CS 540 Introduction to Artificial Intelligence

Search II: Informed Search

Fred Sala
University of Wisconsin-Madison

April 13, 2021
Announcements

• **Homeworks:**
  – HW 8 Due, HW9 being released.

• **Grades:** Midterm & HW4, HW 7 out. HW6 soon

• **Class roadmap:**

<table>
<thead>
<tr>
<th>Tuesday, April 13</th>
<th>Search II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thursday, April 15</td>
<td>Genetic Algorithms</td>
</tr>
<tr>
<td>Tuesday, April 20</td>
<td>Introduction to RL</td>
</tr>
<tr>
<td>Thursday, April 22</td>
<td>RL and Search Summary</td>
</tr>
<tr>
<td>Tuesday, April 27</td>
<td>AI in the Real World</td>
</tr>
</tbody>
</table>
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 weight 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 (recall game heuristic)
Informed Search

Informed search. Know:

• All uninformed search properties, plus
• Heuristic $h(s)$ from $s$ to goal (recall game heuristic)

• Like in games, 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”
  
  ![Diagram showing start, s, and goal nodes with g(s) and h(s) labels]

• 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
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.

- 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.
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*:

Example for A*:

Example for A*:

Example for A*:

Example for A*:

Example for A*:

Example for A*:

Example for A*:

Example for A*:

Example for A*:
Example for A*: 

**OPEN**
- S(0+8)
- A(1+7) B(5+4) C(8+3)

**CLOSED**
- -
- S(0+8)
- 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) S(0+8) A(1+7) B(5+4)
- C(8+3) D(4+inf) E(8+inf) S(0+8) A(1+7) B(5+4) G(9+0)

G → B → S
Example for IDA*:

Threshold = 8

PREFIX
- S
S A
S A H
S A H F
S A D

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

Initial state

Goal state
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

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

Prefix

- S
  S A
  S A H
  S A H F
  S A D
  S B
  S B G
Recap and Examples

**Example for Beam Search: \( k=2 \)**

<table>
<thead>
<tr>
<th>CURRENT</th>
<th>OPEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>S(0+8)</td>
</tr>
<tr>
<td>S</td>
<td>A(1+7) B(5+4)</td>
</tr>
<tr>
<td>A</td>
<td>H(2+2) D(4+4)</td>
</tr>
<tr>
<td>H</td>
<td>D(4+4) F(6+1)</td>
</tr>
<tr>
<td>F</td>
<td>D(4+4) G(10+0)</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
</tbody>
</table>

**Initial state**

![Graph showing Beam Search example with nodes and edges labeled with distances]

**Goal state**

- Initial state: \( h=8 \)
- Goal state: \( h=0 \)
- Distance labels: 1, 2, 3, 4, 5, 7, 9, 8
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 (University of Wisconsin).