Artificial Intelligence & Machine Learning

Bodhayan Roy

Department of Mathematics,
Indian Institute of Technology Kharagpur

Lecture 5

Search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

Informed Search strategies

- An informed search strategy uses problem-specific knowledge beyond the definition of the problem itself.
- More efficient than uninformed strategies.
- We concentrate on **best-first search**,in which a node is selected for expansion based on an evaluation function, f(n). The evaluation function is construed as a cost estimate, so the node with the lowest evaluation is expanded first.

Greedy best-first search

- Tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly.
- Thus, it evaluates nodes by using just the heuristic function; that is, $\mathrm{f}(\mathrm{n}) = \mathrm{h}(\mathrm{n})$.
- For shortest path problems, we use the straight-line distance heuristic, which is denoted by h_{SLD}.

A* search

- Best-first search that combines g(n) and h(n).
- Recall that g(n) is the cost to reach the node, and h(n) is the cost to get from the node to the goal.
- So, $f(n) = g(n) + h(n)$ where f(n) is the estimated cost of the cheapest solution through n.
- The algorithm is identical to UCS except that A* uses $g + h$ instead of g.

A* search

- It requires for optimality is that h(n) be an **admissible heuristic**. An admissible heuristic is one that never overestimates the cost to reach the goal.
- Another slightly stronger condition called **consistency** (or sometimes **monotonicity**) is required only for applications of A* to graph search.

A* search

A heuristic h(n) is consistent if, for every node n and 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' :
$h(n) \leq c(n, a, n') + h(n')$

the tree-search version of A* is optimal if h(n) is admissible, while the graph-search version is optimal if h(n) is consistent.

A* search

If h(n) is consistent, then the values of f(n) along any path are nondecreasing. Suppose n' is a successor of n; then g(n') = g(n) + c(n, a, n') for some action a, and we have $f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n)$.

Whenever A* selects a node n for expansion, the optimal path to that node has been found (logic same as in UCS).
So the sequence of nodes expanded by A* graph search is in nondecreasing order of f(n). Hence, the first goal node selected for expansion must be an optimal solution because f is the true cost for goal nodes (which have h = 0) and all later goal nodes will be at least as expensive.