clear about which ordering is considered. The instructor recommends always specifying the ordering in order to increase readability and reduce misunderstanding.

Question by Fikayo Adetunji (https://piazza.com/class#spring2013/csce921/39)

Q: On page 264, Bucket C: reads as R(C,A), isn't that supposed to be R(B,C,A)?

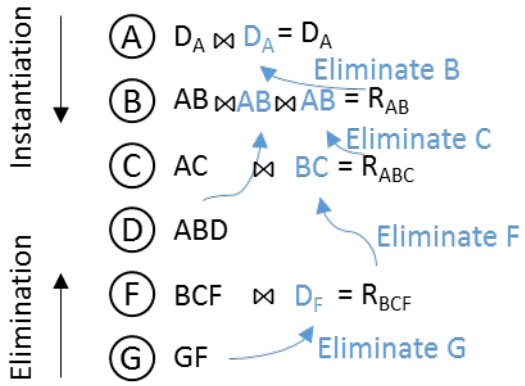
A: AC and AB are binary constraints. Notice, figure 9.13 is the *primal* graph, not the constraint network. The constraints in the network are: AB, AC, BCF, FG, ABD.

The example was then discussed in detail (see illustration in figure below). Solving using *bucket elimination*:

1. Consider the variables in the elimination ordering.
2. Associate a bucket with each variable
3. Place the relations in the deepest bucket in the elimination ordering where one of the variables in their scope appears in (The top bucket contains its variable's domain).
4. Going bottom-up (along the elimination ordering), join all the relations in the bucket, project out the variable of the bucket from the join, and place the result of the projection in the deepest bucket corresponding to of any of its variables. For example, after eliminating F, bucket C receives the constraint BC because it is the first bucket in the elimination ordering where its

scope appears (rather than going to bucket D). This step transfers the influence of the variable in the bucket by filtering the other variables, which allows us to remove it.

5. At the shallowest level, we filter the domain of the top variable.
6. If there is an empty constraint along the way, the problem is inconsistent, otherwise it is *guaranteed consistent*.
7. We can instantiate variables along the instantiation ordering in a backtrack-free manner.

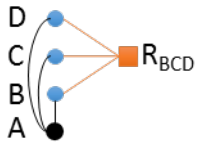


Part B: Directional Consistency (Chapter 4)

Slide 11: (From previous discussion) Why does the algorithm for directional i-consistency enforce strong directional i-consistency?
Robert proposed a possible solution (https://piazza.com/class#spring2013/csce921/37) by adding the following step at the beginning of the Revise(S, x_k), which enforces DAC:

$$\forall x_j \in S, \left[D_{x_j} \leftarrow D_{x_j} \cap \pi_{x_j}\left(R_{x_k,x_j} \bowtie D_{x_j}\right)\right]$$

In the figure below, Revise({B,C,D},A) will filter the domains (individually) of B, C, and D by step one (using R_{AB} to filter B, R_{AC} to filter C, and R_{AD} to filter D), and generate a new constraint R_{BCD} in step two (which is in the Revise definition).



Slides 12: Reviews definitions we had seen in CSCE421/821

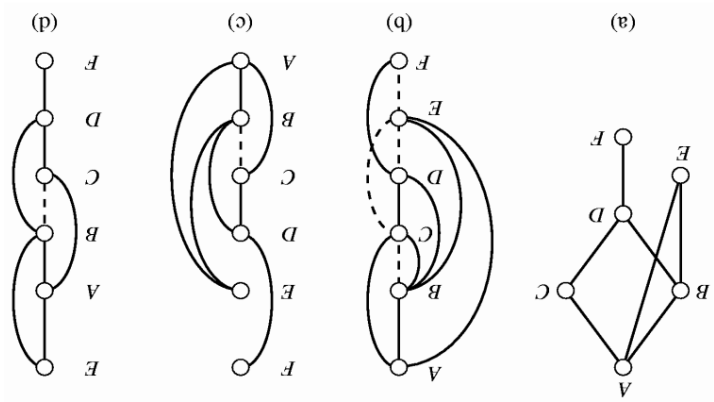
Induced-width (number of parents in induced graph, by marrying parents) is actually two terms:

1. induced width of an ordering (marry parents, then is direct)
2. induced width of a graph (NP-Hard)

Contrast with *width* (number of parents in graph):

1. width of an ordering (linearly computed)
2. width of a graph (quadratic time)

The example for width and induced width (Slide 15) is up-side-down WRT how we consider orderings in class. Flipping the slide:



(d) w*: 2 (c) w*: 2 (b) w*: 4 (a) Primal Network

- (b) A has parents B, C, so marry the parents (dotted line). Its induced width is 4.
- (c) & (d) have an induced width of 2. The induced width is not unique and two orderings have the same induced width. Therefore, you could possibly break ties when finding the best ordering (using domain size, or some other criterion).

