

THE UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Uniformed Search

Many slides courtesy of Dan Klein, Stuart Russell, or Andrew Moore

CS 5300 / CS 6300
Artificial Intelligence
Spring 2009

Hal Daumé III
hal@cs.utah.edu

www.cs.utah.edu/~hal/courses/2009S_AI

Slide 1 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Announcements

- Forgot to tell you login information for web page:
 - User name = "ai" (but no quotes)
 - Password = "pacman" (still no quotes)
- CADE and Python:
 - Lab tomorrow (Friday) 1pm to 4pm in CADE
 - You may develop at home, but must *run* on CADE
- Homework 1 posted now
- Project 1 posted soon

Slide 2 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Today

- Agents that Plan Ahead
- Search Problems
 - Uniformed Search Methods
 - Depth-First Search
 - Breadth-First Search
 - Uniform-Cost Search

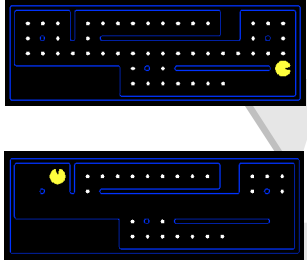
Slide 3 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Reflex Agents

- Reflex agents:
 - Choose action based on current percept and memory
 - May have memory or a model of the world's current state
 - Do not consider the future consequences of their actions
 - Can a reflex agent be rational?



[demo: reflex]

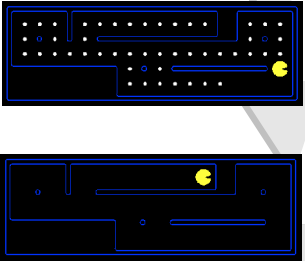
Slide 4 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Goal Based Agents

- Goal-based agents:
 - Plan ahead
 - Decisions based on (hypothesized) consequences of actions
 - Must have a model of how the world evolves in response to actions




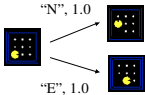
[demo: plan fast / slow]

Slide 5 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Daumé III (hal@cs.utah.edu)

Search Problems

- A search problem consists of:
 - A state space 
 - A successor function 
 - A start state and a goal test
- A solution is a sequence of actions (a plan) which transforms the start state to a goal state

Slide 6 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

Search Trees

- A search tree:
 - This is a "what if" tree of plans and outcomes
 - Start state at the root node
 - Children correspond to successors
 - Nodes labeled with states, correspond to PLANS to those states
 - For most problems, can never build the whole tree
 - So, have to find ways to use only the important parts!

Slide 7 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

State Space Graphs

- There's some big graph in which
- Each state is a node
- Each successor is an outgoing arc
- Important: For most problems we could never actually build this graph
- How many states in Pacman?

Laughably tiny search graph for a tiny search problem

Slide 8 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

Example: Romania

Slide 9 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

Another Search Tree

- Search:
 - Expand out possible plans
 - Maintain a **fringe** of unexpanded plans
 - Try to expand as few tree nodes as possible

Slide 10 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

General Tree Search

```

function TREE-SEARCH( problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
  end
    
```

- Important ideas:
 - Fringe
 - Expansion
 - Exploration strategy
- Main question: which fringe nodes to explore?

Detailed pseudocode is in the book!

Slide 12 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

Example: Tree Search

Slide 13 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

State Graphs vs Search Trees

Each NODE in the search tree is an entire PATH in the problem graph.

We almost always construct both on demand – and we construct as little as possible.

Slide 14 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

Review: Depth First Search

Strategy: expand deepest node first

Implementation: Fringe is a LIFO stack

Slide 15 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

Review: Breadth First Search

Strategy: expand shallowest node first

Implementation: Fringe is a FIFO queue

Search Tiers

Slide 16 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Daumé III (hal@cs.utah.edu)

Search Algorithm Properties

- Complete? Guaranteed to find a solution if one exists?
- Optimal? Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?

Variables:

| | |
|-------|---|
| n | Number of states in the problem |
| b | The average branching factor B (the average number of successors) |
| C^* | Cost of least cost solution |
| s | Depth of the shallowest solution |
| m | Max depth of the search tree |

Slide 17 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Dau

DFS

| | | | | |
|---------------------------|----------|---------|----------|----------|
| Algorithm | Complete | Optimal | Time | Space |
| DFS Depth First Search | N | N | Infinite | Infinite |

n # states
 b avg branch
 C^* least cost
 s shallow goal
 m max depth

- Infinite paths make DFS incomplete...
- How can we fix this?

Slide 18 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH
Hal Dau

DFS

➤ With cycle checking, DFS is complete.

| | | | | |
|-------------------------|----------|---------|--------------|---------|
| Algorithm | Complete | Optimal | Time | Space |
| DFS w/ Path Checking | Y | N | $O(b^{m+1})$ | $O(bm)$ |

➤ When is DFS optimal?

Slide 19 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Dau

BFS

| | |
|-------|--------------|
| n | # states |
| b | avg branch |
| C^* | least cost |
| s | shallow goal |
| m | max depth |

| Algorithm | Complete | Optimal | Time | Space |
|-----------|----------|---------|--------------|----------|
| DFS | Y | N | $O(b^{m+1})$ | $O(bm)$ |
| BFS | Y | N* | $O(b^{s+1})$ | $O(b^s)$ |

s tiers

1 node
b nodes
 b^2 nodes
 b^3 nodes
 b^m nodes

➤ When is BFS optimal?

