CSPs: Search and Arc Consistency

CPSC 322 – CSPs 2

Textbook §4.3 – 4.5

Lecture Overview

1. Recap
2. Search
3. Consistency
4. 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

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. Search
3. Consistency
4. Arc Consistency

CSPs as Search Problems

In order to think about how to solve CSPs, let's map CSPs into search problems.

CSPs as Search Problems

In order to think about how 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?

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

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?

CSPs as Search Problems

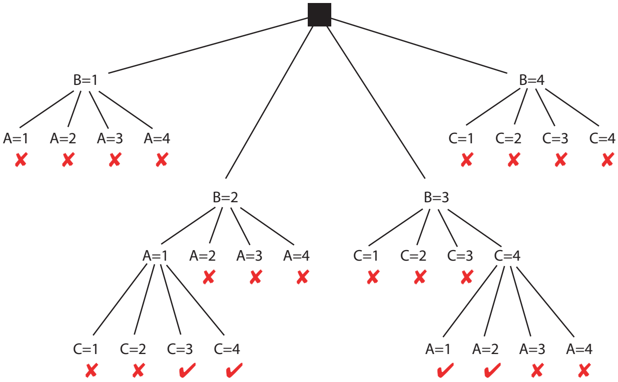
- What search strategy will work well for a CSP?
    - there are no costs, so there's no role for a heuristic function
    - the tree is always finite and has no cycles, so DFS is better than BFS
- How can we prune the DFS search tree?
    - once we reach a node that violates one or more constraints, 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 unpruned DFS, 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. Search
3. Consistency
4. 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

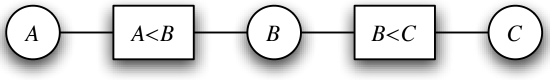
A constraint network is defined by a graph, with

- 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. Search
3. Consistency
4. 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.
  - 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

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!