Search

Dr. Melanie Martin
CS 4480
September 19, 2012

Outline

- Chapter 3
  - Best-first search
  - Greedy best-first search
  - A* search
  - Heuristics
- Chapter 4
  - Local search algorithms
  - Hill-climbing search
  - Simulated annealing search
  - Local beam search
  - Genetic algorithms

Best-first search

- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability"

  → Expand most desirable unexpanded node

- Implementation:
  Order the nodes in frontier in decreasing order of desirability

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

Heuristic

- Problem solving by experimental methods
  - Trial and error
- Heuristic function h(n)
  - Takes node as input
  - Depends only on state of node
  - Estimated cost of cheapest path from node n to a goal node
  - Numerical estimate of the "goodness" of a state

Greedy best-first search

- Evaluation function *f(n) = h(n)* (heuristic)
  = estimate of cost from *n* to *goal*

- e.g., $h_{SLD}(n)$ = straight-line distance from *n* to Bucharest

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

Properties of greedy best-first search

- Complete? No – can get stuck in loops, e.g., Iasi → Neamt → Iasi → Neamt →

- 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 from n to goal
- $f(n)$ = estimated total cost of path through n to goal

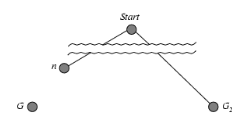
Admissible heuristics

- A heuristic $h(n)$ is admissible if for every node n, $h(n) \le h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)
- Theorem: If $h(n)$ is admissible, A^* using `TREE-SEARCH` is optimal

example

Optimality of A^* (proof)

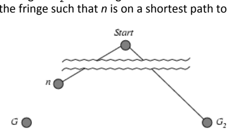
- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on a shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$ since G_2 is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ $f(G_2) = g(G_2) > g(G) = f(G)$

Optimality of A^* (proof)

- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on a shortest path to an optimal goal G.

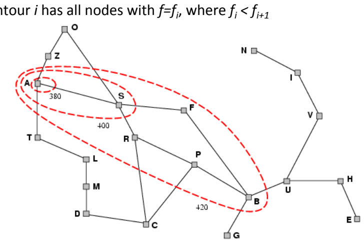


- $f(G_2) > f(G)$ from previous slide
- $h(n) \le h^*(n)$ since h is admissible – h*(n) is true cost
- $g(n) + h(n) \le g(n) + h^*(n)$
- $f(n) \le f(G)$ $\le f(G_2)$

Hence $f(G_2) > f(n)$, and A^* will never select G_2 for expansion

Optimality of A^*

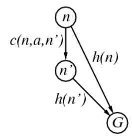
- A^* expands nodes in order of increasing f value
- Gradually adds "f-contours" of nodes
- Contour i has all nodes with $f=f_i$, where $f_i < f_{i+1}$



Consistent heuristics

- A heuristic is consistent if for every node *n*, every successor *n'* of *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'*:

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



- If *h* is consistent, we have

  f(n') = g(n') + h(n')
  = g(n) + c(n,a,n') + h(n')
  ≥ g(n) + h(n)
  = f(n)

- i.e., *f(n)* is non-decreasing along any path.
- Theorem: If *h(n)* is consistent, A* using `GRAPH-SEARCH` is optimal

Properties of A*

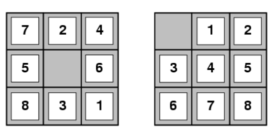
- Complete? Yes (unless there are infinitely many nodes with f ≤ *f(G)*)
- Time? Exponential
- Space? Keeps all nodes in memory
- Optimal? Yes

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)



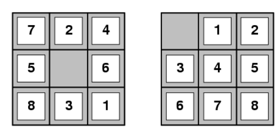
- $h_1(S)$ = ?
- $h_2(S)$ = ?

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = 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+2+3+3+2 = 18

Dominance

- If $h_2(n) \ge h_1(n)$ for all *n* (both admissible)
- then h_2 dominates h_1
- h_2 is better for search

- Typical search costs (average number of nodes expanded):

- *d=12* IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes
- *d=24* IDS = too many nodes
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes

Relaxed problems

- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original 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

Beyond Classical Search

- Chapter 4
  - Hill Climbing
  - Simulated Annealing
  - Beam Search
  - Genetic Algorithms