Informed Search

Russell and Norvig chap. 3

Outline

- Informed: use problem-specific knowledge
- Add a sense of direction to search: work toward the goal
- Heuristic functions: a way to provide information to a search algorithm

What determines a search strategy

function TREE-SEARCH(problem) return a solution or failure
Initialize frontier using the initial state of problem
do
    if the frontier is empty then return failure
    choose leaf node from the frontier
    if node is a goal state then return solution
    else expand the node and add resulting nodes to the frontier
A search strategy is determined by the order in which nodes in the frontier are processed

Best-first search

- Informed search strategy: expand the node that appears best
- Factors going into determination of best:
  - Current cost of the solution path
  - Estimated distance to the nearest goal state
- Node is selected for expansion based on an evaluation function (f(n))
- Implementation:
  - Fringe is a queue sorted by value of f
  - Special cases: greedy search, A* search

Heuristics

Heuristic: “A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions”

- The heuristic function h(n) estimates cost of the cheapest path from node n to goal node.
- If n is a goal node h(n)=0

Greedy best-first search

- Expand node on the frontier closest to goal
- Determination of closest based upon the heuristic function h
Greedy search: An example

- Consider path planning between two cities
- Use the straight line distance heuristic, \( h_{SLD} \)
- The greedy solution is (A, C, D)
- The least cost solution is (A, B, D)

A* Search

- Order states by their total estimated cost
- Always select the node with the lowest value
- Total estimated cost:
  \[ f(n) = g(n) + h(n) \]
- \( g(n) \) the cost to reach \( n \)
- \( h(n) \) the estimated cost to the goal

A* Search

- Order states by their total estimated cost
- Always select the node with the lowest value
- Total estimated cost:
  \[ f(n) = g(n) + h(n) \]
- \( g(n) \) the cost to reach \( n \)
- \( h(n) \) the estimated cost to the goal

Repeated states

- Uninformed search:
  - Add to fringe only if state not already visited.
- A*:
  - If node represents state already visited, update cost according to lower total estimated cost.

Heuristic functions

- \( h_1 \) = the number of misplaced tiles
- \( h_2 \) = the sum of the distances of the tiles from their goal positions (manhattan distance)

Comparison of heuristics

Even very simple heuristics like \( h_1 \) and \( h_2 \) can significantly reduce the search cost:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Depth 10</th>
<th>Depth 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Deepening</td>
<td>47,127</td>
<td>3,473,941</td>
</tr>
<tr>
<td>A* with ( h_1 )</td>
<td>93</td>
<td>539</td>
</tr>
<tr>
<td>A* with ( h_2 )</td>
<td>39</td>
<td>113</td>
</tr>
</tbody>
</table>

A* in Romania

Goal: shortest route from Arad to Bucharest

Expand Arad and determine $f(n)$:
- $f(\text{Sibiu}) = c(\text{Arad, Sibiu}) + h(\text{Sibiu}) = 140 + 253 = 393$
- $f(\text{Timisoara}) = c(\text{Arad, Timisoara}) + h(\text{Timisoara}) = 118 + 329 = 447$
- $f(\text{Zerind}) = c(\text{Arad, Zerind}) + h(\text{Zerind}) = 75 + 374 = 449$
- Best choice is Sibiu

Expand Sibiu and determine $f(n)$:
- $f(\text{Arad}) = c(\text{Sibiu, Arad}) + h(\text{Arad}) = 280 + 366 = 646$
- $f(\text{Fagaras}) = c(\text{Sibiu, Fagaras}) + h(\text{Fagaras}) = 239 + 179 = 418$
- $f(\text{Oradea}) = c(\text{Sibiu, Oradea}) + h(\text{Oradea}) = 291 + 380 = 671$
- $f(\text{Rimnicu Vilcea}) = c(\text{Sibiu, Rimnicu Vilcea}) + h(\text{Rimnicu Vilcea}) = 220 + 192 = 413$
- Best choice is Rimnicu Vilcea

Expand Rimnicu Vilcea and determine $f(n)$:
- $f(\text{Craiova}) = c(\text{Rimnicu Vilcea, Craiova}) + h(\text{Craiova}) = 360 + 160 = 526$
- $f(\text{Pitesti}) = c(\text{Rimnicu Vilcea, Pitesti}) + h(\text{Pitesti}) = 317 + 100 = 417$
- $f(\text{Sibiu}) = c(\text{Rimnicu Vilcea, Sibiu}) + h(\text{Sibiu}) = 300 + 253 = 553$
- Best choice is Fagaras

A* example

Expand Fagaras and determine $f(n)$:
- $f(\text{Sibiu}) = c(\text{Fagaras, Sibiu}) + h(\text{Sibiu}) = 338 + 253 = 591$
- $f(\text{Bucharest}) = c(\text{Fagaras, Bucharest}) + h(\text{Bucharest}) = 450 + 0 = 450$
- Best choice is Pitesti!
A* in Romania

- Expand Pitesti and determine \( f(n) \)
  \( f(\text{Bucharest}) = c(\text{Pitesti, Bucharest}) + h(\text{Bucharest}) = 418 + 0 = 418 \)
- Best choice is Bucharest
- Note values along optimal path!!
- Is the solution optimal?

