# Artificial Intelligence notes class 1 28/8

Search

* Algorithms:
  + Breadth First Search
    - Frontier is first in first out queue.
    - Expand all nodes in a layer before progressing to the next layer
  + Depth First Search
    - Frontier is first in last out stack
    - We expand through all layers until we cannot proceed any further, then backtrack and try other routes.
  + Problems and solutions with BFS and DFS
    - Both BFS and DFS have an issue with storing all nodes that are visited in large databases.
    - If we don’t store the nodes that we have visited, then DFS offers huge memory advantages
    - Easier to implement in a tree
    - DFS will find the goal node if there are no loops in a finite search space. If the space is infinite or contains directed loops there is no guarantee
  + Iterated Deepening
    - Seeks to solve problems with DFS with loops
    - Run a DFS alg repeatedly, with a tuning parameter t
    - At each iteration we treate nodes at depth t as if there are no transitions from them, terminating depth first strings and it means we backtrack to examine the remainder of the nodes reachable in up to t transitions from the origin
    - Guaranteed to reach the goal in finite steps so long as
      * Each state has only finitely many transitions from it
      * The goal state is accessible by finite transitions from the origin
  + Bidirectional Search
    - Only possible if we know the goal state
    - Run two simultaneous searches: forward from the initial state and backward from the goal state
    - Requires backwards search
    - Useful in finding paths to a destination
  + Best First Search
    - Order frontier by fitness
    - At each step in the search process we expand the fittest currently known node
    - Problem: Search spaces are very large, even if we use a tree search best first search will end up storing a lot of nodes
    - Possible solution: If we think of the node we choose to expand at each step as our current location, we can hope that simply continuing to choose the best node from whatever our current location is will lead us to find the optimally fit node
    - Still have issue of local maxima
    - Best First Search types: Local Search
      * Parts:
        + Current node, initially the origin
        + Transition function, f(n,N)=m, that takes the current node n and a set of nodes M and returns a node m from that set (essentially just used to determine where to go next from current location)
        + Termination function (g(n,m,T), that takes the current node n and another node m and possible tuning parameters T as arguments and returns a bool that decides whether the algorithm terminates.
      * Steps:
        + Expand the current node, n, performing all possible transitions on it to find its neighbors, M
        + Find the potential transition m = f(n,M) and see if we should transition to m or terminate and return the current node as the output of the algorithm by checking g(n,m,T)
  + Greedy Hill Climb
    - Transition to the fittest neighbor’s node if it is fitter than the current node, otherwise terminate the search and return the current node as the solution
    - Suffers from issue of local maxima/optima as well.
    - Can mitigate local maxima problem by using random restarts
  + Simulated Annealing
    - Attempts to avoid local maxima if occasionally the algorithm goes down instead of up all the time, might transition from a 6 to a 5 if 6 is the highest to “see what’s over there”
    - Typically make transition from n to m based on a probability that changes the more transitions are made, represented by t
    - As t approaches 0 the chances of choosing a less fit state approach 0 as well.
    - Important to note cooling function of t as it controls the probability of going downhill
  + Local Beam Search
    - Select n initial nodes, each called a ‘particle’
    - At each step:
      * Create a set s of nodes consisting of the nodes the particles are currently at and their respective neighbors
      * Transition the particle to the n fittest nodes in S
    - The search terminates when no transitions are made
    - Note that we do not choose the best neighbor for each climb, instead we choose the best n neighbors among all climbs and have the particles transition to these
    - The individual particles communicate with each other, like swarm intelligence
    - Helps to find the global optimum
    - Has issue where particles end up clustered together quickly, limiting the effectiveness of getting a greater awareness state than GHC
  + Stochastic Local Beam Search
    - Like local Beam Search but the particles transition to n nodes in S (the set of neighboring nodes) at random but weighted by the relative fitness
    - Helps to avoid clustering issues
  + Dynamic Programming for Path Finding
    - Assign the origin the value 0
    - Proceed iteratively through each layer, L of the state space. Treat the origin as the – layer and begin at Layer 1
      * Assign each node n in L the value (min(value(m) + (the value of an edge from node m to node n)), m∈L - 1
      * Add a pointer from the node m that minimized the above equation to n
    - Back track from the goal to the origin using these pointers to find the optimal path
    - More efficient, if we have m layers each with n nodes, dynamic programming reduces an optimal path finding problem from nm to m(n2)
  + Algorithm A
    - Best-first search algorithm that uses the following cost function
      * F(n) = g(n) + h(n)
      * F(n) = estimated shortest path from start to goal via n
      * G(n) = length of shortest path to n found so far
      * H(n) = heuristic, estimated distance from n to goal
  + Algorithm A\*
    - Best-first search algorithm where
      * f(n) = g(n) + h(n)
      * h is optimistic
        + h(n) ≤h\*(n) for all nodes n
      * h is monotonic
        + h(n) ≤ h(n`) +t(n,n`) for all nods n and all successors n` of n
    - Guarantees that when we have visited the goal, we have found the shortest path possible
    - Requires optimism
    - If we are using graph search (not revisiting visited nodes) it must be monotonic (implies optimism)