CSPs: Arc Consistency

CPSC 322 – CSPs 3

Textbook §4.5

Lecture Overview

Constraint Satisfaction Problems: Definition

Definition

A constraint satisfaction problem consists of:

- a set of variables
- a domain for each variable
- a set of constraints

Definition

A model of a CSP is an assignment of values to variables that satisfies all of the constraints.

CSPs as Search Problems

We map CSPs into search problems:

- nodes: assignments of values to a subset of the variables
- neighbours of a node: nodes in which values are assigned to one additional variable
- start node: the empty assignment (no variables assigned values)
- goal node: a node which assigns a value to each variable, and satisfies all of the constraints

Note: the path to a goal node is not important

Lecture Overview

1. Recap
2. Consistency
3. Arc Consistency

Consistency Algorithms

- Idea: prune the domains as much as possible before selecting values from them.

Definition

A variable is domain consistent if no value of the domain of the node is ruled impossible by any of the constraints.

- Example: $dom(B) = \{1, 2, 3, 4\}$ isn't domain consistent if we have the constraint $B \neq 3$.

Constraint Networks

- Domain consistency only talked about constraints involving a single variable
  - what can we say about constraints involving multiple variables?

Definition

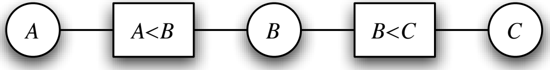
A constraint network is defined by a graph, with

- one node for every variable
- one node for every constraint

and undirected edges running between variable nodes and constraint nodes whenever a given variable is involved in a given constraint.

- When all of the constraints are binary, constraint nodes are not necessary: we can drop constraint nodes and use edges to indicate that a constraint holds between a pair of variables.
  - why can't we do the same with general k-ary constraints?

Example Constraint Network



Recall:

- Variables: A, B, C
- Domains: $\{1, 2, 3, 4\}$
- Constraints: $A < B$, $B < C$

Lecture Overview

1. Recap
2. Consistency
3. Arc Consistency

Arc Consistency

Definition

An arc $\langle X, r(X, \bar{Y}) \rangle$ is arc consistent if for each value of X in $dom(X)$ there is some value $\bar{Y}$ in $dom(\bar{Y})$ such that $r(X, \bar{Y})$ is satisfied.

- In symbols, $\forall X \in dom(X)$, $\exists \bar{Y} \in dom(\bar{Y})$ such that $r(X, \bar{Y})$ is satisfied.
- A network is arc consistent if all its arcs are arc consistent.
- If an arc $\langle X, \bar{Y} \rangle$ is *not* arc consistent, all values of X in $dom(X)$ for which there is no corresponding value in $dom(\bar{Y})$ may be deleted from $dom(X)$ to make the arc $\langle X, \bar{Y} \rangle$ consistent.
  - This removal can never rule out any models (do you see why?)

Arc Consistency Outcomes

- Three possible outcomes (when all arcs are arc consistent):
  - One domain is empty $\Rightarrow$ no solution
  - Each domain has a single value $\Rightarrow$ unique solution
  - Some domains have more than one value $\Rightarrow$ may or may not be a solution
    - in this case, arc consistency isn't enough to solve the problem: we need to perform search

Arc Consistency Algorithm

- Consider the arcs in turn making each arc consistent.
  - An arc $\langle X, r(X, \bar{Y}) \rangle$ needs to be revisited if the domain of Y is reduced.
- Regardless of the order in which arcs are considered, we will terminate with the same result: an arc consistent network.
- Worst-case complexity of this procedure:
  - let the max size of a variable domain be d
  - let the number of constraints be e
  - complexity is $O(ed^3)$
- Some special cases are faster
  - e.g., if the constraint graph is a tree, arc consistency is $O(ed)$

Arc Consistency Algorithm (binary constraints case)

```
procedure AC(V, dom, R)
    Inputs
        V: a set of variables
        dom: a function such that dom(X) is the domain of variable X
        R: set of relations to be satisfied
    Output
        arc consistent domains for each variable
    Local
        D_X is a set of values for each variable X
    for each variable X do
        D_X ← dom(X)
    end for each
    TDA ← {⟨X, r⟩ | r ∈ R is a constraint that involves X}
    while TDA ≠ {} do
        select ⟨X, r⟩ ∈ TDA;
        TDA ← TDA − {⟨X, r⟩};
        ND_X ← {x | x ∈ D_X and there is ȳ ∈ D_Ȳ such that r(x, ȳ)};
        if ND_X ≠ D_X then
            TDA ← TDA ∪ {⟨Z, r'⟩ | r' ≠ r and r' involves X and Z ≠ X};
            D_X ← ND_X;
        end if
    end while
    return {D_X : X is a variable}
end procedure
```

Adding edges back to TDA (binary constraints case)

- When we change the domain of a variable X in the course of making an arc $\langle X, r \rangle$ arc consistent, we add every arc $\langle Z, r' \rangle$ where r' involves X and:
  - $r \neq r'$
  - $Z \neq X$

- Thus we don't add back the same arc:
  - This makes sense—it's definitely arc consistent.

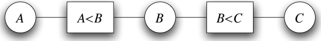
Adding edges back to TDA (binary constraints case)

- When we change the domain of a variable X in the course of making an arc $\langle X, r\rangle$ arc consistent, we add every arc $\langle Z, r'\rangle$ where r' involves X and:
  - $r \neq r'$
  - $Z \neq X$
- We don't add back other arcs that involve the same variable X
  - We've just *reduced* the domain of X
  - If an arc $\langle X, r\rangle$ was arc consistent before, it will still be arc consistent
    - in the "for all" we'll just check fewer values

Adding edges back to TDA (binary constraints case)

- When we change the domain of a variable X in the course of making an arc $\langle X, r\rangle$ arc consistent, we add every arc $\langle Z, r'\rangle$ where r' involves X and:
  - $r \neq r'$
  - $Z \neq X$
- We don't add back other arcs that involve the same constraint and a different variable:
  - Imagine that such an arc—involving variable Y—had been arc consistent before, but was no longer arc consistent after X's domain was reduced.
  - This means that some value in Y's domain could satisfy r only when X took one of the dropped values
  - But we dropped these values precisely because there were no values of Y that allowed r to be satisfied when X takes these values—contradiction!

Arc Consistency Example



- $dom(A) = \{1, 2, 3, 4\}; dom(B) = \{1, 2, 3, 4\}; dom(C) = \{1, 2, 3, 4\}$
- Suppose you first select the arc $\langle A, A < B \rangle$.
  - Remove $A = 4$ from the domain of A.
  - Add nothing to TDA.
- Suppose that $\langle B, B < C \rangle$ is selected next.
  - Prune the value 4 from the domain of B.
  - Add $\langle A, A < B \rangle$ back into the TDA set (why?)
- Suppose that $\langle B, A < B \rangle$ is selected next.
  - Prune 1 from the domain of B.
  - Add no element to TDA (why?)
- Suppose the arc $\langle A, A < B \rangle$ is selected next
  - The value $A = 3$ can be pruned from the domain of A.
  - Add no element to TDA (why?)
- Select $\langle C, B < C \rangle$ next.
  - Remove 1 and 2 from the domain of C.
  - Add $\langle B, B < C \rangle$ back into the TDA set

The other two edges are arc consistent, so the algorithm terminates with $dom(A) = \{1, 2\}$, $dom(B) = \{2, 3\}$, $dom(C) = \{3, 4\}$.