CSCI 5582
Artificial Intelligence
Lecture 4
Jim Martin

Today 9/7
• Review
• Depth-limits
• Administration
• Uninformed and informed methods
• A* and Heuristics
• Beam search
• IDA*

Review
• What’s the primary way to specialize the general search function?
Review

Note: I’ll probably wind up using the terms agenda and queue (among others) fairly interchangeably to refer to the list of generated but not yet explored states (the nodes variable in the general search code).

Review

• What’s the difference between the book’s Tree-Search and Graph-Search algorithms.

BFS, Uniform Cost, DFS

• BFS
  – Insert new nodes at the end
• Uniform cost
  – Sort the agenda by cost
• DFS
  – Insert new nodes at the front
BFS and DFS Trade-offs

- **BFS**
  - Complete, memory inefficient, generally not optimal, generally slow
- **Uniform Cost**
  - Complete, optimal, generally memory inefficient, generally slow
- **DFS**
  - Not complete, not optimal, memory efficient, can be fast (or slow)

Iterative Deepening

Best of BFS and DFS

- Depth-limited DFS search with an ever increasing depth limit
  - Memory behavior of DFS
  - Completeness of BFS

ID-search, example

- Limit=0

[Diagram of ID-search]
ID-search, example

• Limit=1

ID-search, example

• Limit=2

ID-search, example

• Limit=3
Administration

• Homework questions?
  – import package vs. reload(package)
  – Current python is 2.4.3; 2.5 is scheduled for release in a couple of weeks; first release candidate is available now
  – Homework details
    • Lastname-mobiles.py means your last name not “last-name” and not “mobiles.py”.
    • Attach means attach as a file not include text in the message body.

Informed and Uninformed Techniques

• What do we know with uninformed methods?
  – What states we can get to from other states
  – The nodes that have been generated
  – We know a goal when we see it
  – We can know the cost of solution thus far

Informed and Uninformed Techniques

• So what are we uninformed about?
  – We’re uninformed about how close a given state is to a goal state
  – More precisely, we’re uninformed about the cost of achieving a goal state from some non-goal state
Informed Searches

- Review Uniform Cost
- Best first
- A*
- IDA*
- Recursive Best First Search

Uniform Cost

- One more time… what’s the basis for the ordering of nodes in uniform cost?
  - They’re examined in order of their cost so far (we’ll call this the g-cost).

Greedy (Apparently) Best First

- In this scheme, nodes are expanded on the basis of a guess about the cost of getting from a state to a goal (ignoring g, the cost so far).
- We’ll call a method for making such a guess a heuristic (call it h).
Greedy Example

- Cost function is $h(n)$ the guess about the cost of getting from $n$ to the goal
- In the map domain this could be the straight line distance from a city to Bucharest
- Greedy search expands the node that is currently closest to the goal

Greedy Search

- Complete?
  - Nope
- Optimal?
  - Nope
- Time and Space?
  - It depends
Best of Both

• In an A* search we combine the best parts of Uniform-Cost and Best-First.
• We want to use the cost so far to allow optimality and completeness, while at the same time using a heuristic to draw us toward a goal.

A*

In an A* search nodes are ordered according to g(n)+h(n).

• If h(n) does not overestimate the real cost then the search is optimal.
• An h function that does not overestimate is called admissible
Key Points

- Agenda is based on $g+h$
- $h$ must not overestimate to guarantee optimality
- The location of the goal test in the general search code is critical

Optimality of A*

- Proof by contradiction: Assume an A* search returned a non-optimal answer.
- What would that mean?
  - There’s another solution out there that costs less than the one that was returned as the answer.

A* Optimality

- What do you know about any node in the agenda that is on the path to this supposedly better solution?
  - Its cost is ≤ the solution cost. Why?
- What do you know about the relation between the cost of the solution found and this intermediate node?
  - Its cost is ≤ the intermediate node. Why?
A* Optimality

- So… The cost of the found solution is $\leq$ the cost of the supposedly better solution.
- This contradicts the assumption we began with.

A* and Intelligence

- Where’s the intelligence in an A* search?
  - In the heuristic

Admissible Heuristics

The 8-puzzle (a small version of the 15 puzzle).

Sample heuristics
- Number of misplaced tiles
- Manhattan distance
8 Puzzle Example

\[
\begin{array}{ccc}
5 & 4 & \\
6 & 1 & 8 \\
7 & 3 & 2
\end{array}
\quad
\begin{array}{ccc}
1 & 2 & 3 \\
8 & 4 & \\
7 & 6 & 5
\end{array}
\]

- H1(S) = 7
- H2(S) = 2+3+3+2+4+2+0+2 = 18
Which heuristic is better?

Sources of Heuristics

- Ad-hoc, informal, rules of thumb (guesswork)
- Approximate solutions to problems
- Exact solutions to different (relaxed) problems

Approximate Solution Example

- TSP is hard
- Minimum spanning tree is easy
- So… use MST to get an approximate solution to TSP
TSP/MST Example

TSP/MST Example

TSP/MST cont.

• An MST generated estimate is guaranteed to be no more than twice the optimal tour.
• How do you use that as an admissible heuristic?
Exact Solutions to Different Problems

- Transform the problem into a different (easier problem).
- Easier usually means a problem with some constraints removed or relaxed.
- The cost of an exact solution to a relaxed problem is an estimate of the cost of the real problem.

Restrictions on Heuristics

- Why not embed an exhaustive solution to the problem as a heuristic?
- More realistic issue: the more computationally complex a heuristic is, the less of the search space you'll be able to examine.

A* and Memory

- Does A* solve the memory problems with BFS and Uniform Cost?
  - Well sort of. Solve is a loaded term. A* has better memory performance than BFS or Uniform Cost.
  - But it might not enough better to make a difference in practical terms.
A* Agenda Management

• What mechanism does A* use to control the size of its internal agenda?
  – None
• So what happens when it gets too big?

Beam Search

• Simple solution
  – Chop the end of the agenda when a fixed limit is reached.
• What’s wrong with this?
  – Gives up optimality and completeness
• But this is a practical solution often used in real applications (commercial speech recognizers use beam search)

IDA*

• Use depth first memory management.
• But add an iteratively increasing depth bound.
• And make the bound based on g+h rather than depth in the tree.
Next Time

• Optimization search (sec. 4.3)
• Constraint sat search (Ch. 5)