6.825 Techniques in Artificial Intelligence

Problem Solving and Search

Problem Solving
ñ Agent knows world dynamics [learning]
ñ World state is finite, small enough to enumerate [logic]
ñ World is deterministic [uncertainty]
ñ Utility for a sequence of states is a sum over path
ñ Agent knows current state [logic, uncertainty]

Few real problems are like this, but this may be a useful abstraction of a real problem

Relaxation of assumptions later in the course

Example: Route Planning in a Map

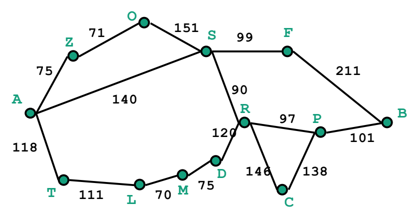
A map is a graph where nodes are cities and links are roads. This is an abstraction of the real world.
ñ Map gives world dynamics: starting at city X on the map and taking some road gets to you to city Y.
ñ World (set of cities) is finite and enumerable.
ñ World is deterministic: taking a given road from a given city leads to only one possible destination.
ñ Utility for a sequence of states is usually either total distance traveled on the path or total time for the path.
ñ We assume current state is known

Formal Definition

Problem:
• Set of states: S
• Initial state
• Operators (actions): S S
• Goal test: S { t, f }
• Path cost: (S, O)* **R**
• Sum of costs: c(S,O)

• Criteria for algorithms:
• Computation time/space
• Solution quality

Route Finding



Romania Map

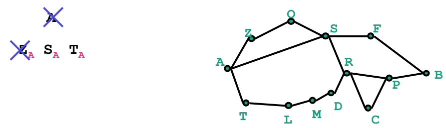
Search

ñ Put start state in the agenda
ñ Loop
  ñ Get a state from the agenda
    – If goal, then return
    – Expand state (put children in agenda)

Which state is chosen from the agenda defines the type of search and may have huge impact on effectiveness.

Depth-First Search

ñ Treat agenda as a stack (get most recently added node)
  ñ Expansion: put children at top of stack
  ñ Get new nodes from top of stack

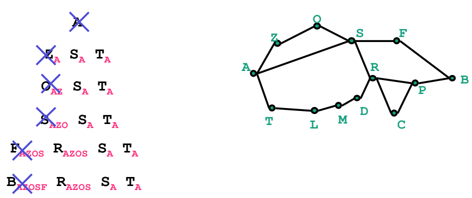


Avoiding Loops

- Method 1:
  - Don't add a node to the agenda if it's already in the agenda
  - Causes problems when there are multiple paths to a node and we want to be sure to get the shortest
- Method 2:
  - Don't expand a node (or add it to the agenda) if it has already been expanded.
  - We'll adopt this one for all of our searches

Depth-First Search

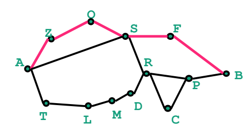
ñ Treat agenda as a stack (get most recently added node)
  ñ Expansion: put children at top of stack
  ñ Get new nodes from top of stack



Result = B_{AZOSF}

Properties of DFS

Sub-optimal answer:

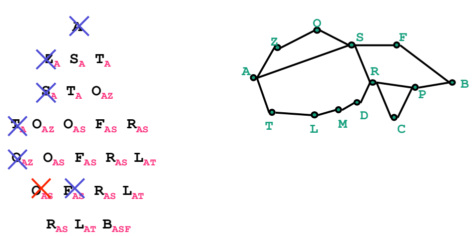


Let
- b = branching factor
- m = maximum depth
- d = goal depth

- $O(b^m)$ time
- O(mb) space

Breadth-First Search

ñ Treat agenda as a queue (get least recently added node)
  ñ Expansion: put children at end of queue
  ñ Get new nodes from the front of queue

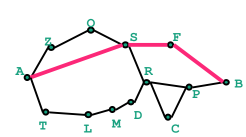


Result = B_{ASF}

Properties of BFS

Guaranteed to return shortest path (measured in number of arcs) from start to goal.

Let
- b = branching factor
- m = maximum depth
- d = goal depth



- $O(b^d)$ time
- $O(b^d)$ space

Iterative Deepening

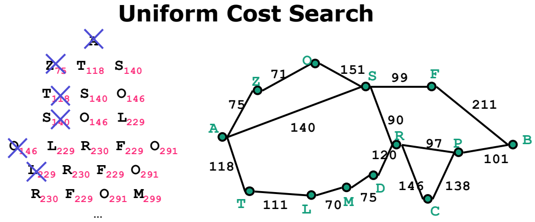
- DFS is efficient in space, but has no path-length guarantee
- BFS finds min-step path but requires exponential space
- Iterative deepening: Perform a sequence of DFS searches with increasing depth-cutoff until goal is found.

DFS cutoff depth	Space	Time
1	O(b)	O(b)
2	O(2b)	$O(b^2)$
3	O(3b)	$O(b^3)$
4	O(4b)	$O(b^4)$
…	…	…
d	O(db)	$O(b^d)$
Total	Max = O(db)	Sum = $O(b^{d+1})$

Uniform Cost Search

- Breadth-first and Iterative-Deepening find path with fewest steps (hops).
- If steps have unequal cost, this is not interesting.
- How can we find the shortest path (measured by sum of distances along path)?
- Uniform Cost Search:
  - Nodes in agenda keep track of total path length from start to that node
  - Agenda kept in priority queue ordered by path length
  - Get shortest path in queue
- Explores paths in contours of total path length; finds optimal path.

