Chapter 3 Solving Problems by Searching
3.5 – 3.6 Informed (heuristic) search strategies

CS4811 - Artificial Intelligence

Nilufer Onder
Department of Computer Science
Michigan Technological University
Outline

Best-first search
  Greedy search
  A* search

Heuristics

(Iterative deepening A* search)
Best-first search

- Remember that the **frontier** contains the unexpanded nodes
- Idea: use an **evaluation function** for each node  
  (the evaluation function is an estimate of “desirability”)  
- Expand the most desirable unexpanded node
- Implementation:
  Frontier is a queue sorted in decreasing order of desirability
- Special cases:
  - Greedy search
  - A* search
Romania with step costs in km
Greedy search

- Evaluation function
  \[ h(n) = \text{estimate of cost from } n \text{ to the closest goal} \]
  h is the \textit{heuristic} function

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

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

Arad
After expanding Arad
After expanding Sibiu
After expanding Fagaras

- Bucharest
- Sibiu
- Fagaras
- Oradea
- Rimnicu V. V.
- Sibiu
- Arad
- Timisoara
- Zerind
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 every node 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
  - if $h$ is an admissible heuristic then $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from $n$.
  - Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal $G$.
  - An admissible heuristic is allowed to underestimate, but can never overestimate cost.
  - E.g., $h_{SLD}(n)$ never overestimates the actual road distance.
A* search example

Arad

366 = 0 + 366
After expanding Arad

Sibiu
393 = 140 + 253

Timisoara
447 = 118 + 329

Zerind
449 = 75 + 374
After expanding Sibiu
After expanding Rimnicu Vilcea

Arad | Fagaras | Oradea | Rimnicu V. | Sibiu | Timisoara | Zerind
646=280+366 | 415=239+176 | 671=291+380 | 447=118+329 | 526=366+160 | 449=75+374
417=317+100 | 553=300+253

Sibiu

Arad

Fagaras

Oradea

Rimnicu V.

Craiova

Pitesti

Sibiu
After expanding Fagaras

- Bucharest
- Sibiu
- Fagaras
- Oradea
- Rimnicu V.

- Arad
- Timisoara
- Zerind

646 = 280 + 366
591 = 338 + 253
450 = 450 + 0
671 = 291 + 380
526 = 366 + 160
553 = 300 + 253
447 = 118 + 329
449 = 75 + 374
417 = 317 + 100
After expanding Pitesti

Arad

Sibiu

Fagaras

Oradea

Rimnicu V.

Bucharest

Timisoara

Zerind

447=118+329
449=75+374

671=291+380
526=366+160 553=300+253

418=418+0 615=455+160 607=414+193
447=118+329 449=75+374

591=338+253
450=450+0

553=300+253

418=418+0
615=455+160
607=414+193

450=450+0
Optimality of A*

**Theorem:** A* search is optimal.

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 \( G_1 \).
Proof for the optimality of A*:

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

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

- **Complete:** Yes, unless there are infinitely many nodes with $f \leq f(G)$
- **Time:** Exponential in $(\text{relative error in } h \times \text{length of solution})$
- **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 $f(n) > C^*$
Admissible heuristics

E.g., 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)

$h_1(S) = ??$

$h_2(S) = ??$
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)

\[ h_1(S) = 8 \]
\[ h_2(S) = 3+1+2+2+3+2+2+3 = 18 \]
Dominance

A “better” heuristic is one that minimizes the effective branching factor, $b^*$.

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:
- $d = 12 \quad$ IDS $= 3,644,035$ nodes \quad $b^* = 2.78$
  \quad $A^*(h_1) = 539$ nodes \quad $b^* = 1.42$
  \quad $A^*(h_2) = 113$ nodes \quad $b^* = 1.24$
- $d = 24 \quad$ IDS $\approx 54,000,000,000$ nodes
  \quad $A^*(h_1) = 39,135$ nodes \quad $b^* = 1.48$
  \quad $A^*(h_2) = 1,641$ nodes \quad $b^* = 1.26$
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.
Iterative Deepening A* (IDA*)

- Idea: perform iterations of DFS. The cutoff is defined based on the $f$-cost rather than the depth of a node.
- Each iteration expands all nodes inside the contour for the current $f$-cost, peeping over the contour to find out where the contour lies.
Summary

- Heuristic search algorithms
- Finding good heuristics for a specific problem is an area of research
- Think about the time to compute the heuristic
Sources for the slides

- AIMA textbook (3rd edition)
- AIMA slides (http://aima.cs.berkeley.edu/)