Informed search algorithms

Chapter 4, Sections 1–2
Outline

♦ Best-first search
♦ A* search
♦ Heuristics
function TREE-SEARCH(problem, frontier) returns a solution, or failure
    frontier ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), frontier)
    loop do
        if frontier is empty then return failure
        node ← REMOVE(frontier)
        if GOAL-TEST[problem] applied to STATE(node) succeeds return node
        frontier ← INSERTALL(EXPAND(node, problem), frontier)

A strategy is defined by picking the order of node expansion
Best-first search

**Idea:** use an evaluation function for each node
- estimate of “desirability”

⇒ Expand most desirable unexpanded node

**Implementation:**
*frontier* is a priority queue sorted in decreasing order of desirability

Special cases:
- greedy search
- A* search
<table>
<thead>
<tr>
<th>Location</th>
<th>Straight-line 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>178</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
</tr>
<tr>
<td>Hirsova</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>Neamţ</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Pitesti</td>
<td>98</td>
</tr>
<tr>
<td>Rimnicu Vielcea</td>
<td>193</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
</tr>
<tr>
<td>Timişoara</td>
<td>329</td>
</tr>
<tr>
<td>Urziceni</td>
<td>80</td>
</tr>
<tr>
<td>Vaslui</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
</tr>
</tbody>
</table>

Chapter 4, Sections 1–2
Greedy search

Evaluation function $h(n)$ (heuristic)

$= \text{estimate of cost from } n \text{ to the closest goal}$

E.g., $h_{SLD}(n) = \text{straight-line distance from } n \text{ to Bucharest}$

Greedy search expands the node that appears to be closest to goal
Greedy search example

Arad
366
Greedy search example

Chapter 4, Sections 1–2
Greedy search example
Greedy search example

- **Arad**
  - **Sibiu** (366)
    - **Bucharest** (253)
    - **Fagaras**
      - **Sibiu** (253)
    - **Oradea** (380)
      - **Bucharest** (0)
    - **Rimnicu Vilcea** (193)
  - **Timisoara** (329)
  - **Zerind** (374)
Properties of greedy search

Complete??

Time??

Space??

Optimal??
Properties of greedy search

**Complete??** No—can get stuck in loops, e.g.,
Iasi → Neamt → Iasi → Neamt →
Complete in finite space with repeated-state checking

**Time??** $O(b^m)$, 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 to goal from \( n \)
- \( f(n) \) = estimated total cost of path through \( n \) to goal

A* search uses an admissible heuristic

i.e., \( h(n) \leq h^*(n) \) where \( h^*(n) \) is the true cost from \( n \).

(Also require \( h(n) \geq 0 \), so \( h(G) = 0 \) for any goal \( G \).)

E.g., \( h_{SLD}(n) \) never overestimates the actual road distance

Theorem: A* search is optimal
A* search example

Arad

366 = 0 + 366
A* search example

- Sibiu: 393 = 140 + 253
- Timisoara: 447 = 118 + 329
- Zerind: 449 = 75 + 374
A* search example

- Arad
  - Sibiu
    - Arad
    - Fagaras
    - Oradea
    - Rimnicu Vilcea
  - Timisoara
    - 447 = 118 + 329
  - Zerind
    - 449 = 75 + 374

- Arad
  - 646 = 280 + 366
  - 415 = 239 + 176
  - 671 = 291 + 380
  - 413 = 220 + 193
A* search example

**Chapter 4, Sections 1–2**
A* search example
A* search example
Conditions for optimality of $A^*$

**admissible**: $h(n)$ is never overestimates the cost to reach the goal. (optimistic) exert straight-line distance.

**consistent (or monotonic)**: (stronger condition) $h(n)$ is required for $A^*$ in graph search framework.

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

A heuristic $h(n)$ is consistent if, for every node $n$ and every successor $n'$ for $n$ generated by any action $a$, the estimated cost of reaching the goal from $n$ is no greater than the step cost of getting to $n'$ plus the estimated cost of reaching the goal from $n'$.
Admissible heuristics

