CSP Introduction

CPSC 322 – CSPs 1

Textbook §4.0 – 4.2

Lecture Overview

Branch-and-Bound Search Algorithm

- Follow exactly the same search path as depth-first search
  - treat the frontier as a stack: expand the most-recently added node first
  - the order in which neighbors are expanded can be governed by some arbitrary node-ordering heuristic
- Keep track of a lower bound and upper bound on solution cost at each node
  - lower bound: $LB(n) = cost(n) + h(n)$
  - upper bound: $UB = cost(n')$, where n' is the best solution found so far.
    - if no solution has been found yet, set the upper bound to ∞.
- When a node n is selected for expansion:
  - if $LB(n) \geq UB$, remove n from frontier without expanding it
    - this is called "pruning the search tree" (really!)
  - else expand n, adding all of its neighbours to the frontier

Branch and Bound Example

- http://aispace.org/search/
- Example: Load from URL http://cs.ubc.ca/~kevinlb/teaching/cs322/BnBSearchDemo.xml

Summary of Search Strategies

Strategy	Frontier Selection	Complete?	Space
Depth-first	Last node added	No	Linear
Breadth-first	First node added	Yes	Exp
A^*	Minimal $f(n)$	Yes	Exp
Branch-and-Bound	Last node added, with pruning	No	Linear

Non-heuristic pruning

What can we prune besides nodes that are ruled out by our heuristic?

- Cycles
  - this one is really easy
- Multiple paths to the same node
  - if we want to maintain optimality, either keep the shortest path, or ensure that we always find the shortest path first

Other Search Ideas

The main problem with A^* is that it uses exponential space.
Branch and bound was one way around this problem.
Two others are:

- Iterative deepening
- Memory-bounded $A*$

Other search paradigms:

- Backwards search
- bi-directional search

Lecture Overview

1. Recap
2. Dynamic Programming
3. Variables
4. Constraints
5. CSPs

Dynamic Programming

Idea: for statically stored graphs, build a table of $dist(n)$ the actual distance of the shortest path from node n to a goal.

Initialize $dist(n) = \infty$ for each node n

Then repeatedly, until no $dist(n)$ value changes, set each $dist(n)$ value to the smallest (neighboring $dist(n')$ value + cost of reaching n' from n):

$$dist(n) = \begin{cases} 0 & \text{if } is\_goal(n), \\ \min_{\langle n,m\rangle \in A}(|\langle n,m\rangle| + dist(m)) & \text{otherwise.} \end{cases}$$

Dynamic Programming

There are two main problems:

- You need enough space to store the graph.
- The $dist$ function needs to be recomputed for each goal.

Complexity: polynomial in the size of the graph.

- but so is DFS (in fact, it's linear)
- the gain is when there are lots of nested cycles

Lecture Overview

Variables

- Recall that we defined the state of the world as an assignment of values to a set of (one or more) 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

Examples

- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are English words of the correct length
  - possible worlds: all ways of assigning words

Examples

- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are English words of the correct length
  - possible worlds: all ways of assigning words
- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - possible worlds: all ways of assigning letters to cells

Examples

- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are English words of the correct length
  - possible worlds: all ways of assigning words
- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - possible worlds: all ways of assigning letters to cells
- Sudoku
  - variables are cells
  - domains are numbers between 1 and 9
  - possible worlds: all ways of assigning numbers to cells

More Examples

- Scheduling Problem:
  - variables are different tasks that need to be scheduled (e.g., course in a university; job in a machine shop)
  - domains are the different combinations of times and locations for each task (e.g., time/room for course; time/machine for job)
  - possible worlds: time/location assignments for each task

More Examples

- Scheduling Problem:
  - variables are different tasks that need to be scheduled (e.g., course in a university; job in a machine shop)
  - domains are the different combinations of times and locations for each task (e.g., time/room for course; time/machine for job)
  - possible worlds: time/location assignments for each task
- n-Queens problem
  - variable: location of a queen on a chess board
    - there are n of them in total, hence the name
  - domains: grid coordinates
  - possible worlds: locations of all queens

Lecture Overview

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

Examples

- Crossword Puzzle:
  - variables are words that have to be filled in
  - domains are valid English words
  - constraints: words have the same letters at points where they intersect
- Crossword 2:
  - variables are cells (individual squares)
  - domains are letters of the alphabet
  - constraints: sequences of letters form valid English words
- Sudoku
  - variables are cells
  - domains are numbers between 1 and 9
  - constraints: rows, columns, boxes contain all different numbers

More Examples

- Scheduling Problem:
  - variables are different tasks that need to be scheduled (e.g., course in a university; job in a machine shop)
  - domains are the different combinations of times and locations for each task (e.g., time/room for course; time/machine for job)
  - constraints: tasks can't be scheduled in the same location at the same time; certain tasks can't be scheduled in different locations at the same time; some tasks must come earlier than others; etc.
- n-Queens problem
  - variable: location of a queen on a chess board
  - domains: grid coordinates
  - constraints: no queen can attack another

Lecture Overview

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