CSPs: Arc Consistency

CPSC 322 – CSPs 3

Textbook §4.5, 4.8

Lecture Overview

CSPs as Search Problems

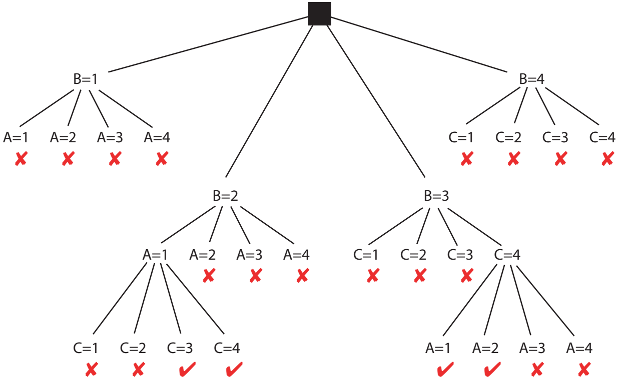
We map CSPs into search problems:

- 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

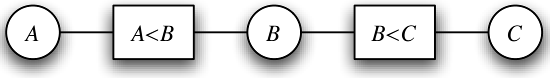
Example

An example of solving a CSP using depth-first search, with pruning whenever a partial assignment violates a constraint



Constraint Networks

- A constraint network:
  - Two kinds of nodes in the graph
    - one node for every variable
    - one node for every constraint
  - Edges run between variable nodes and constraint nodes: they indicate that a given variable is involved in a given constraint



Lecture Overview

1. Recap
2. 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 Algorithm

- Consider the arcs in turn making each arc consistent.
  - Arcs may need to be revisited whenever the domains of other variables are 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

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**
        $D_X \leftarrow dom(X)$
    end for each
    $TDA \leftarrow \{\langle X, r\rangle \mid r \in R$ is a constraint that involves $X\}$
    while $TDA \neq \{\}$ **do**
        select $\langle X, r\rangle \in TDA$;
        $TDA \leftarrow TDA - \{\langle X, r\rangle\}$;
        $ND_X \leftarrow \{x \mid x \in D_X$ and there is $\overline{y} \in D_{\overline{Y}}$ such that $r(x, \overline{y})\}$;
        if $ND_X \neq D_X$ **then**
            $TDA \leftarrow TDA \cup \{\langle Z, r'\rangle \mid r' \neq r$ and r' involves X and $Z \neq X\}$;
            $D_X \leftarrow ND_X$;
        end if
    end while
    return $\{D_X : X$ is a variable$\}$
end procedure

Adding edges back to TDA

- 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

- 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

- 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!