CS 540 Introduction to Artificial Intelligence
Search II: Informed Search
Josiah Hanna
University of Wisconsin-Madison
November 16, 2021
Announcements

- **Homeworks:**
  - HW 8 due next Tuesday.

- **Class roadmap:**

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, Nov 16</td>
<td>Search II: Informed search</td>
<td></td>
</tr>
<tr>
<td>Thursday, Nov 18</td>
<td>Advanced Search and Review on Search</td>
<td>HW7 Due; HW8 Released</td>
</tr>
<tr>
<td>Tuesday, Nov 23</td>
<td>Games - Part I</td>
<td></td>
</tr>
<tr>
<td>Thursday, Nov 25</td>
<td>Happy Thanksgiving! (No class)</td>
<td></td>
</tr>
<tr>
<td>Tuesday, Nov 30</td>
<td>Games - Part II</td>
<td></td>
</tr>
<tr>
<td>Thursday, Dec 2</td>
<td>Reinforcement Learning I</td>
<td>HW8 Due; HW9 Released</td>
</tr>
<tr>
<td>Tuesday, Dec 7</td>
<td>Reinforcement Learning II</td>
<td></td>
</tr>
<tr>
<td>Thursday, Dec 9</td>
<td>Review on Games and Reinforcement Learning</td>
<td></td>
</tr>
<tr>
<td>Tuesday, Dec 14</td>
<td>Ethics and Trust in AI</td>
<td>HW10 Due</td>
</tr>
</tbody>
</table>

*Everything below here is tentative and subject to change.*

**FINAL EXAM Dec 23**
Outline

• Uninformed vs Informed Search
  – Review of uninformed strategies, adding heuristics

• A* Search
  – Heuristic properties, stopping rules, analysis

• Extensions: Beyond A*
  – Iterative deepening, beam search
Breadth-First Search

Recall: expand **shallowest** node first

- Data structure: queue

**Properties:**
- Complete
- Optimal (if edge cost 1)
- Time $O(b^d)$
- Space $O(b^d)$
Uniform Cost Search

Like BFS, but keeps track of cost

- Expand least cost node
- Data structure: priority queue

Properties:
- Complete
- Optimal (if cost lower bounded by $\varepsilon$)
- Time $O(b^{c*/\varepsilon})$
- Space $O(b^{c*/\varepsilon})$

Optimal goal path cost
Depth-First Search

Recall: expand **deepest** node first

- Data structure: stack

**Properties:**
- Incomplete (stuck in infinite tree...)
- Suboptimal
- Time $O(b^m)$
- Space $O(bm)$
Iterative Deepening DFS

Repeated limited DFS

• Search like BFS, fringe like DFS

• **Properties:**
  – Complete
  – Optimal (if edge cost 1)
  – Time $O(b^d)$
  – Space $O(bd)$

A good option!
Uninformed vs Informed Search

Uninformed search (all of what we saw). Know:
- Path cost $g(s)$ from start to node $s$
- Successors.

Informed search. Know:
- All uninformed search properties, plus
- Heuristic $h(s)$ from $s$ to goal
Informed Search

Informed search. Know:

• All uninformed search properties, plus
• Heuristic \(h(s)\) from \(s\) to goal

Use information to **speed up search.**
Using the Heuristic

Back to uniform-cost search

• We had the priority queue
• Expand the node with the smallest $g(s)$
  – $g(s)$ “first-half-cost”
• Now let’s use the heuristic (“second-half-cost”)
  – Several possible approaches: let’s see what works
Attempt 1: Best-First Greedy

One approach: just use $h(s)$ alone

- Specifically, expand node with smallest $h(s)$
- This isn’t a good idea. Why?

