Announcements

• HW8 released today, due next Tuesday
• Annotated slides
• Grading Info
• Class roadmap

Today: informed search
Thursday: advanced search
Next week: Games.
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

Recall: expand **least-cost** node first

- Generalization of BFS
- Data structure: priority queue

**Properties:**
- Complete
- Optimal (if weight lower bounded by $\epsilon$)
- Time $O(b^{C*/\epsilon})$
- Space $O(b^{C*/\epsilon})$

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

\[ \begin{align*}
\text{start} & \quad \text{s} \quad \text{goal} \\
\quad & \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad g(s) \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad h(s)
\end{align*} \]

• Use side information to \textit{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

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

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

- Demand that $h(s) \leq h^*(s)$
- 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 take 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) \( h(s) = \min(2, h^*(s)) \) \( \leq h^*(s) \)
- (iv) \( h(s) = h^*(s) - 2 \) \( \text{may be negative} \)
- (v) \( h(s) = \sqrt{h^*(s)} \)

• A. All of the above
• 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 above
- 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 above
• 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.
Heuristic Function Tradeoffs

- Example: 8 Game

- Previous heuristic: $h_1(s) = \text{number of tiles in wrong position}$

- Better heuristic?

$$h_2(s) = \sum_{\text{tile \ in \ destination}} \text{Mahattan distance (tile, destination)}$$
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.

• **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 $g+h$
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.
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 $l_1$/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. A
- B. B
- C. 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. A
- B. B
- C. 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*:

S (Initial state) -> A (h=7) -> D (h=inf)
S (Initial state) -> B (h=4) -> E (h=inf)
S (Initial state) -> C (h=3) -> G (h=0)

Goal state
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)
C(8+3) D(4+inf) E(8+inf) G(9+0)
C(8+3) D(4+inf) E(8+inf)

CLOSED

G → B → S

Initial state

Goal state

h=0
h=inf
h=inf
h=8
h=7
h=4
h=3
h=5
h=1
h=9

Example for IDA*: Threshold = 8
Recap and Examples

Example for IDA*:

Threshold = 9

PREFIX | OPEN  
---|---
- | S(0+8)  
S | A(1+7) B(5+4)  
S A | B(5+4) H(2+2) D(4+4)  
S A H | B(5+4) D(4+4) F(6+1)  
S A H F | B(5+4) D(4+4)  
S A D | B(5+4)  
S B | G(9+0)  
S B G

Graph showing the IDA* algorithm with threshold = 9.
Recap and Examples

Example for Beam Search: $k=2$

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

Initial state

Goal state
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 and Fred Sala (University of Wisconsin-Madison).