CS 232: Artificial Intelligence

Spring 2024

Prof. Carolyn Anderson
Wellesley College
Reminders

✦ Homework 3 will be released today
✦ Lepei has help hours Thursday
✦ I have help hours Friday

*DALL-E3 labeled chocolate cross-section from AI Weirdness blog*
Recap
Search Trees

Choose leaf node from frontier for expansion according to the search strategy.

Determines the search process.
function **GRAPH-SEARCH**(**PROBLEM**) returns a solution or failure
initialize the FRONTIER using the start state of the **PROBLEM**
initialize the EXPLORED set to be empty
loop do
  if the FRONTIER is empty:
    return failure
  else:
    choose a leaf node by **STRATEGY** and remove it from FRONTIER
    if the node contains a goal state:
      return the corresponding solution
    else:
      add node to EXPLORED
      expand the chosen node to get set of children
      for each child:
        if child not in FRONTIER and child not in EXPLORED:
          add child to FRONTIER
Search Strategies

Review: *Strategy* = order of tree expansion
  - Implemented by different queue structures (LIFO, FIFO, priority)

Dimensions for evaluation
  - *Completeness* - always find the solution?
  - *Optimality* - finds a least cost solution (lowest path cost) first?
  - *Time complexity* - # of nodes generated (*worst case*)
  - *Space complexity* - # of nodes simultaneously in memory (*worst case*)

Time/space complexity variables
  - \( b \), *maximum branching factor* of search tree
  - \( d \), *depth* of the shallowest goal node
  - \( m \), maximum length of any path in the state space (potentially \( \infty \))
Search Conundrum

Breadth-first

- Complete,
- Optimal
- but uses $O(b^d)$ space

Depth-first

- Not complete unless $m$ is bounded
- Not optimal
- Uses $O(b^m)$ time; terrible if $m >> d$
- but only uses $O(b^m)$ space
Depth-limited search: A building block

Depth-First search *but with depth limit* $l$.
- i.e. nodes at depth $l$ *have no successors*.
- No infinite-path problem!

If $l = d$ (by luck!), then optimal
- But:
  - If $l < d$ then incomplete 😞
  - If $l > d$ then not optimal 😞

Time complexity: $O(b^l)$
Space complexity: $O(bl)$ 😞
# Summary of algorithms

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Breadth-First</th>
<th>Depth-First</th>
<th>Depth-limited</th>
<th>Iterative deepening</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td><strong>YES</strong></td>
<td><strong>NO</strong></td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>Time</td>
<td>(b^d)</td>
<td>(b^m)</td>
<td>(b^l)</td>
<td>(b^d)</td>
</tr>
<tr>
<td>Space</td>
<td>(b^d)</td>
<td>(bm)</td>
<td>(bl)</td>
<td>(bd)</td>
</tr>
<tr>
<td>Optimal?</td>
<td><strong>YES</strong></td>
<td><strong>NO</strong></td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
</tr>
</tbody>
</table>
Informed Search
Uninformed Search

Uses only information available in problem definition

Informally:

**Uninformed search**: All non-goal nodes in frontier look equally good

**Informed search**: Some non-goal nodes can be ranked above others.
Informed Search

An informed search strategy uses domain-specific information about the location of the goals in order to find a solution more efficiently than uninformed search.

Hints will come as part of a heuristic function denoted $h(n)$.

One of the most famous informed search algorithms is $A^*$ which was developed for robot navigation.

Shakey the robot was developed at the Stanford Research Institute from 1966 to 1972.
Motivation: Navigation Tasks

So far, we have assumed that all actions have the same cost. But this isn't true for many applications.

Some of these buildings are much closer than others!
g(N): the path cost function

- Our assumption so far: All moves equal in cost
  - Cost = # of nodes in path-1
  - \( g(N) = \text{depth}(N) \) in the search tree

\[
C(i,j) = \text{Cost of going from } N_i \text{ to } N_j
\]
\[
C(\text{Founders, Servett}) = 15
\]

\[
g(\text{Sci}) = C(\text{Founders, Green}) + C(\text{Green, Sci})
\]

If No is the start state, then \( g(N_3) = C(0,1) + C(1,2) + C(2,3) \)
Uniform-cost search (UCS)

Extend BFS:

Expand the node with the lowest path cost

Frontier:

priority queue ordered by \( g(n) \)

Subtle difference:

Test if a node is too good when it is expanded
not when it is added to the frontier
Uniform-cost search (UCS)

Before updating Jewett cost:

After updating Jewett cost:

Frontier

- Green: 2
- Pendleton: 7
- Jewett: 15
- Davis: 15
- Si: 24
- Lulu: 27

Visited

- Green
- Pendleton
- Jewett
- Davis
Shape of Search

- **Breadth First Search** explores equally in all directions. Its frontier is implemented as a FIFO queue. This results in smooth contours or “plys”.

