Inf2D 05: Informed Search and Exploration for Agents

Stefano Albrecht

School of Informatics, University of Edinburgh

25/01/18

Slide Credits: Jacques Fleuriot, Michael Rovatsos, Michael Herrmann
Outline

- Best-first search
- Greedy best-first search
- $A^*$ search
- Heuristics
- Admissibility
A search strategy is defined by picking the order of node expansion from the frontier.

```plaintext
function TREE-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        expand the chosen node, adding the resulting nodes to the frontier
```

Stefano Albrecht
Inf2D 05: Informed Search and Exploration for Agents
Best-first search

- An instance of general TREE-SEARCH or GRAPH-SEARCH
- **Idea:** use an evaluation function $f(n)$ for each node $n$
  - estimate of "desirability"
    - Expand most desirable unexpanded node, usually the node with the lowest evaluation
- **Implementation:**
  Order the nodes in frontier in decreasing order of desirability
- **Special cases:**
  - Greedy best-first search
  - $A^*$ search
Romania with step costs in km

<table>
<thead>
<tr>
<th>City</th>
<th>Distance to Bucharest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arad</td>
<td>366</td>
</tr>
<tr>
<td>Bucharest</td>
<td>0</td>
</tr>
<tr>
<td>Craiova</td>
<td>160</td>
</tr>
<tr>
<td>Dobrogea</td>
<td>242</td>
</tr>
<tr>
<td>Eforie</td>
<td>161</td>
</tr>
<tr>
<td>Fagaras</td>
<td>176</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
</tr>
<tr>
<td>Hirsorn</td>
<td>151</td>
</tr>
<tr>
<td>Iasi</td>
<td>226</td>
</tr>
<tr>
<td>Lugoj</td>
<td>244</td>
</tr>
<tr>
<td>Mehadia</td>
<td>241</td>
</tr>
<tr>
<td>Neamț</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Pitești</td>
<td>104</td>
</tr>
<tr>
<td>Rimnicu Vâlcea</td>
<td>193</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
</tr>
<tr>
<td>Timișoara</td>
<td>329</td>
</tr>
<tr>
<td>Urziceni</td>
<td>80</td>
</tr>
<tr>
<td>Vâlcea</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
</tr>
</tbody>
</table>
Greedy best-first search

- Evaluation function $f(n) = h(n)$ (heuristic)
- $h(n)$: estimated cost of cheapest path from state at node $n$ to a goal state
  - e.g., $h_{SLD}(n)$: straight-line distance from $n$ to goal (Bucharest)
  - Greedy best-first search expands the node that appears to be closest to goal
What is a heuristic?

- From the greek word “heuriskein” meaning “to discover” or “to find”
- A heuristic is any method that is believed or practically proven to be useful for the solution of a given problem, although there is no guarantee that it will always work or lead to an optimal solution.
- Here we will use heuristics to guide tree search. This may not change the worst case complexity of the algorithm, but can help in the average case.
- We will introduce conditions (admissibility, consistency, see below) in order to identify good heuristics, i.e. those which actually lead to an improvement over uninformed search.
- See also: https://en.wikipedia.org/wiki/Heuristic
Greedy best-first search example

[Diagram of a search tree with a node labeled Arad 366]
Greedy best-first search example
Greedy best-first search example
Greedy best-first search example
Properties of greedy best-first search

- **Complete?** No – can get stuck in loops
  - Graph search version is complete in finite space, but not in infinite ones
- **Time?** $O(b^m)$ for tree version, but a good heuristic can give dramatic improvement
- **Space?** $O(b^m)$ – keeps all nodes in memory
- **Optimal?** No
**A* search**

- **Idea:** avoid expanding paths that are already expensive
- **Evaluation function** $f(n) = g(n) + h(n)$
  - $g(n)$: cost so far to reach $n$
  - $h(n)$: estimated cost from $n$ to goal
  - $f(n)$: estimated total cost of path through $n$ to goal
- $A^*$ is both complete and optimal if $h(n)$ satisfies certain conditions
A* search example

\[ 366 = 0 + 366 \]
A* search example
A* search example
A* search example

Stefano Albrecht
Inf2D 05: Informed Search and Exploration for Agents
A* search example
A* search example
A heuristic \( h(n) \) is admissible if for every node \( n \),
\[ h(n) \leq h^*(n), \]
where \( h^*(n) \) is the true cost to reach the goal state from \( n \).