Admissible heuristics

- A heuristic is admissible if it never overestimates the cost to reach the goal (optimistic)
  - Formally:
    1. \( h(n) \leq h^*(n) \) where \( h^*(n) \) is the true cost from \( n \)
    2. \( h(n) = 0 \) so \( h(G) = 0 \) for any goal \( G \).
- Examples:
  - \( h_{SLD}(n) \) never overestimates the actual road distance
  - Heuristics for 8 puzzle

Consistency

- A heuristic is consistent if:
  \[ h(n) \leq c(n, a, n') + h(n') \]
- Given a consistent heuristic:
  \[ f(n') = g(n') + h(n') \]
  \[ \geq g(n) + c(n, a, n') + h(n') \]
  \[ = g(n) + h(n) = f(n) \]
- A consequence of consistency: \( f(n) \) non-decreasing along a path

Consistency and admissibility

- Consistency implies admissibility
- Hard to find heuristics that are admissible but not consistent
- Focus on consistent heuristics for proving optimality of A*

Consistency and the optimality of A*

- **Lemma:** Whenever A* selects a node \( n \) for expansion the optimal path to that node has been found (assuming consistent heuristic).
- Suppose not: Then there is an unexpanded node \( n' \) on the optimal path to \( n \).
  - From monotonicity: \( f(n) \geq f(n') \), so \( n' \) should have already been expanded.
- Therefore whenever a goal node is expanded, it is the lowest cost, i.e. optimal goal node
Properties of A*

- A* expands all nodes with \( f(n) < C^* \)
- But there can still be exponentially many such nodes!

A* expansion contours

- Expansion represented as contours of states with equal \( f \) value
- A* expands all nodes with \( f(n) < C^* \)
- A* may expand nodes on the goal contour

When a heuristic is “almost” admissible

- Graceful Decay of Admissibility
  - If a heuristic rarely overestimates cost by more than \( \delta \), then the A* algorithm will rarely find a solution whose cost is more that \( \delta \) greater than the cost of the optimal solution.
- Means:
  - So long as we undershoot almost all the time, and bound how much we overshoot, we seldom get in trouble, and the trouble is minor.

Evaluation of A*

- Completeness: YES
- Time complexity:
  - Number of nodes with \( f(n) < C^* \) can be exponential

Evaluation of A*

- Completeness: YES
- Time complexity:
  - Number of nodes with \( f(n) < C^* \) can be exponential
  - Space complexity: also exponential.
Evaluation of A*

- Completeness: YES
- Time complexity:
  - Number of nodes with \( f(n) < C^* \) can be exponential
- Space complexity: also exponential.
- Optimality: YES
  - A* does not expand any node with \( f(n) > C^* \)
  - Also optimally efficient (no other optimal algorithm is guaranteed to expand fewer nodes)

Memory-bounded heuristic search

- Some solutions to A* space problem (maintaining completeness and optimality)
  - Iterative-deepening A* (IDA*)
    - Like IDS, but cutoff information is the f-cost \((g+h)\) instead of depth
    - Expands by contour
  - Recursive best-first search (RBFS)
  - (Simplified) Memory-bounded A* ((S)MA*)
    - SMA*: Drop the worst-leaf node when memory is full (regenerate it later if necessary; back up the value of the forgotten node to its parent)

Comparing heuristics

Heuristics for the 8 puzzle:
- \( h_1 \): the number of misplaced tiles
- \( h_2 \): the sum of the Manhattan distances of the tiles from their goal positions
- For every state \( s \), \( h_2(s) \geq h_1(s) \)
- We say that \( h_2 \) dominates \( h_1 \)
- A dominating heuristic is better for search. WHY?

Inventing heuristics

- Admissible heuristics can be derived from the solution cost of a subproblem of a given problem.
  - For every state \( s \), \( h_j(s) \geq h_k(s) \) for any other heuristic \( k \).
  - For every state \( s \), \( h_j(s) \geq h_k(s) \) for any other heuristic \( k \).
- Admissibility: The optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem.

Inventing heuristics

- Admissible heuristics can also be derived from the solution cost of a subproblem of a given problem.
  - This cost is a lower bound on the cost of the real problem.
  - Construct a database of solutions for subproblems.
  - Use a combination of subproblems to define the heuristic.
Inventing heuristics

- Having the best of all worlds: given admissible heuristics $h_1, \ldots, h_m$
  
  $$h(n) = \max(h_1(n), \ldots, h_m(n))$$

  is a dominating admissible heuristic.

Inventing heuristics

- Learning from experience:
  - Experience = solving lots of 8-puzzles
  - A learning algorithm can be used to predict costs for states that arise during search.