- **Uniform Cost Search** lets us prioritize which paths to explore. Instead of exploring all possible paths equally, it favors lower cost paths. Its frontier is a priority queue. This results in “cost contours”.

Slides adapted from Chris Callison-Burch
A Better Idea...

- Node expansion based on *an estimate* which *includes distance to the goal*
- General approach of informed search:
  - *Best-first search*: node selected for expansion based on an *evaluation function* $f(n)$
    - $f(n)$ includes *estimate* of distance to goal (*new idea!*)
  - Implementation: Sort frontier queue by this new $f(n)$.
    - Special cases: *greedy search*, and *A*\(^*\) *search*
Greedy Best-First Search
Greedy Best First Search

Idea: expand the node that is estimated to be closest to the goal. Ignore the actual cost $g(n)$ and rely totally on the heuristic $h(n)$.

In our navigation example, this is like relying on the straight-line distance from a building to the goal (estimated from Google Maps).
Greedy Best-First Search

Frontier
- Jewett
- Davis
- Lulu
- Pendleton
- Green

Visited
- Founders
- Jewett
- Davis
- Lulu
- Pendleton
- Green

Straight-line Distance to Lulu
- Davis 10
- Founders 30
- Global Flora 40
- Green 20
- Jewett 15
- Lulu 0
- Munger 25
- Pendleton 16
- Sci 15

Building
Properties of Greedy Best First Search

Complete? Yes

Optimal? No!

\[ F \rightarrow J \rightarrow D \rightarrow L : 32 \]
\[ F \rightarrow J \rightarrow P \rightarrow D \rightarrow L : 42 \]
\[ F \rightarrow G \rightarrow P \rightarrow D \rightarrow J : 27 \]
Properties of Greedy Best First Search

**Complete?**  YES

**Optimal?**  NO!

Path found:
Founders -> Jewett -> Davis -> LuLu
Cost:  32

Best path:
Founders -> Green -> Pendleton -> Davis -> LuLu
Cost:  27
A* Search
A* search

Key Idea: Avoid expanding paths that are known to be expensive, but expand most promising paths first.

Simple: $f(n) = g(n) + h(n)$

- $g(n)$: actual cost so far of going from start to $n$
- $h(n)$: estimated cost of reaching goal from $n$
- $f(n)$: estimated total cost of start to goal through $n$

Implementation: frontier is a priority queue
Key concept: Admissible heuristics

A heuristic $h(n)$ is admissible if it never overestimates the cost to reach the goal.

Formally, for a node $n$:

$$h(n) \leq h^*(n)$$

where $h^*(n)$ is the true cost from $n$ to goal

$$h(n) \geq 0 \quad h(\text{goal}) = 0$$

If $h(n)$ is admissible, $A^*$ is optimal.
Idea: Admissibility

Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the frontier

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

Slides adapted from Chris Callison-Burch
A* Search Example

Frontier
- Green: 2 + 20
- Pendleton: 7 + 16 = 23
- Davis: 15 + 10 = 25
- Lulu: 27 + 0
- Jewett: x / 15 = 29
- Sci: 24 + 15 = 39

Visited
- Green
- Pendleton
- Davis
- Lulu
- Jewett
- Sci

Building | Straight-line Distance to LuLu
- Davis 10
- Founders 30
- Global Flora 40
- Green 20
- Jewett 15
- Lulu 0
- LuLu 0
- Munger 25
- Pendleton 16
- Sci 15

Founders → Green → Pendleton → Davis → Lulu
**UCS vs A* Contours**

Uniform-cost expands equally in all “directions”

A* expands mainly toward the goal, but does hedge its bets to ensure optimality
A* Applications

Pathing / routing problems (A* is in your GPS!)
Video games
Robot motion planning
Resource planning problems
...

Slides adapted from Chris Callison-Burch
Heuristics
Heuristic Functions

For the 8-puzzle

- *Avg. solution cost is about 22 steps*
  - *(branching factor $\leq 3$)*
  - *(branching factor $\leq 3$)*
  - *A good heuristic function can reduce the search process*
Admissible Heuristics

For the 8-puzzle:

- $h_{oop}(n) = \text{number of out of place tiles}$
- $h_{md}(n) = \text{total Manhattan distance (i.e., \# of moves from desired location of each tile)}$

$h_{oop}(S) = 8$

$h_{md}(S) = 3+1+2+2+2+3+3+2 = 18$
Key: Admissibility

Inadmissible (pessimistic) heuristics break optimality by pushing good plans too far back on the frontier, which means they may never get expanded.

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs. That means that the true best plan will always be expanded.

Slides adapted from Chris Callison-Burch