CSPs: Search and Arc Consistency

CPSC 322 – CSPs 2

September 28, 2007
Textbook §4.3 – 4.5
Lecture Overview

1. Recap

2. Generate-and-Test

3. Search

4. Consistency

5. 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 $\text{dom}(V)$ of possible values

Variables can be of several main kinds:

- **Boolean**: $|\text{dom}(V)| = 2$
- **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
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.
Lecture Overview

1. Recap
2. Generate-and-Test
3. Search
4. Consistency
5. Arc Consistency
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.
Lecture Overview

1. Recap
2. Generate-and-Test
3. Search
4. Consistency
5. Arc Consistency
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
What search strategy will work well for a CSP?
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
- DFS is one way of implementing generate-and-test
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
  - DFS is one way of implementing generate-and-test
- How can we prune the DFS search tree?
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
  - DFS is one way of implementing generate-and-test

- 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
Lecture Overview

1 Recap
2 Generate-and-Test
3 Search
4 Consistency
5 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**: \( D_B = \{1, 2, 3, 4\} \) 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?

### 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?
Recall:

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

1 Recap
2 Generate-and-Test
3 Search
4 Consistency
5 Arc Consistency
Arc Consistency

**Definition**

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

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