Uniform Cost Search



- We examine a node to see if it is the goal only when we take it off the agenda, not when we put it in.
- The algorithm is optimal only when the costs are non-negative.

Uniform Cost Search

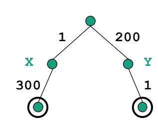
- Time cost is $O(b^m)$ [not $O(b^d)$ as in Russell&Norvig]
- Space cost could also be $O(b^m)$, but is probably more like $O(b^d)$ in most cases.

Uninformed vs. Informed Search

- Depth-first, breadth-first and uniform-cost searches are uninformed.
- In informed search there is an estimate available of the cost (distance) from each state (city) to the goal.
- This estimate (heuristic) can help you head in the right direction.
- Heuristic embodied in function h(n), estimate of remaining cost from search node n to the least cost goal.
- Graph being searched is a graph of states. Search algorithm defines a tree of search nodes. Two paths to the same state generate two different search nodes.
- Heuristic could be defined on underlying state; the path to a state does not affect estimate of distance to the goal.

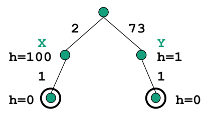
Using Heuristic Information

- Should we go to X or Y?
- Uniform cost says go to X
- If h(X) À h(Y), this should affect our choice
- If g(n) is path-length of node n, we can use g(n) + h(n) to prioritize the agenda
- This method is called A*
[pronounced "A star"]



Admissibility

- What must be true about h for A* to find optimal path?
- A* finds optimal path if h is admissible; h is admissible when it never overestimates.
- In this example, h is not admissible.
- In route finding problems, straight-line distance to goal is admissible heuristic.

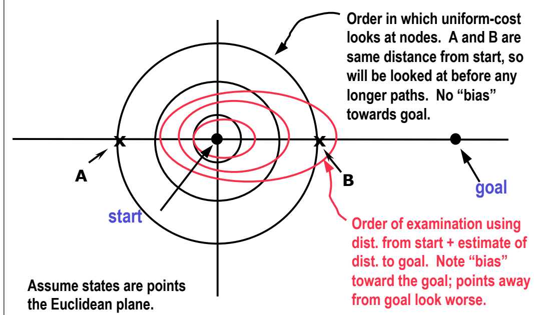


g(X)+h(X) = 102

g(Y)+h(Y) = 74

Optimal path is not found!

Why use estimate of goal distance?



Heuristics

- If we set h=0, then A* is uniform-cost search; h=0 is admissible heuristic (when all costs are non-negative).
- Very difficult to find heuristics that guarantee sub-exponential worst-case cost.
- Heuristic functions can be solutions to "relaxed" version of original problem, e.g. straight line distance is solution to route-finding when we relax constraint to follow the roads.

Search Problems

- In problem-solving problems, we want a path as the answer, that is, a sequence of actions to get from one state to another.
- In search problems, all we want is the best state (that satisfies some constraints).
  - Set of states: S
  - Initial state
  - Operators: S S
  - Cost (Utility): S

Example: Traveling Salesman

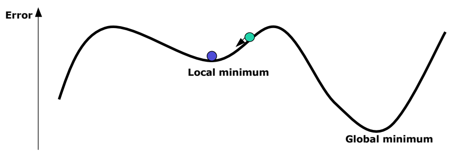
- In traveling salesman problem (TSP) we want a least-cost path that visits all the cities in a graph once.
- Note that this is not a route-finding problem, since we must visit every city, only the order of visit changes.
- A state in the search for TSP solution is a complete tour of the cities.
- An operator is not an action in the world that moves from city to city, it is an action in the information space that moves from one potential solution (tour) to another.
- Possible TSP operators: Swap two cities in a tour

Example: Square Root

- Given $y=x^2$ find x
- Utility of x is a measure of error, e.g. $U = 1/2\,(x^2 - y)^2$
- Operator: x x − r $_x$ U (for small stepsize r)
  - take a step down gradient (wrt x) of the error
  - For example, $x = x - r\,(x^2 - y)\,2x$
  - Assume y = 7, start with guess x = 3, let r = 0.01
  - Next guesses are: 2.880, 2.805, 2.756, …, 2.646
- We can prove that there is a unique x whose error value is minimal and that applying this operator repeatedly (for some value of r) will find this minimum x (to some specified accuracy).

Multiple Minima

- Most problems of interest do not have unique global minima that can be found by gradient descent from an arbitrary starting point.
- Typically, local search methods (such as gradient descent) will find local minima and get stuck there.
- How can we escape from local minima?
  - Take some random steps!
  - Re-start from randomly chosen starting points



Simulated Annealing

- T = initial temperature
- x = initial guess
- v = Energy(x)
- Repeat while T > final temperature
  - Repeat n times
    - x^0 □ Move(x)
    - v^0 = Energy(x^0)
    - If v^0 < v then accept new x [x □ x^0]
    - Else accept new x with probability exp(-(v^0 – v)/kT)
  - T = 0.95T /* for example */
- At high temperature, most moves accepted (and can move between "basins")
- At low temperature, only moves that improve energy are accepted

Search Demonstration

- There's a cool applet at UBC for playing around with search algorithms: www.cs.ubc.ca/labs/lci/CIspace

Recitation Problems

Problems 3.17 a, b, f (Russell & Norvig)

What would be a good heuristic function in that domain? (from 3.17)

What would be a good heuristic function for the Towers of Hanoi problem? (look this up on the web, if you don't know about it)

Other practice problems that we might talk about in recitation: 4.4, 4.11a,b