**Informed search algorithms**

Chapter 4, Sections 1–2

### Outline
- Best-first search
- A* search
- Heuristics

---

**Review: Tree search**

<table>
<thead>
<tr>
<th>function</th>
<th>Tree-Search(problem, fringe)</th>
<th>returns</th>
<th>a solution, or failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>fringe</td>
<td>Insert(Make-Node[Initial-State[problem]], fringe)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

 loop do

  if fringe is empty then return failure

  node ← Remove-Front(fringe)

  if Goal-Test(problem) applied to State(node) succeeds return node

  fringe ← InsertAll(Expand(node, problem), fringe)

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:**
fringe is a queue sorted in decreasing order of desirability

**Special cases:**
- greedy search
- A* search

---

**Romania with step costs in km**

Straight-line distance to Bucharest

<table>
<thead>
<tr>
<th>City</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arad</td>
<td>266</td>
</tr>
<tr>
<td>Bucharest</td>
<td>0</td>
</tr>
<tr>
<td>Craiova</td>
<td>200</td>
</tr>
<tr>
<td>Dolj</td>
<td>242</td>
</tr>
<tr>
<td>Elcica</td>
<td>284</td>
</tr>
<tr>
<td>Fagaras</td>
<td>178</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
</tr>
<tr>
<td>Hlava</td>
<td>251</td>
</tr>
<tr>
<td>Iasi</td>
<td>226</td>
</tr>
<tr>
<td>Lugoj</td>
<td>244</td>
</tr>
<tr>
<td>Medias</td>
<td>241</td>
</tr>
<tr>
<td>Neamt</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>340</td>
</tr>
<tr>
<td>Pitesti</td>
<td>98</td>
</tr>
<tr>
<td>Romanesti</td>
<td>105</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
</tr>
<tr>
<td>Slobozia</td>
<td>329</td>
</tr>
<tr>
<td>Suceava</td>
<td>226</td>
</tr>
<tr>
<td>Timișoara</td>
<td>138</td>
</tr>
<tr>
<td>Timişoara</td>
<td>146</td>
</tr>
<tr>
<td>Timişoara</td>
<td>85</td>
</tr>
<tr>
<td>Vaslui</td>
<td>234</td>
</tr>
<tr>
<td>Vatra</td>
<td>80</td>
</tr>
<tr>
<td>Vâlcea</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>214</td>
</tr>
</tbody>
</table>
Greedy search example

Complete??

Properties of greedy search

Complete??
No—can get stuck in loops, e.g., with Oradea as goal.

Iasi — Neamt — Iasi — Neamt —

Complete in finite space with repeated-state checking.

Time??
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??

\[ f(n) = g(n) + h(n) \]

\( g(n) \) = 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
Optimality of A∗ (standard proof)

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 $G_1$.

$$f(G_2) = g(G_2) \quad \text{since } h(G_2) = 0$$
$$> g(G_1) \quad \text{since } G_2 \text{ is suboptimal}$$
$$\geq f(n) \quad \text{since } h \text{ is admissible}$$

Since $f(G_2) > f(n)$, A∗ will never select $G_2$ for expansion.

Optimality of A∗ (more useful)

Lemma: A∗ expands nodes in order of increasing $f$ value

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

Time?? Exponential in $[\text{relative error in } h \times \text{length of soln.}]$

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 $f(n) > C^*$

Proof of lemma: Consistency

A heuristic is consistent if

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

If $h$ is consistent, we have

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

I.e., $f(n)$ is nondecreasing along any path.
Admissible heuristics

E.g., for the 8-puzzle:

\begin{align*}
h_1(n) &= \text{number of misplaced tiles} \\
h_2(n) &= \text{total Manhattan distance} \\
&= \text{(i.e., no. of squares from desired location of each tile)}
\end{align*}

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

\begin{align*}
h_1(S) &= \text{??} \\
h_2(S) &= \text{??}
\end{align*}

Relaxed problems

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.

Dominance

If \( h_2(n) \geq h_1(n) \) for all \( n \) (both admissible) then \( h_2 \) dominates \( h_1 \) and is usually better for search.

Typical search costs:

- \( d = 14 \) : IDS = 3,473,941 nodes
- \( \Delta^*(h_1) = 539 \) nodes
- \( \Delta^*(h_2) = 113 \) nodes
- \( d = 21 \) : IDS \( \approx 54,000,000,000 \) nodes
- \( \Delta^*(h_1) = 39,135 \) nodes
- \( \Delta^*(h_2) = 1,641 \) nodes

Given any admissible heuristics \( h_a, h_b \),

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

is also admissible and dominates \( h_a, h_b \).

Summary

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

\( \Delta^* \) 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.