Not optimal! Get $A \rightarrow C \rightarrow G$. Want: $A \rightarrow B \rightarrow C \rightarrow G$
Attempt 2: A Search

Next approach: use both $g(s) + h(s)$ alone

- Specifically, expand node with smallest $g(s) + h(s)$
- Again, use a priority queue
- Called “A” search

```
Fringe | g(s) + h(s)
------ |----------
A      | 3
B,C    | 1001,1000
B,G    | 1001,1000
A -> C -> G
```

- Still not optimal! (Does work for former example).
Attempt 3: A* Search

Same idea, use $g(s) + h(s)$, with one requirement

- Demand that $h(s) \leq h^*(s)$ – Optimal cost to goal
- If heuristic has this property, “admissible”
  - Optimistic! Never over-estimates
- Still need $h(s) \geq 0$
  - Negative heuristics can lead to strange behavior
- This is A* search
Attempt 3: A* Search

Origins: robots and planning

Shakey the Robot, 1960’s

Credit: Wiki

Animation: finding a path around obstacle

Credit: Wiki
Admissible Heuristic Functions

Have to be careful to ensure admissibility (optimism!)

- Example: 8 Game

One useful approach: relax constraints
  - $h(s) = \text{number of tiles in wrong position}$
    - allows tiles to fly to destination in a single step
Q 1.1: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the number of hours needed to ride a bike from city $s$ to your destination. $h(s)$ is

- A. An admissible heuristic
- B. Not an admissible heuristic
Q 1.1: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the number of hours needed to ride a bike from city $s$ to your destination. $h(s)$ is

- A. An admissible heuristic
- B. Not an admissible heuristic
Q 1.1: Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let $h(s)$ be the number of hours needed to ride a bike from city $s$ to your destination. $h(s)$ is

- A. An admissible heuristic No: riding your bike takes longer.
- B. Not an admissible heuristic
Q 1.2: Which of the following are admissible heuristics?

(i) \( h(s) = h^*(s) \)
(ii) \( h(s) = \max(2, h^*(s)) \)
(iii) \( h(s) = \min(2, h^*(s)) \)
(iv) \( h(s) = h^*(s) - 2 \)
(v) \( h(s) = \sqrt{h^*(s)} \)

- A. All of the below
- B. (i), (iii), (iv)
- C. (i), (iii)
- D. (i), (iii), (v)
Q 1.2: Which of the following are admissible heuristics?

(i)  \( h(s) = h^*(s) \)
(ii) \( h(s) = \max(2, h^*(s)) \)
(iii) \( h(s) = \min(2, h^*(s)) \)
(iv) \( h(s) = h^*(s) - 2 \)
(v) \( h(s) = \sqrt{h^*(s)} \)

- A. All of the below
- B. (i), (iii), (iv)
- C. (i), (iii)
- D. (i), (iii), (v)
Q 1.2: Which of the following are admissible heuristics?

(i) \( h(s) = h^*(s) \)
(ii) \( h(s) = \max(2, h^*(s)) \)  No: \( h(s) \) might be too big
(iii) \( h(s) = \min(2, h^*(s)) \)
(iv) \( h(s) = h^*(s) - 2 \)  No: \( h(s) \) might be negative
(v) \( h(s) = \sqrt{h^*(s)} \)  No: if \( h^*(s) < 1 \) then \( h(s) \) is bigger

- A. All of the below
- B. (i), (iii), (iv)
- C. (i), (iii)
- D. (i), (iii), (v)
Heuristic Function Tradeoffs

Dominance: $h_2$ dominates $h_1$ if for all states $s$,

$$h_1(s) \leq h_2(s) \leq h^*(s)$$

- **Idea**: we want to be as close to $h^*$ as possible
  - But not over!
- **Tradeoff**: being very close might require a very complex heuristic, expensive computation
  - Might be better off with cheaper heuristic & expand more nodes.
A* Termination

When should A* stop?

- One idea: as soon as we reach goal state?

- $h$ admissible, but note that we get $A \rightarrow B \rightarrow G$ (cost 1000)!
A* Termination

When should A* stop?

