CSPs: Search and Arc Consistency

CPSC 322 Lecture 10

January 25, 2006
Textbook §3.3 – 3.5

Lecture Overview

Recap

Generate-and-Test

Search

Consistency

Arc Consistency

Variables

- We define the state of the world as an assignment of values to a set of variables
  - variable: a synonym for feature
  - we denote variables using capital letters
  - each variable V has a domain $dom(V)$ of possible values
- Variables can be of several main kinds:
  - Boolean: $|dom(V)| = 2$
  - Finite: the domain contains a finite number of values
  - Infinite but Discrete: the domain is countably infinite
  - Continuous: e.g., real numbers between 0 and 1
- We'll call the set of states that are induced by a set of variables the set of possible worlds

Constraints

Constraints are restrictions on the values that one or more variables can take

- Unary constraint: restriction involving a single variable
  - of course, we could also achieve the same thing by using a smaller domain in the first place
- k-ary constraint: restriction involving the domains of k different variables
  - it turns out that k-ary constraints can always be represented as binary constraints, so we'll often talk about this case
- Constraints can be specified by
  - giving a list of valid domain values for each variable participating in the constraint
  - giving a function that returns true when given values for each variable which satisfy the constraint
- A possible world satisfies a set of constraints if the set of variables involved in each constraint take values that are consistent with that constraint

Constraint Satisfaction Problems: Definition

A constraint satisfaction problem consists of:

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

Model: an assignment of values to variables that satisfies all of the constraints

Generate-and-Test Algorithm

- The assignment space of a CSP is the space of possible worlds
- Algorithm:
  - Generate possible worlds one at a time from the assignment space
  - Test them to see if they violate any constraints
- This procedure is able to solve any CSP
- However, the running time is proportional to the size of the state space
  - always exponential in the number of variables
  - far too long for many CSPs

CSPs as Search Problems

In order to think about better ways to solve CSPs, let's 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

CSPs as Search Problems

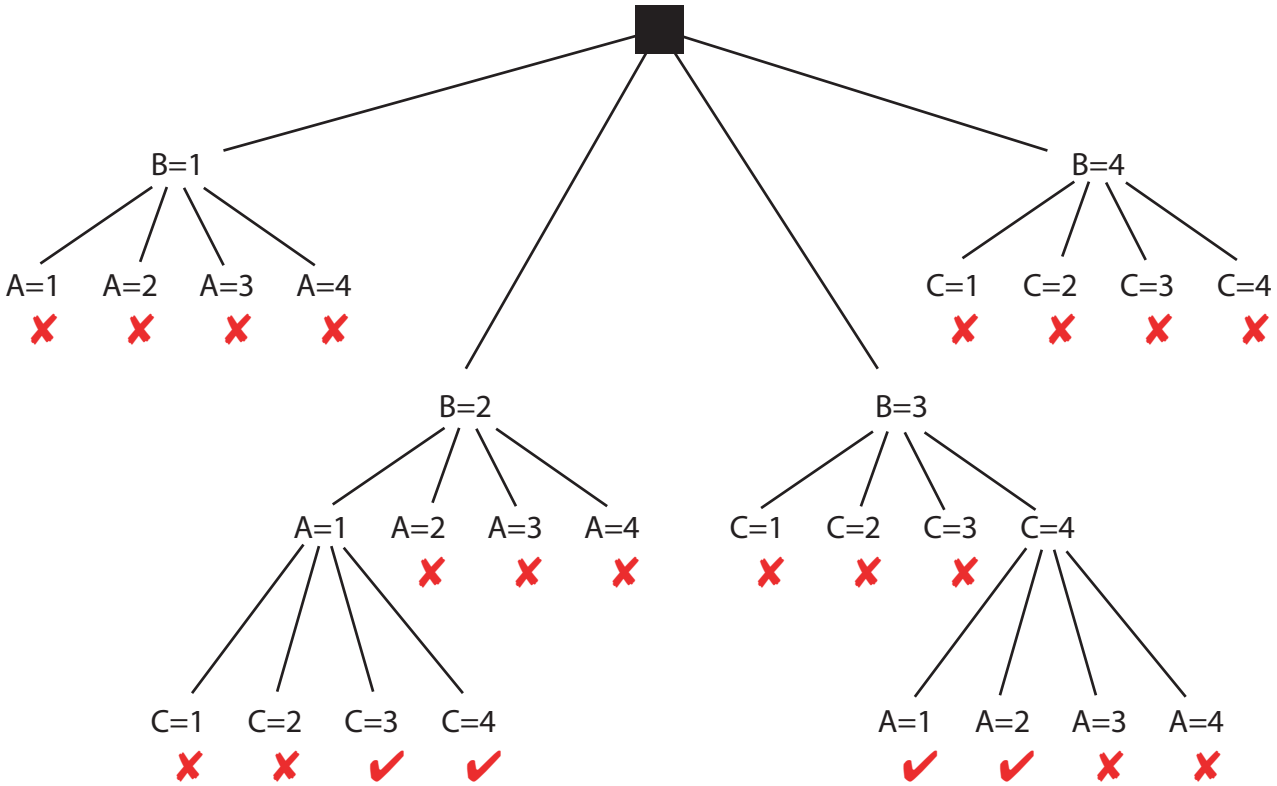
- What search strategy will work well for a CSP?
  - there are no costs, so there's no role for a heuristic function
  - the tree is always finite and has no cycles, so DFS is better than BFS
- How can we prune the DFS search tree?
  - once we reach a node that violates one or more constraints, we know that a solution cannot exist below that point
  - thus we should backtrack rather than continuing to search
  - this can yield us exponential savings over generate-and-test, though it's still exponential

Example

Problem:

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

Example



Note: the algorithm's efficiency depends on the order in which variables are expanded

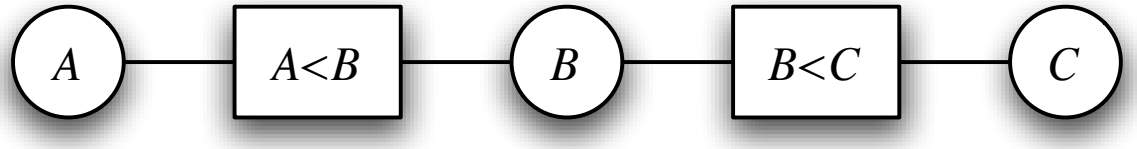
Consistency Algorithms

- Idea: prune the domains as much as possible before selecting values from them.
- A variable is domain consistent if no value of the domain of the node is ruled impossible by any of the constraints.
- Example: $\mathbf{D}_B = \{1, 2, 3, 4\}$ 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?
- First, let's define 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
- When all of the constraints are binary, constraint nodes are not necessary: in this case 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: $A < B$, $B < C$

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

- The arcs can be considered 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, it 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 $O(ed)$