E.g., for the 8-puzzle:

\[ h_1(n) = \text{number of misplaced tiles} \]
\[ h_2(n) = \text{total Manhattan distance} \]

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

\[
\begin{align*}
    h_1(S) &= ?? \\
    h_2(S) &= ??
\end{align*}
\]
Admissible heuristics

E.g., for the 8-puzzle:

\[ h_1(n) = \text{number of misplaced tiles} \]
\[ h_2(n) = \text{total Manhattan distance} \]

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

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

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

Start State

Goal State

\[ h_1(S) = 6 \]
\[ h_2(S) = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1 = 14 \]
If $h_2(n) \geq h_1(n)$ for all $n$ (both admissible) then $h_2$ dominates $h_1$ and is better for search.

Typical search costs:

\[ d = 14 \quad \text{IDS} = 3,473,941 \text{ nodes} \]
\[ A^*(h_1) = 539 \text{ nodes} \]
\[ A^*(h_2) = 113 \text{ nodes} \]

\[ d = 24 \quad \text{IDS} \approx 54,000,000,000 \text{ nodes} \]
\[ A^*(h_1) = 39,135 \text{ nodes} \]
\[ A^*(h_2) = 1,641 \text{ nodes} \]

Given any admissible heuristics $h_a, h_b,$

\[ h(n) = \max(h_a(n), h_b(n)) \]

is also admissible and dominates $h_a, h_b$
Admissible heuristics can be derived from the exact solution cost of a relaxed version of the 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.

Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem.
Relaxed problems contd.

Well-known example: travelling salesperson problem (TSP)
Find the shortest tour visiting all cities exactly once

Minimum spanning tree can be computed in $O(n^2)$
and is a lower bound on the shortest (open) tour
The tree-search version of A* is optimal if \( h(n) \) is admissible.

Suppose some suboptimal goal \( G_2 \) has been generated and is in the queue. Let \( n \) be an unexpanded node on a shortest path to an optimal goal \( G \).

\[
f(G_2) = g(G_2) \quad \text{since } h(G_2) = 0
\]
\[
> g(G) \quad \text{since } G_2 \text{ is suboptimal}
\]
\[
\geq f(n) \quad \text{since } h \text{ is admissible}
\]
Since $f(G_2) > f(n)$, A* will never select $G_2$ for expansion.
Optimality of A* - graph search

The graph-search version of A* is optimal if $h(n)$ is consistent.

**Lemma:** A* using graph-search expands in nondecreasing order of $f(n)$.

1. A heuristic is consistent if

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

If $h$ is consistent, we have

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

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

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

$$ = f(n) $$

I.e., $f(n)$ is nondecreasing along any path.
2. Whenever $A^*$ selects a node $n$ for expansion, the optimal path to that node has been found.

(Proof by contradiction) - If optimal path has not been found when $A^*$ selects a node, there would have to be another frontier node $n'$ on the optimal path from the start node to $n$, by the graph separation property of graph-search; because, $f$ is nondecreasing along any path, $n'$ would have lower $f$-cost than $n$ and would have been selected first.
From 1. and 2. we can see that A* gradually adds “f-contours” of nodes (cf. breadth-first adds layers)
Contour \( i \) has all nodes with \( f = f_i \), where \( f_i < f_{i+1} \)
Properties of $A^*$

Complete
Time
Space
Optimal
Properties of $A^*$

**Complete** Yes, unless there are infinitely many nodes with $f \leq f(G)$

**Time** Exponential in [relative error in $h \times$ length of soln.]

**Space** Keeps all nodes in memory

**Optimal** Yes—cannot expand $f_{i+1}$ until $f_i$ is finished

$A^*$ expands all nodes with $f(n) < C^*$

$A^*$ expands some nodes with $f(n) = C^*$

$A^*$ expands no nodes with $f(n) > C^*$
Heuristic functions estimate costs of shortest paths

Good heuristics can dramatically reduce search cost

Greedy best-first search expands lowest $h$
  – incomplete and not always optimal

$A^*$ search expands lowest $g + h$
  – complete and optimal
  – also optimally efficient (up to tie-breaks, for forward search)

Admissible heuristics can be derived from exact solution of relaxed problems