- **Rule**: terminate *when a goal is popped* from queue.

  ![Graph]

  - Note: taking $h = 0$ reduces to uniform cost search rule.
A* Revisiting Expanded States

Possible to revisit an expanded state, get a shorter path:

- Put D back into priority queue, smaller $g + h$

Fringe $g(s) + h(s)$

- A: 0+1
- B,C: 1+1,1+900
- C,D: 1+900,3+1
- C,G: 1+900,1002+0
- D,G: 2+1,1002+0
- G: 1001+0

A -> C -> G
A* Full Algorithm

1. Put the start node $S$ on the priority queue, called OPEN
2. If OPEN is empty, exit with failure
3. Remove from OPEN and place on CLOSED a node $n$ for which $f(n)$ is minimum (note that $f(n) = g(n) + h(n)$)
4. If $n$ is a goal node, exit (trace back pointers from $n$ to $S$)
5. Expand $n$, generating all successors and attach to pointers back to $n$. For each successor $n'$ of $n$
   1. If $n'$ is not already on OPEN or CLOSED estimate $h(n')$, $g(n') = g(n) + c(n, n')$, $f(n') = g(n') + h(n')$, and place it on OPEN.
   2. If $n'$ is already on OPEN or CLOSED, then check if $g(n')$ is lower for the new version of $n'$. If so, then:
      1. Redirect pointers backward from $n'$ along path yielding lower $g(n')$.
      2. Put $n'$ on OPEN.
   3. If $g(n')$ is not lower for the new version, do nothing.
6. Go to 2.
A* Analysis

Some properties:

- Terminates!
- A* can use **lots of memory**: $O(\# \text{ states})$.
- Will run out on large problems.

- Next, we will consider some alternatives to deal with this.
Q 2.1: Consider two heuristics for the 8 puzzle problem. $h_1$ is the number of tiles in wrong position. $h_2$ is the $l_1$/Manhattan distance between the tiles and the goal location. How do $h_1$ and $h_2$ relate?

- A. $h_2$ dominates $h_1$
- B. $h_1$ dominates $h_2$
- C. Neither dominates the other
Q 2.1: Consider two heuristics for the 8 puzzle problem. \( h_1 \) is the number of tiles in wrong position. \( h_2 \) is the \( l_1 / \text{Manhattan} \) distance between the tiles and the goal location. How do \( h_1 \) and \( h_2 \) relate?

- A. \( h_2 \) dominates \( h_1 \)
- B. \( h_1 \) dominates \( h_2 \)
- C. Neither dominates the other
Q 2.1: Consider two heuristics for the 8 puzzle problem. $h_1$ is the number of tiles in wrong position. $h_2$ is the $l_1$/Manhattan distance between the tiles and the goal location. How do $h_1$ and $h_2$ relate?

- A. $h_2$ dominates $h_1$
- B. $h_1$ dominates $h_2$ (No: $h_1$ is a distance where each entry is at most 1, $h_2$ can be greater)
- C. Neither dominates the other
Q 2.2: Consider the state space graph below. Goal states have bold borders. $h(s)$ is shown next to each node. What node will be expanded by A* after the initial state I?

- A.
- B.
- C.
Q 2.2: Consider the state space graph below. Goal states have bold borders. $h(s)$ is show next to each node. What node will be expanded by A* after the initial state I?

- A.
- B.
- C.
IDA*: Iterative Deepening A*

Similar idea to our earlier iterative deepening.

