Summary of Sections 4.4, 4.5, and 4.6

Speaker: Madelene Hardojo Scribe: Cate Anderson

February 28, 2003

1 Chapter 4, Section 4

At this point in our search for islands of tractability we have seen that it is highly desirable to have a criterian that could identify, in advance of ordinary search, the level of consistency sufficient for generating a backtrack-free representation for a given constraint network [1]. This backtrack free (BT-free)representation is critical because BT-free networks are tractable.

The approach taken was to link the level of consistency sufficient to guarantee BT-free search, with the shape of the graph. It is known that if the consistency level is greater than or equal to one plus the width of the graph, the network is BT-free [Freuder, 1985]. However, the width can increase while we are processing the consistency algorithm, which would require that the next higher level of consistency be achieved. Since the algorithms do not change the shape of the induced graph, this induced graph is used instead.

2 Chapter 4, Section 4.1

- For a tree ($w(d) = 1$) directional consistency over an ordering of width one is sufficient to guarantee BT-free network. Full Arc-consistency is overkill. If the tree-network is not Directional Arc-Consistent, it is enforced using `Revise(node, parent)`.
- When ring to solve a tree-structured CSP, the ordering must have width $w(d) = 1$. It is possible for a tree to have orderings of width two or more, so not just any ordering is valid. An ordering with width over 1 would violate the theorem.
- The proof of theorem 4.41: Let $x_1, ..., x_n$ be an ordering of a tree. Let $x_1, ..., x_n$ be instantiated. Next, x_{i+1} is instantiated. Since the width is one, x_{i+1} has at most one parent and since that parent is directionally arc-consistent with x_{i+1}, then there is support for x_{i+1} and it can be instantiated. Since this argument can be made for all nodes, then the full ordering can be instantiated BT-free.
- The `Tree-Solving` algorithm uses this sequence:
  - Generate an ordering of width one
  - Going in reverse search order, revise the domain of each parent, such that all remaining values have support in the child. (Remember, since the width is one, each node as at most one parent).
  - If domain annihilation occurs, then there is no solution.
- - The complexity of the `Tree-Solving` algorithm is $O(nk^2)$, where n is the number of variables and k is the bound of the domains. This is the same as the complexity of `DAC` when applied to a graph of width 1. The complexity of Full Arc consistency which is acheived by sweeping `DAC` in both directions over the ordering, is $O(2nK^2)$ which reduces to $O(nK^2)$.

- It can be seen that the $O(nk^2)$ complexity of the `Tree-Solving` algorithm is smaller then the $O(nk^3)$ of `AC-3`, and while it only ties the complexity of `AC-4` it does not require the complicated data structures that `AC-4` does.

3 Chapter 4, Section 4.2

- Now the discussion moves to networks of width 2. The theorem states that if R (the constraint network) is both Directionally Arc-Consistent and Directionally Path-Consistent over a given ordering of width 2, then the network is BT-Free along this ordering.

- However, for a given ordering, while processing `DPC` (Directional Path-Consistency) edges may be added that transform the graph into one with a width greater than the original width, which, in this instance, means greater than 2. In this case, a BT-free network is no longer guaranteed.

- The importance of the induced graph is apparent. Since all parents have been connected in a induced graph, the `DPC` will not change the shape of the graph and if no domains are annihilated, a BT-Free network can be achieved.

- The consequence of a binary constraint network R having an induced width of 2 can be solved in linear time $O(nk^3)$. The min-induced-width algorithm (MIW) can determine if the graph has an ordering of width 2, in linear time.

4 Chapter 4, Section 4.2

- Now the discussion moves to non-binary networks (graphs of width greater than 2). The general idea is that given a non-binary network R and an ordering d (with width i), if R is strongly Directional $(i+1)$-Consistent, then the network is BT-Free along this ordering.

- However, in the application of the DIC_i, the width may increase to be i or greater, in which case Direction-$(i+1)$-Consistency would have to be applied. It should be noted that the resulting graph is subsumed by the induced graph along the ordering, which leads to the following procedure: Given a problem, select an ordering of small width $w(d)$, compute the induced width, $w^*(d)$ and then apply strong directional $(w^*(d)+1)$-consistency. The resulting network is back-track free.

5 Chapter 4, Section 4.2

- Here it is noted that `DAC`, `DPC`, `DIC` are not complete inference procedures. Even though they can render a graph directionally consistent, they do not guarantee the existence of a solution. What is needed is a procedure to make any problem BT-free relative to a given ordering, and to determine if there is a solution (complete inference).

- Adaptive consistency (`ADC1`) implements the procedure suggested in the last item of the previous section of the summary. Both `ADC1` and `AC` and apply strong Directional $(i+1)$-Consistency and the resulting graph is BT-free along the ordering. The adaptive nature of these algorithms stems from the fact that they adapt to the changing width of each node at the time of processing. The resulting network has width bounded by i because the ordering had an induced width of i.

- Outline of `ADC1`,

- Given a constraint network R and an ordering d, for each node, find its width.
- Establish `DIC` depending on the width of the current node at the time processing.
- Enforce $(i+1)$-Consistency, which revises the constraint of arity i.

- `Adaptive-C` takes the aspect of variable elimination. At each step, one variable and all of its related constraints are solved and a constraint is inferred on all the rest of the variables in the scope. By solved, it is meant to generate all partial solutions over the parents of a variable that can be extended to the variable.
- An alternate description of adaptive consistency uses a data structure called a bucket, which avoids all explicit reference to the constraint graph. This data structure can best be implemented as a hash-table.
- Bucket elimination:
  - One bucket is assigned to each variable.
  - Each constraint is placed in the bucket of the variable that appears latest in its scope.
  - The buckets are processed in the reverse order of search. The natural join of all the constraints in the bucket is taken and projected over the set of variables included in the scopes of the constraints minus the variable that the bucket is assigned to.
  - This new constraint is placed in the bucket of the variable that appears latest in it. It should be noted that even in a binary network, this new constraint can be non-binary.
- This algorithm has the effect of taking all the information in the preceding variables and compiling it into the constraints of the last variables processed.
- Given a network R and and ordering d, `Adaptive-Consistency` determines the consistency of R and implies $E_d(R)$ is a BT-free network along d.
- Time complexity of `AdaptC` is $O(n(2k)^{w^*+1})$ and the space complexity is $O(n(2k)^{w^*})$, where n is the number of variables, k is the domain size and w^* is the induced-width given and ordering.

6 Chapter 4, summary

- It is shown that there is a class of tractable problems based on induced width, with w^*-based tractability.
- Adaptive-Consistency transforms a network into an equivalent one from which every solution can be generated BT-free.
- Theorem 4.55 characterizes the type of problem that can be taken as input.
- The task of looking for island of tractability is accomplished by looking at the topology of the graph and/or the semantics of the graph.

References

[Freuder, 1985] Eugene C. Freuder. A Sufficient Condition for Backtrack-Bounded Search. *JACM*, 32 (4):755–761, 1985.