An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic.

Thus, \( f(n) = g(n) + h(n) \) never overestimates the true cost of a solution.

Example: \( h_{SLD}(n) \) (never overestimates the actual road distance).

**Theorem:** If \( h(n) \) is admissible, \( A^* \) using TREE-SEARCH is optimal.
Suppose some suboptimal goal $G_2$ has been generated and is in the frontier. Let $n$ be an unexpanded node in the frontier such that $n$ is on a shortest path to an optimal goal $G$.

- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $f(G) = g(G)$ since $h(G) = 0$
- $g(G_2) > g(G)$ since $G_2$ is suboptimal
- $f(G_2) > f(G)$ from above
Suppose some suboptimal goal $G_2$ has been generated and is in the frontier. Let $n$ be an unexpanded node in the frontier such that $n$ is on a shortest path to an optimal goal $G$.

- $f(G) < f(G_2)$ from above ($G_2$ is suboptimal)
- $h(n) \leq h^*(n)$ since $h$ is admissible
- $g(n) + h(n) \leq g(n) + h^*(n) = f(G)$
- $f(n) \leq f(G)$

Hence $f(n) < f(G_2) \Rightarrow A^*$ will never select $G_2$ for expansion.
A heuristic is **consistent** if for every node $n$, every successor $n'$ of $n$ generated by any action $a$,

$$h(n) \leq c(n, a, n') + h(n')$$

If $h$ is consistent, we have

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

$$= g(n) + c(n, a, n') + h(n')$$

$$\geq g(n) + h(n)$$

$$\geq f(n)$$

i.e., $f(n)$ is non-decreasing along any path.

**Theorem:**

If $h(n)$ is consistent, $A^*$ using GRAPH-SEARCH is optimal.
Optimality of $A^*$

- $A^*$ expands nodes in order of increasing $f$ value
- Gradually adds "$f$-contours" of nodes
- Contour $i$ has all nodes with $f = f_i$, where $f_i < f_{i+1}$
Properties of A* 

- **Complete?** Yes (unless there are infinitely many nodes with $f \leq f(G)$)
- **Time?** Exponential
- **Space?** Keeps all nodes in memory
- **Optimal?** Yes
Admissible heuristics

Example:

- for the 8-puzzle:
  - $h_1(n)$: number of misplaced tiles
  - $h_2(n)$: total Manhattan distance

(i.e., no. of squares from desired location of each tile)

Exercise: Calculate these two values:

- $h_1(S) = ?$
- $h_2(S) = ?$
If $h_2(n) \geq h_1(n)$ for all $n$ (both admissible) then

- $h_2$ dominates $h_1$
- $h_2$ is better for search

**Typical search costs** (average number of nodes expanded):

- $d = 12$  \hspace{1cm} IDS = 3,644,035 nodes
  \hspace{1cm} $A^*(h_1) = 227$ nodes
  \hspace{1cm} $A^*(h_2) = 73$ nodes

- $d = 24$  \hspace{1cm} IDS = too many nodes
  \hspace{1cm} $A^*(h_1) = 39,135$ nodes
  \hspace{1cm} $A^*(h_2) = 1,641$ nodes
Relaxed problems

- A problem with fewer restrictions on the actions is called a **relaxed problem**
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere,
  - then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square,
  - then $h_2(n)$ gives the shortest solution
- **Can use relaxation to automatically generate admissible heuristics**
Smart search based on **heuristic scores**.

- Best-first search
- Greedy best-first search
- $A^*$ search
- Admissible heuristics and optimality.