Harvard CS’s Artificial Intelligence with Python

Search

Knowledge

Uncertainty

Learning

Neural Networks

Language

---

Search / Tìm kiếm

Some analogy

Agent = entity that perceives its environment and acts upon that environment.

State = an configuration of the agent and its environment.

Initial state = the starting point of the algorithm.

Actions = choices that can be made in state.

Action(S) return the set of actions that can be executed in state s.

Transition model = a description of what state results from performing any appilcable action in any state.

Result(s,a) returns the state resulting from performing action a in state s.

State space = the set of all states reacheable from the initial state by any sequence of actions.

Goal test = way to determine whether a given state is a goal state.

Path cost = numerical cost associated with given path.

Search problems

* Initial state
* Actions
* Transition model
* Goal test
* Path cost function

Optimal Solution = a sequence that has the lowest path cost among all solutions.

Node / a data structure that keeps track of a state/ a parent (node that generate this node)/ an action (action that applied to parent to get node)/ a path cost (from initial state to node).

Approach

* Start with a frontier that contains the initial state
* Repeat:

+if the frontier is empty, then no solution

+remove a node from frontier

+if node contains goal state, return the solution

+expand node, add resulting nodes to the frontier

What could go wrong ?

Revised approach

* Start with a frontier that contains the initial state
* Start with an empty explored set
* Repeat:

+if the frontier is empty, then no solution

+remove a node from frontier \*

+if node contains goal state, return the solution

+add the node to the explored set

+expand node, add resulting nodes to the frontier if they aren’t already in the frontier or the explored set.

Depth-First search = search algorithm that always expands the deepest node in the frontier (Stack / last-in first-out data type LIPO)

Breadth-First search = search algorithm that always expand the shallowest node in the frontier (Queue / first-in first out data type FIFO

)

Uniformed search = search strategy that uses no problem- specific knowledge

Informed search = search strategy that uses problem-specific knowledge to find solutions more efficiently

Greedy best-first search = search algo that expand the node that is closest to the goal, as estimated by a heuristic function h(n)

A\* search

Search Algorithm that expands node with lowest value of g(n) + h(n)

G(n) = cost to reach node

H(n) = estimated cost to goal

Optimal if

H(n) is admissible (never overesitmates the true cost), and

H(n) is consistent (for evert node n and successor n’ with step cost c, h(n) <= h(n’) + c)

Adversarial search

Minimax algorithm

Max (X) aims to maximize score

Min (O) aims to minimize score

Game :

S0 = initial state

Player (s) : return which player to move in state s

Action (s): return legal moves in state s

Result (s,a): return state after action a taken in state s

Terminal (s): check if state s is a terminal state

Ultility (s): final numerical value for terminal state s

Minimax

Given a state s:

* Max picks action a in ACTIONS(s) that produces highest value of MIN-Value(RESULT(s,a))
* MIN picks action a in ACTIONS(s) that produces smallest value of MAX-VALUE(RESULT(s,a))

Function MAX-VALUE(state):

If TERMINAL(state):

Return ULTILITY(state)

V = -infiinty

For action in ACTIONS(state):

V = MAX(v,MIN-VALUE(RESULT(state,action)))

Return v

Function MIN-VALUE(state):

If TERMINAL(state):

Return ULTILITY(state)

V = infiinty

For action in ACTIONS(state):

V = MIN(v,MAX -VALUE(RESULT(state,action)))

Return v

Optimization

Alpha-Beta Pruning

Depth-limited minimax

Evaluation function = function that estimates the expected ultility of the game from a given state.