Agents

Rational: Maximally achieving goals (actions that maximize utility function)

Reflex Based: Chooses action based on current percept (no future consideration)

Goal Based: Chooses action based on consequences (model of how the world reacts)

Utility Based: Goal based with trading off of multiple goals and uses probabilities

Search

Def: Possible states, Successor function $f(n) \rightarrow (n'$, action, cost), start and goal state

Complete: Guaranteed to find a solution if one exists

Optimal: Guaranteed to find the least cost path

Properties: n= number of states, b= maximum branching factor, C^*= optimal cost, d= depth of shallowest solution, m= max depth, ϵ = min cost of all actions

Conformant Planning: Set of actions that always work (sterilizing surgical gear)

Blind Search

DFS: Fringe uses a Stack, complete iff finite, not optimal, time: O(b^m), space: O(bm)

BFS: Fringe uses a Queue, complete, optimal (constant), time and space: O(b^d)

IDDFS: Fringe uses a Stack, complete, optimal (constant), time: O(b^d), space: O(bm)

Heuristic Search

Heuristic $h(n)$ = An estimate of how close a state is to a goal

Admissible: Always an underestimate to the true lowest cost

Consistent: Always $h(n) <= h(n') + stepCost(n')$ where n' is a neighbor of n

Best First: Fringe uses a PriorityQueue with cost fuction for each node

Uniform Cost: Best First with $f(n)$ = sum of edge costs from start to n (explores increasing contours), complete, optimal, time and space: O($b^{\frac{C^*}{\epsilon}}$)

Greedy: Best First with $f(n) = h(n)$ (suboptimal goal is common)

A*: Best First with $f(n) = g(n) + h(n)$ with $g(n)$ = sum of costs from start to n

IDA*: Depth bound is now $F_{limit} = h(start)$, prune if $f(n) > F_{limit}$, $F'_{limit} = min(pruned\ nodes)$, uses space of DFS, time depends on # of unique F values

Beam: Best First with $|Fringe| = K$, not complete, time: O(b^d), space: O($b + K$)

Hill Climbing: Always choose best child (Beam Search with K = 1)

Tabu: Keep fixed length queue of states to not visit again (use with hill climbing)

Stochastic Search

Hill Climbing++ Restarts: Generate random state when plateaued

Hill Climbing++ Walk: With prob p move to the neighbor with largest value, with $(1-p)$ move to a random neighbor

Hill Climbing++ (Both): Greedy move, random walk, or random restart

Simulated Annealing: Pick a random neighbor and calculate the change in ‘energy’ or objective function δ, if it is positive then move to that state. Otherwise, move to this state with probability $e^{\frac{\delta}{T}}$ where T is decreased as the algorithm runs longer. High T $\rightarrow$ probability of bad move is higher and vice versa

Genetic: Start with a population of random states, use an evaluation (fitness) function, produce next generation using random selection / crossover / random mutation

Gradient Descent: Move in the direction of the gradient at each step

Constraint Satisfaction Problems