CSPs: Search and Arc Consistency

CPSC 322 Lecture 10

January 29, 2007
Textbook §3.3 – 3.5

Lecture Overview

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

Lecture Overview

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

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.

Constraint Satisfaction Problems: Variants

We may want to solve the following problems with a CSP:

- determine whether or not a model exists
- find a model
- find all of the models
- count the number of models
- find the best model, given some measure of model quality
  - this is now an optimization problem
- determine whether some property of the variables holds in all models

CSPs: Game Plan

It turns out that even the simplest problem of determining whether or not a model exists in a general CSP with finite domains is $\mathcal{NP}$-hard

- we can't hope to find an efficient algorithm.

However, we can try to:

- find algorithms that are fast on "typical" cases
- identify special cases for which algorithms are efficient (polynomial)
- find approximation algorithms that can find good solutions quickly, even they may offer no theoretical guarantees
- develop parallel or distributed algorithms so that additional hardware can be used

Lecture Overview

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

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

Lecture Overview

1. Recap
2. CSPs
3. Generate-and-Test
4. Search
5. 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

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
  - 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

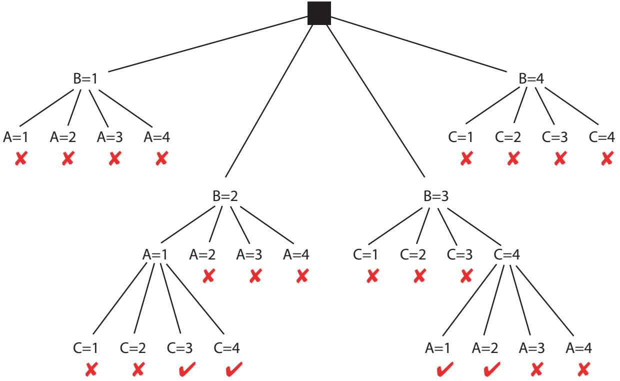
- What search strategy will work well for a CSP?
  - there are no costs, so there's no role for a heuristic function
  - the tree is always finite and has no cycles, so DFS is better than BFS
  - DFS is one way of implementing generate-and-test
- How can we prune the DFS search tree?
  - once we reach 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. CSPs
3. Generate-and-Test
4. Search
5. 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: $\mathbf{D}_B = \{1, 2, 3, 4\}$ 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?

Definition

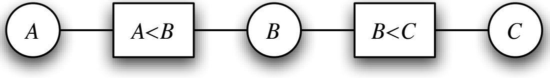
A constraint network is defined by a graph, with

- 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: $A < B$, $B < C$