Algorithms for Data Structures: Heuristic Search

Phillip Smith
20/11/13
Aims

• Once you’ve understood this you should be able to:
  • Explain the idea of a heuristic
  • Devise simple heuristics
  • Carry out best-first search, hill climbing and A* search
Heuristics

• So far we’ve looked at strategies for searching when we know very little about the problem

• Heuristics are rules of thumb:
  • Approximate
  • Quick to compute
  • Not guaranteed to work

• Informed (or heuristic) search uses rules of thumb to guide search and cut down the amount of work we have to do

• Heuristics are used throughout AI

• We will go through a heuristic estimate of the distance (or cost) between the current state and the goal state
Example Heuristic: Estimate of distance to go

• Consider the 8-puzzle tile sliding game:

• Goal state:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

• Which of the following is closer to the goal?
• One heuristic is to count the number of tiles out of place:

  \[ \hat{H}(A) = 4 \quad \hat{H}(B) = 2 \]

  \[ H(A) = 5 \quad H(B) = 2 \]
Hill Climbing

• How can we use our heuristic estimate of the distance to a goal state?

• In steepest-ascent hill climbing we generate the children of the current state

• We calculate the heuristic value of each

• Then select the one with the ‘best’ heuristic value

• Repeat until you can’t improve
Hill Climbing Example
Hill Climbing Gets Stuck

- Often hill climbing will reach a point where it can’t improve further:

\[
\begin{array}{ccc}
2 & 8 & 3 \\
1 & 4 \\
7 & 6 & 5 \\
\end{array}
\]

\[\hat{H} = 3\]

- This is an example of a plateau

- There is no efficient way to cross a large plateau if there is (by definition) no information to guide the search
Hill Climbing Gets Stuck

• Hill climbing can also get stuck on local maxima (or minima if we’re doing gradient descent)

• We can see this in the 8-puzzle example if we change the heuristic:

• Heuristic 2 $h_2$: for each tile add its vertical and horizontal displacement from its desired position. Sum these values across all the tiles.
Hill Climbing Gets Stuck

h₂ = 1 + 1 + 3 = 5

h₂ = 1 + 1 + 3 = 5

h₂ = 6

h₂ = 6

h₂ = 6

h₂ = 6
Best-First Search

• Remember the complete search tree you’ve explored so far (as in breadth-first search)

• But use $\hat{H}$ (evaluation function) to decide which leaf node to expand next, instead of path cost

• A venerable, but inaccurate name

  • If we really could choose the best node to expand, then it wouldn’t really be a search at all

  • All we can do is choose the ‘best’ according to an evaluation function
Best-First Search

\[ \hat{H} = 4 \]

\[ \hat{H} = 5 \]

\[ \hat{H} = 3 \]

\[ \hat{H} = 5 \]
Best-First Search

We can still wander aimlessly however...
A* Search

- To obtain better searching we need to take into account the cost of the path so far

- $g(A) = \text{cost (length) of the path from the root node to node } A$

- $\hat{H}(A) = \text{heuristic estimate of the cost (length) of the path from node } A \text{ to a goal state}$

- $f(A) = g(A) + \hat{H}(A)$

- $f(A)$ is an estimate of the total cost of the path through $A$ that starts at the root node and ends in the goal node
A* Search: Example

```
2 8 3
1 6 4
7 5
f = 0 + 4
f = 1 + 5
f = 1 + 3
f = 1 + 5
f = 2 + 3
f = 2 + 3
f = 2 + 4
f = 3 + 3
f = 3 + 4
f = 3 + 2
f = 3 + 4
```
A* Search: Example

GOAL, woohoo!
Inventing Heuristics

• Ĥ and h₂ are fairly good heuristics, but how do we invent one which is possibly better?
• Is it possible for a machine to create such a heuristic?
• Composite heuristic
  • Uses whichever currently defined heuristic returns the best result
• Statistical information:
  • Run our search 100 times and examine patterns
  • When h₂( n ) = 14, it turns out that 90% of the time the real distance to the goal is 18. We can therefore use 18 as the real value when 14 is returned
Search: the story so far...

• We’ve seen:
  • depth-first ( depth-limited, DFID )
  • breadth-first
  • best-first with $\hat{H}$
  • best-first with $f$ ( A* search)

• We can unify all these ( mostly ) into a single framework

• We can do this using the idea of an agenda
Agenda Based Search

- In all our algorithms we have to choose which leaf node in the search tree to expand.

  - OPEN = [D E C] - nodes to expand (leaf)
  - CLOSED = [A B] - nodes already expanded (internal)

We can split the nodes into two lists:
Agenda Based Search

• Suppose we reorder the nodes in OPEN according to some criterion?

  • e.g reorder by depth of node in tree
    • deepest first (depth-first search)
      • OPEN = [ D E C ]
    • shallowest first (breadth-first search)
      • OPEN = [ C D E ]
Agenda Based Search

- We then:
  - Expand the first node in OPEN
  - put it in CLOSED
  - put its children in OPEN
  - reorder OPEN
  - (NB to obtain depth-first search we also need to delete nodes from CLOSED when we backtrack)
Agenda Based Search

• We can also implement best-first search in this way

• If we reorder OPEN by $\hat{H}$ then we have best-first search as described in the last lecture

• This is actually called greedy search

• Best-first search using $\hat{H}$ to reorder OPEN = greedy search

• Best-first search using $g$ to reorder OPEN = uniform cost search

• Best-first search using $f = g + \hat{H}$ to reorder OPEN = A* search

• NB if $g$ is just the depth of the node in the tree then uniform cost search = breadth-first search
Agenda Based Search:
Example

\[ f = 0 + 4 \]
\[ f = 1 + 5 \]
\[ f = 1 + 3 \]
\[ f = 3 + 4 \]

\[ f = 2 + 3 \]
\[ f = 3 + 3 \]
Agenda Based Search: Example

- OPEN: (B D F G H I)
- g: (1 1 2 2 3 3)
- \( \hat{H} \): (5 5 3 4 3 4)
Agenda Based Search: Example

\[ f = 0 + 4 \]

\[ f = 1 + 5 \]

\[ f = 1 + 3 \]

\[ f = 1 + 5 \]

\[ f = 2 + 3 \]

\[ f = 2 + 3 \]

\[ f = 3 + 4 \]

\[ f = 3 + 3 \]

\[ f = 3 + 4 \]

OPEN \( ( B \ D \ F \ G \ H \ I ) \)

g \( ( 1 \ 1 \ 2 \ 2 \ 3 \ 3 ) \)

\( \hat{H} \) \( ( 5 \ 5 \ 3 \ 4 \ 3 \ 4 ) \)

Uniform cost: reorder by g
Agenda Based Search: Example

Greedy search: reorder by $\hat{H}$
Agenda Based Search: Example

A* search: reorder by $f = g + \hat{H}$
Summary

- Heuristic evaluations of cost to reach goal
- Hill climbing
- Best-first search
- A* Search
- Agenda-based search