Slide 17: Using induced width, we *can obtain a tighter bound for DPC*:

- Bound for DPC is: $O(n^3 \cdot a^3)$
- Using induced width bound: $O(w^*_d{}^2 \cdot n \cdot a^3)$

where w^*_d is the induced width according to ordering d, n is the number of variables, and a is the largest domain size.

In the new bound, two of the n's are replaced by w^*_d, because there is a bound to number of triangles (depends on the largest number of parents a vertex has, because every two parents form a triangle).

This bound is interesting/better because we are one step closer to single-parameter tractability. The exponent on n is constant, and we introduced a new parameter that does not depend on n.

Generalizing the above to DIC_i, the tighter bound becomes: $O(n \cdot (w^*_d)^i \cdot (2k)^i)$
Notice i is constant. Only parameter that varies is n, and the exponent is 1.

- i is the arity of the constraints being generated (enforcing level i+1).
- In DPC, binary constraints, arity=2, are being generated.

Slide 20: Discussed ordering heuristics that exploit the structure

- MinFill algorithm is empirically the best heuristic for selecting the ordering.
- See Shant's Working Note for efficient implementation: http://consystlab.unl.edu/our_work/StudentReports/Working-Note-1-2010.pdf.

Slide 21:

- Perfect graphs are graphs where ω(g) = max clique size, where ω(g) is the chromatic number of the graph, i.e., the max number of colors needed to color the graph.
- One subclass of perfect graphs are triangulated or chordal graphs.

- For chordal graphs, MaxClique algorithm can find the maximum cliques in polynomial time [Golumbic 1980, Algorithmic Graph Theory and Perfect Graphs].

Slide 27: Relation between width and consistency level

If a constraint network along an ordering has induced width w*= i+1, enforcing strong directional i-consistent, allows us to solve the problem in a backtrack-tree manner along the ordering.

- *Reason for i-consistency*? Consider a clique of size i, which is the largest clique in the problem. The bottleneck will be over this clique (recall Daniel's question in Part A).
- *Reason for backtrack-free*?
  - Parents are already married (considering induced width), enforcing i-consistency will not modify the structure (not changing induced width).
  - By forcing an ordering, we do not need to guarantee i-consistency, but only *directional* i-consistency.
  - Because we do not modify the structure by enforcing i-consistency, we can guarantee backtrack-free search.

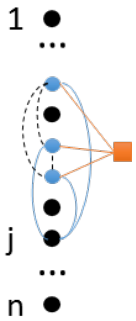
Slide 29: Adaptive Consistency

However, we do not always necessarily need to enforce the i everywhere. In A*daptive Consistency*, the algorithm looks at the parents, and apply only the level that is required. (This approach yields the CTE of Chapter 9 and bucket elimination algorithms.)

ADAPTIVE-CONSISTENCY (AC1)
Input: a constraint network $\mathcal{R} = (X, D, C)$, its constraint graph $G = (V, E)$, $d = (x_1, \ldots, x_n)$.
output: A backtrack-free network along d
Initialize: $C' \leftarrow C$, $E' \leftarrow E$
1. **for** $j = n$ to 1 do
2. Let $S \leftarrow parents(x_j)$.
3. $R_S \leftarrow Revise(S, x_j)$ (generate all partial solutions over S that can extend to x_j).
4. $C' \leftarrow C' \cup R_S$
5. $E' \leftarrow E' \cup \{(x_k, x_r) | x_k, x_r \in parents(x_j)\}$ (connect all parents of x_j)
5. **endfor**.

Notice, this revise uses the revise Robert proposed for directional i-consistency (which filters domains).

Idea of Adaptive Consistency:



- At j, take the parents (S) of the variable at level j
- Generate a new constraint over the parents and filter their domains (Revise). Note that we do not consider subsets of the parents, but generate one join of all the parents.
- Connect edges the parents (dashed lines), which corresponds to step 5 in the pseudo-code.

Q: Fikayo asked when considering j, would you ever want to filter the domain of j?

A: You could filter j, but it is useless work. Recall, going up the elimination ordering we are removing the influence of j, guaranteeing that when coming down the instantiation ordering we can continue the solution (easy to do with back-checking). Further, some of the supports in j might get deleted later on anyways, so it is better to save work from filtering j and focus first on getting to the top to guarantee consistency, then just build the solution.