Slide 20 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Dau

Comparisons

| | |
|-------|--------------|
| n | # states |
| b | avg branch |
| C^* | least cost |
| s | shallow goal |
| m | max depth |

- When will BFS outperform DFS?
- When will DFS outperform BFS?

Slide 21 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Dau

Iterative Deepening

Iterative deepening uses DFS as a subroutine:

- Do a DFS which only searches for paths of length ≤ 1 (DFS gives up on path of length 2)
- If "1" failed, do a DFS which only searches paths of length 2 or less.
- If "2" failed, do a DFS which only searches paths of length 3 or less.and so on.

| Algorithm | Complete | Optimal | Time | Space |
|-----------|----------|---------|--------------|----------|
| DFS | Y | N | $O(b^{m+1})$ | $O(bm)$ |
| BFS | Y | N* | $O(b^{s+1})$ | $O(b^s)$ |
| ID | Y | N* | $O(b^{s+1})$ | $O(bs)$ |

Slide 22 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Dau

Costs on Actions

Notice that BFS finds the shortest path in terms of number of transitions. It does not find the least-cost path. We will quickly cover an algorithm which does find the least-cost path.

Slide 23 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Dau

Uniform Cost Search

Expand cheapest node first:
Fringe is a priority queue

Cost contours

Slide 24 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH

Hal Dau

Priority Queue Refresher

- A priority queue is a data structure in which you can insert and retrieve (key, value) pairs with the following operations:

| | |
|----------------------------------|---|
| <code>pq.push(key, value)</code> | inserts (key, value) into the queue. |
| <code>pq.pop()</code> | returns the key with the lowest value, and removes it from the queue. |

- You can promote or demote keys by resetting their priorities
- Unlike a regular queue, insertions into a priority queue are not constant time, usually $O(\log n)$
- We'll need priority queues for most cost-sensitive search methods.

Slide 25 CS 5300: Introduction to AI

Hal Daumé III (hal@cs.utah.edu)

Uniform Cost Search

n # states
 b avg branch
 C^* least cost
 s shallow goal
 m max depth

| Algorithm | Complete | Optimal | Time | Space |
|-----------|----------|---------|---------------------------|-----------------------|
| DFS | Y | N | $O(b^{m+1})$ | $O(bm)$ |
| BFS | Y | N | $O(b^{s+1})$ | $O(b^s)$ |
| UCS | Y* | Y | $O(C^* b^{C^*/\epsilon})$ | $O(b^{C^*/\epsilon})$ |

C^*/ϵ tiers

We'll talk more about uniform cost search's failure cases later...

Slide 27 CS 5300: Introduction to AI

Hal Daumé III (hal@cs.utah.edu)

Uniform Cost Problems

- Remember: explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
 - Explores options in every "direction"
 - No information about goal location

[demo: ucs contours]
 CS 5300: Introduction to AI

Slide 28

Hal Daumé III (hal@cs.utah.edu)

Heuristics

Straight-line distance to Bucharest

| | |
|----------------|-----|
| Arad | 366 |
| Bucharest | 0 |
| Cluj | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 178 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 244 |
| Neamt | 214 |
| Oradea | 380 |
| Pitesti | 98 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

Slide 29 CS 5300: Introduction to AI

Hal Daumé III (hal@cs.utah.edu)

Best First / Greedy Search

- Expand the node that seems closest...

- What can go wrong?

Slide 30 CS 5300: Introduction to AI

Hal Daumé III (hal@cs.utah.edu)

Best First / Greedy Search

Slide 31 CS 5300: Introduction to AI

Hal Daumé III (hal@cs.utah.edu)

Best First / Greedy Search

- A common case:
 - Best-first takes you straight to the (wrong) goal
- Worst-case: like a badly-guided DFS in the worst case
 - Can explore everything
 - Can get stuck in loops if no cycle checking
- Like DFS in completeness (finite states w/ cycle checking)

Slide 32 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Search Gone Wrong?

Slide 33 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Extra Work?

- Failure to detect repeated states can cause exponentially more work. Why?

Slide 34 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Graph Search

- In BFS, for example, we shouldn't bother expanding the circled nodes (why?)

Slide 35 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Graph Search

- Very simple fix: never expand a state type twice

```

function GRAPH-SEARCH(problem, fringe) returns a solution, or failure
  closed ← an empty set
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    if STATE[node] is not in closed then
      add STATE[node] to closed
      fringe ← INSERTALL(EXPAND(node, problem), fringe)
  end
  
```

- Can this wreck completeness? Why or why not?
- How about optimality? Why or why not?

Slide 36 CS 5300: Introduction to AI

THE UNIVERSITY OF UTAH Hal Daumé III (hal@cs.utah.edu)

Some Hints

- Graph search is almost always better than tree search (when not?)
- Fringes are sometimes called "closed lists" – but don't implement them with lists (use sets)!
- Nodes are conceptually paths, but better to represent with a state, cost, and reference to parent node

Slide 37 CS 5300: Introduction to AI