CSP Introduction

CPSC 322 Lecture 9

January 26, 2006
Textbook §3.0 – 3.2
Lecture Overview

1. Recap

2. Variables

3. Constraints
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: \( LB(n) = \text{cost}(n) + h(n) \)
  - upper bound: \( UB = \text{cost}(n') \), where \( n' \) is the best solution found so far.
    - if no solution has been found yet, set the upper bound to \( \infty \).
- 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
# Summary of Search Strategies

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Frontier Selection</th>
<th>Complete?</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth-first</td>
<td>Last node added</td>
<td>No</td>
<td>Linear</td>
</tr>
<tr>
<td>Breadth-first</td>
<td>First node added</td>
<td>Yes</td>
<td>Exp</td>
</tr>
<tr>
<td>Lowest-cost-first</td>
<td>Minimal $\text{cost}(n)$</td>
<td>Yes</td>
<td>Exp</td>
</tr>
<tr>
<td>Best-first</td>
<td>Global min $h(n)$</td>
<td>No</td>
<td>Exp</td>
</tr>
<tr>
<td>$A^*$</td>
<td>Minimal $f(n)$</td>
<td>Yes</td>
<td>Exp</td>
</tr>
<tr>
<td>Branch-and-Bound</td>
<td>Last node added, with pruning</td>
<td>No</td>
<td>Linear</td>
</tr>
</tbody>
</table>
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
- Dynamic programming
Lecture Overview

1 Recap

2 Variables

3 Constraints
States and Features

- In practical problems, there are usually too many states to reason about explicitly.
- However, the states usually have some internal structure, this is why people can understand the problem in the first place!
- **Features**: a set of variables that together define the state of the world.

- Many states can be described using few features:
  - 10 binary features ⇒ 1,024 states
  - 20 binary features ⇒ 1,048,576 states
  - 30 binary features ⇒ 1,073,741,824 states
  - 100 binary features ⇒ 1,267,650,600,228,229,401,496,703,205,376 states
So, 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.
Syntax and Semantics

- **Syntax**: the symbols that are manipulated by the computer, and the rules that are used to perform the manipulation

- **Semantics**: the meaning assigned to the symbols by the system designer
  - for example, the variable `black_queen_location` might correspond to the location on the chessboard of the black queen

- **Important point**: the computer only works at the syntactic level
  - it doesn’t understand what the symbols mean!
  - things that seem obvious to us must be made explicit
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

1 Recap

2 Variables

3 Constraints
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