- Bound the memory in search.
- At each phase, don’t expand any node with $g(s) + h(s) > k$,
  - Assuming integer costs, do this for $k=0$, then $k=1$, then $k=2$, and so on

- Complete + optimal, might be costly time-wise
  - Revisit many nodes
- Lower memory use than A*
IDA*: Properties

How many restarts do we expect?
- With integer costs, optimal solution $C^*$, at most $C^*$

What about non-integer costs?
- Initial threshold $k$. Use the same rule for non-expansion
- Set new $k$ to be the min $g(s) + h(s)$ for non-expanded nodes
- Worst case: restarted for each state
Beam Search

General approach (beyond A* too)

- Priority queue with fixed size $k$; beyond $k$ nodes, **discard**!
- **Upside**: good memory efficiency
- **Downside**: not complete or optimal

Variation:

- Priority queue with nodes that are at most $\varepsilon$ worse than best node.
Recap and Examples

Example for A*:

```
Recall the A* algorithm:

A* is a search algorithm that finds the best path from the initial state to the goal state.

The algorithm uses two heuristics: the g-value, which represents the cost of the path from the initial state to the current state, and the h-value, which represents the estimated cost from the current state to the goal state.

A* selects the next state to expand based on the f-value, which is the sum of the g-value and the h-value:

f(n) = g(n) + h(n)

The algorithm terminates when the goal state is reached or when all possible states have been explored.

---

Initial state: S

Goal state: G

Heuristics:
- h(S) = 8
- h(A) = 7
- h(B) = 4
- h(C) = 3
- h(D) = ∞
- h(E) = ∞
- h(G) = 0

Algorithm steps:
1. Expand the initial state S.
2. Evaluate the states A, B, and C.
3. Expand the state with the lowest f-value.
4. Repeat until the goal state is reached.

```

The diagram illustrates the algorithm's steps with cost values and estimated costs.
Recap and Examples

Example for A*:

OPEN
S(0+8)
A(1+7) B(5+4) C(8+3)
B(5+4) C(8+3) D(4+inf) E(8+inf) G(10+0) S(0+8) A(1+7)
C(8+3) D(4+inf) E(8+inf) G(9+0)
C(8+3) D(4+inf) E(8+inf)

CLOSED
S(0+8)
S(0+8) A(1+7)
S(0+8) A(1+7) B(5+4)
S(0+8) A(1+7) B(5+4) G(9+0)

G → B → S

G → B → S
Recap and Examples

Example for IDA*: Threshold = 8

PREFIX
-  OPEN
S  S(0+8)
SA  A(1+7)
SAH  H(2+2) D(4+4)
SAHF  D(4+4) F(6+1)
SAFD  D(4+4)

Initial state

Goal state

OPEN
S(0+8)
A(1+7)
H(2+2) D(4+4)
D(4+4) F(6+1)
D(4+4)

PREFIX
-  S
S  S A
SA  S A H
SAH  S A H F
SAFD  S A D

Goal state

h=8
h=7
h=4
h=3
h=2
h=1
h=0
Example for IDA*:
Threshold = 9

Prefix  OPEN
-       S(0+8)
S       A(1+7) B(5+4)
SA      B(5+4) H(2+2) D(4+4)
SAH     B(5+4) D(4+4) F(6+1)
SAHF    B(5+4) D(4+4)
SAD     B(5+4)
SB      G(9+0)
S B G

Recap and Examples
Recap and Examples

Example for Beam Search: $k=2$

**CURRENT**
- 
S
A
H
D
F
D
G

**OPEN**
- 
S(0+8)
A(1+7) B(5+4)
H(2+2) D(4+4)
D(4+4) F(6+1)
D(4+4) G(10+0)

Goal state
- 
F

Initial state
- 
S
A
B
C

h=8
h=4
h=3
h=0
h=1
h=2
h=4
h=inf
h=inf
h=inf
h=inf
h=0
h=2
h=inf
h=inf
h=inf
h=inf
Summary

• Informed search: introduce heuristics
  – Not all approaches work: best-first greedy is bad
• A* algorithm
  – Properties of A*, idea of admissible heuristics
• Beyond A*
  – IDA*, beam search. Ways to deal with space requirements.
Acknowledgements: Adapted from materials by Jerry Zhu, Fred Sala (University of Wisconsin).