CSPs: Arc Consistency, Local Search

CPSC 322 Lecture 11

January 27, 2006
Textbook §3.5, 3.8

Lecture Overview

Recap

Arc Consistency

Local Search

CSPs as Search Problems

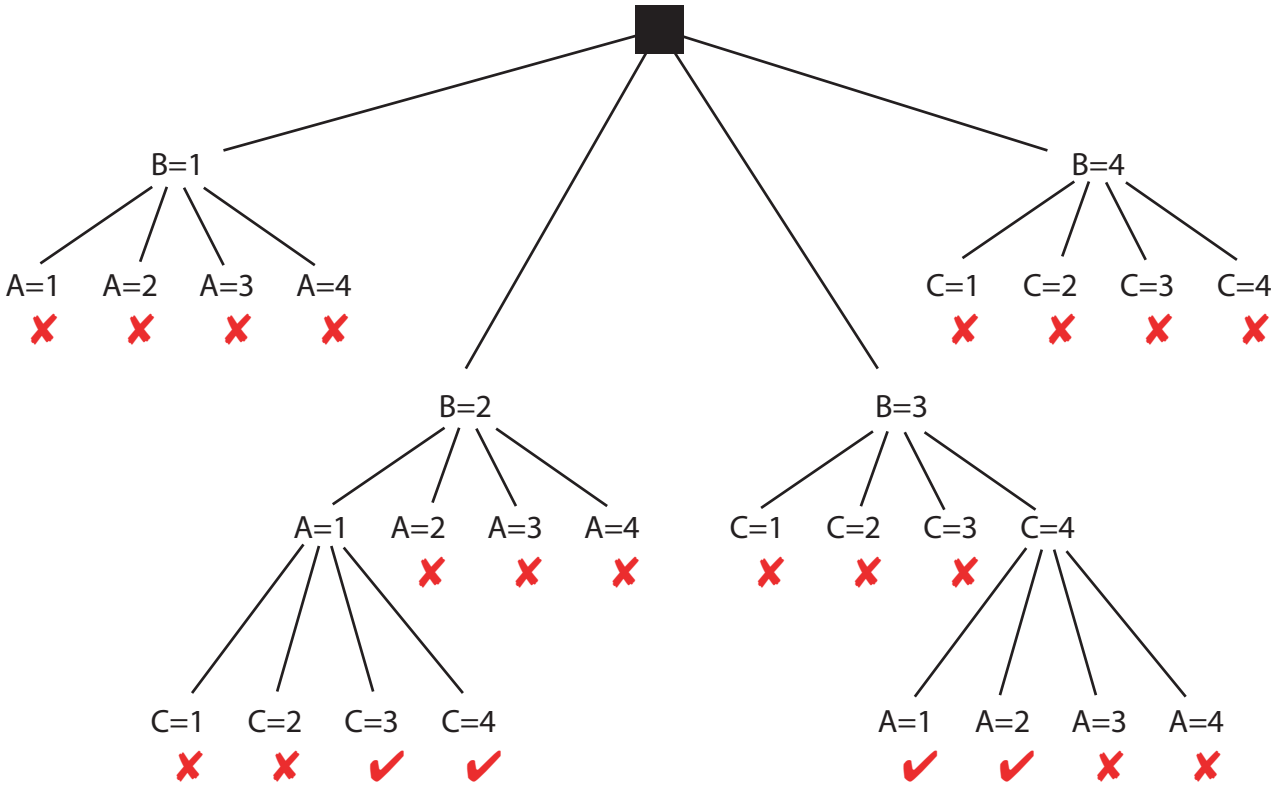
We map CSPs into search problems:

- 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)
- leaf node: a node which assigns a value to each variable
- goal node: leaf node which satisfies all of the constraints

Note: the path to a goal node is not important

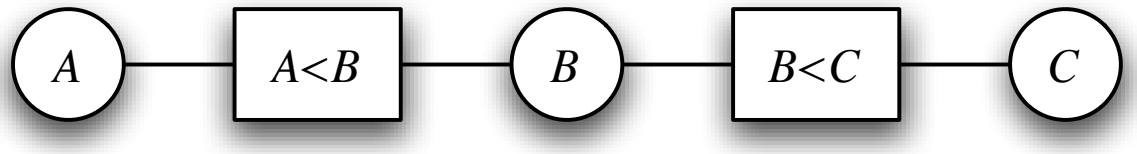
Example

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



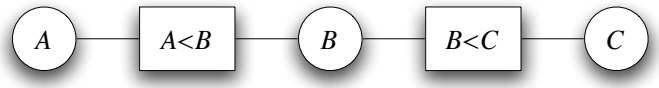
Arc Consistency

- An arc $\langle X, r(X, \bar{Y})\rangle$ is arc consistent if for each value of X in $\mathbf{D}_X$ there is some value for each $\bar{Y}$ in $\mathbf{D}_{\bar{Y}}$ such that $r(X, \bar{Y})$ is satisfied.
  - A network is arc consistent if all its arcs are arc consistent.
- If an arc $\langle X, Y\rangle$ is *not* arc consistent, all values of X in $\mathbf{D}_X$ for which there is no corresponding value in $\mathbf{D}_Y$ may be deleted from $\mathbf{D}_X$ to make the arc $\langle X, Y\rangle$ consistent.
  - This removal can never rule out any models

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

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 $\langle B, A < B \rangle$ is selected next.
  - Prune 1 from the domain of B.
  - Again add no element 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 the arc $\langle A, A < B \rangle$ is selected next
  - The value $A = 3$ can be pruned from the domain of A.
- The remaining arc on TDA is $\langle C, B < C \rangle$.
  - Remove 1 and 2 from the domain of C.
  - No arcs are added to TDA and TDA becomes empty.

The algorithm terminates with $dom(A) = \{1, 2\}$, $dom(B) = \{2, 3\}$, $dom(C) = \{3, 4\}$.

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

Local Search

- Many search spaces are too big for systematic search.
- A useful method in practice for some consistency and optimization problems is local search
  - idea: consider the space of complete assignments of values to variables
  - neighbours of a current node are similar variable assignments
  - move from one node to another according to a function that scores how good each assignment is

Local Search

Thus, a local search problem is defined by a:

- Set of Variables. A node in the search space will be a complete assignment to all of the variables.
- Neighbour relation. An edge in the search space will exist when the neighbour relation holds between a pair of nodes.
- Scoring function. This can be used to incorporate information about how many constraints are violated. It can also incorporate information about the cost of the solution in an optimization context.

Selecting Neighbours

How do we choose the neighbour relation?

- Usually this is simple: some small incremental change to the variable assignment
  - assignments that differ in one variable's value
  - assignments that differ in one variable's value, by a value difference of one
  - assignments that differ in two variables' values, etc.
- There's a trade-off: bigger neighbourhoods allow more nodes to be compared before a step is taken
  - the best step is more likely to be taken
  - each step takes more time: in the same amount of time, multiple steps in a smaller neighbourhood could have been taken
- Usually we prefer pretty small neighbourhoods