Informed search algorithms

Chapter 4, Sections 1–2, 4
Outline

◊ Best-first search
◊ A* search
◊ Heuristics
◊ Hill-climbing
◊ Simulated annealing
function TREE-SEARCH(problem, fringe) returns a solution, or failure
fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
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 ← INSERT-ALL(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

- Arad: 366
- Bucharest: 0
- Craiova: 160
- Dobretta: 242
- Eforie: 161
- Fagaras: 178
- Giurgiu: 77
- Hirsova: 151
- Iasi: 226
- Lugoj: 244
- Mehadia: 241
- Neamt: 234
- Oradea: 380
- Pitesti: 98
- Rimnicu Vilcea: 193
- Sibiu: 253
- Timisoara: 329
- Urziceni: 80
- Vaslui: 199
- Zerind: 374
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 \textit{appears} to be closest to goal
Greedy search example
Greedy search example

Diagram:
- Arad
  - Sibiu: 253
  - Timisoara: 329
  - Zerind: 374
Greedy search example

- Arad
  - Fagaras
    - Rimnicu Vilcea
  - Sibiu
    - Oradea
    - Timisoara
  - Zerind
    - Arad
    - Oradea
    - Rimnicu Vilcea
Greedy search example

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

Complete?? No—can get stuck in loops, e.g.,
lasi $\rightarrow$ Neamt $\rightarrow$ lasi $\rightarrow$ Neamt $\rightarrow$
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
A* search example

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

Costs:
- Arad: 646 = 280 + 366
- Fagaras: 415 = 239 + 176
- Oradea: 671 = 291 + 380
- Rimnicu Vilcea: 526 = 366 + 160
- Craiova: 417 = 317 + 100
- Pitesti: 553 = 300 + 253
A* search example

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

- Arad
  - Sibiu
    646 = 280 + 366
  - Bucharest
    450 = 450 + 0
  - Craiova
    526 = 366 + 160
  - Pitesti
    417 = 317 + 100
  - Sibiu
    553 = 300 + 253
A* search example
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 \( G_1 \).

\[
\begin{align*}
  f(G_2) &= g(G_2) \quad \text{since } h(G_2) = 0 \\
  &> g(G_1) \quad \text{since } G_2 \text{ is suboptimal} \\
  &\geq f(n) \quad \text{since } h \text{ is admissible}
\end{align*}
\]

Since \( f(G_2) > f(n) \), A* will never select \( G_2 \) for expansion.
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

\[
\begin{align*}
  f(n') &= g(n') + h(n') \\
  &= g(n) + c(n, a, n') + h(n') \\
  &\geq g(n) + h(n) \\
  &= f(n)
\end{align*}
\]

i.e., \( f(n) \) is nondecreasing along any path.
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)

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

Start State

Goal State

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

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

Start State

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

Goal State

\[
h_1(S) = \text{?? 7} \\
h_2(S) = \text{?? 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1 = 14}
\]
Dominance

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:

\[
d = 14 \quad \text{IDS} = 3,473,941 \text{ nodes} \\
\quad A^*(h_1) = 539 \text{ nodes} \\
\quad A^*(h_2) = 113 \text{ nodes} \\
d = 24 \quad \text{IDS} \approx 54,000,000,000 \text{ nodes} \\
\quad A^*(h_1) = 39,135 \text{ nodes} \\
\quad A^*(h_2) = 1,641 \text{ nodes}
\]
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.
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
Hill-climbing (or gradient ascent/descent)

“Like climbing Everest in thick fog with amnesia”

function HILL-CLIMBING(problem) returns a state that is a local maximum
inputs: problem, a problem
local variables: current, a node
neighbor, a node

current ← MAKE-NODE(INITIAL-STATE[problem])
loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] < VALUE[current] then return STATE[current]
    current ← neighbor
end
Hill-climbing contd.

Problem: depending on initial state, can get stuck on local maxima

In continuous spaces, problems w/ choosing step size, slow convergence
Simulated annealing

Idea: escape local maxima by allowing some “bad” moves

but gradually decrease their size and frequency

function SIMULATED-ANNEALING(problem, schedule) returns a solution state

inputs: problem, a problem
          schedule, a mapping from time to “temperature”

local variables: current, a node
                  next, a node
                  T, a “temperature” controlling prob. of downward steps

current ← MAKE-NODE(INITIAL-STATE[problem])
for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE[next] - VALUE[current]
    if ΔE > 0 then current ← next
    else current ← next only with probability $e^{ΔE/T}$