Lecture 3 - informed search strategies

Tuesday, October 14, 2025 8:03 AM

Last time => uninformed search strategies
- Have a predefined model for investigating the states
- This is a disadvantage => cannot be tuned to particular problem
- We discussed bfs and dfs

Today => Informed search strategies
- Have the ability to investigate subspace in a smart way
- They use heuristic functions => help guide search towards solution
- They will be defined depending on the problem we wish to solve
- Not always easy to define such a function

We will see 3 strategies:
- How to select next node to be expanded during search
- How to decide successors based on heuristic func
- How to eliminate part of generated nodes from the search space
    - Next time we'll see an algo exactly w this

1. Best-first strategy
2. A* algorithm => improvement to the best first strat

Best-first strategy:
- Quality of node is determined in more ways:
    - Assign a difficulty degree to the node
        - For ex by the nr of successors
    - Check parts of solution containing that node
    - Forgot what 3rd one was
- In all these cases, the quality of a node is determined by the heuristic function
    - Heuristic func is denoted by **w(n)**
    - In general, w is chosen such that the cost is minimized
        - The node to be expanded is the one with minimum value of w(n)
- **How to choose w(n):**
    - We choose it as being the sum of the cost of the edges up to that point
        - If we don't have cost, we can associate cost 1 to each edge so the heuristic function will be the length of the path
        - So this cost is from to the initial state up to the current state
    - We can choose is by trying to minimize the search effort
        - We can try to estimate how many edges we still need to pass through from the current state from the next state
        - We cant know from the beginning but we can estimate
            - We usually underestimate this rather than overestimate, that's how a good heuristic function is (I don't understand why and he said that's just how its done xd)

A algorithm = a best-first algorithm*
- Components of heuristic func in A*:
    - We have part g(S) and part h(S)
        - h(s) is the estimation?
    - The big function is f(S), f = g + h
    - But for us the interesting part is how to choose this h

- The node to be expanded will be the one with the minimum value of the function f
- Computing g(S):
  - g(S) is the sum of the cost in the path between intial and current state
- Computing h(S):
  - Always a positive value for h
  - This heuristic function must always underestimate => its an optimistic function
  - It usually says that the number of steps needed is smaller than the number that is actually needed in reality (again, idk why)
- **Definition**:
  - If we are capable of finding such a function h we will say that the heuristic function is **admissible: 0<=h(S) <= h*(S)**
    - It should be admissible wrt any state, not just the current one
  - Then we can define the **admissibility property** of the A* strategy algorithm
- **Theorem**:
  - An A* algo is admissible when it is guaranteed to find the minimal cost path towards the solution
    - This happens if:
      - The function h is admissible
      - The cost is finite
  - So in fact it all reduces to choosing a good heuristic function
- More characteristics of heuristic func choosing:
  - Must be positive
  - Must underestimate real cost
  - Must be as close as possible to real cost
  - Worst case scenario: h(S) is 0 => this would be nonsense
    - The A* algo would be reduced to an uniform cost search strategy
  - For ex if we have f1(S) = g(S) + h1(S) and f2(S) = g(S) + h2(S)
  - Then A2 is more informed than A1 if for any S we have h2(s)>h1(S)
    - This means that A2 will never expand more states than A1
- Determining g(S):
  - Usually sum of cost of the edges from initial state to current state
  - If no costs then we choose 1 for each
- Determining h(S) is most important

Missionaries and Cannibals Problem

- We have 3 missionaries and 3 cannibals on the east side of a river
- We wish to transport them towards west side
- We have a boat with just 2 places
- Our goal is to transport all persons towards the west side such that they are safely transported
- Safely means that on each side of the river it is impossible to have more cannibals than missionaries because if this happens, the cannibals will eat the missionaries
- For this problem we can choose multiple heuristic functions
  - H1(S) = $n^E(S)$
  - H2(S) = $n^E(S)/2$
  - H3 - more complex

Observation:

- A* algo is similar to Dijkstra
  - Diff is that in A* we have this heuristic function
  - Also Dijkstra works poorly on big graphs
  - This one works good on huge graphs
  - So it is an improvement to dijkstra

Other examples:

1. 8-puzzle problem
    - It was found that in order to reach final state we need around 20 steps on avergage (20 movements)
    - Branching factor is around 3 (he says last time you discussed that its 2.67)
    - If we perform an exhaustive search (generating all successors for each state) we can generate all the subspace and we would have around 20 movements so our subspace would have depth 20
        - So how many states?
        - Branching factor ^{depth} = 3^{20} ~ 3.5 * 10^9 states
        - Even if we eliminate the repeated states we still have 9! = 362880
        - This is too large so we need to find a good heuristic function that results in smth way lower
    - ***First possibility of heuristic function:***
        - H1 = the number of tiles that are in the wrong position in our current state wrt the position in the final state
        - For ex if just one tile is in the correct place then we can assume that we would need to move each other tile at least once
            - so we can say h_1 = 8-1 => h_1 = 7

              In other words: h_1= 1+1+1+1+1+0+1+1 = 8
        - This is admissible but it is not as close as possible to the real cost which is ~20
    - ***Second heuristic function:***
        - H_2= the sum of the distances of the tiles from their goal position
            - This is also called **Manhattan distance**
            - H_2 = 2+3+3+2+4+2+0+2=18 => this is a better approximation so it is a better heuristic function than h_1

2. Distances of roads between cities in romania
    - Using Best-first search:
        - The distances are not the real ones
        - Our goal is to go from Arad to Bucharest
        - As refference we use the straight-line distances and we are sure that they will be lower than the cost we have by going through the road
        - So w(n) is the straight line distance
        - We always take the one with the least w value => greedy approach => not always optimal
        - Problem: when the smallest w is on a path that simply ends before reaching the final state, we get blocked there
        - Worst case time and space complexity: $O(B^m)$
    - Using A*:
        - f(n) = g(n) + h(n)
            - G = cost to reach n by going through the path on the map
            - H = straight line distances
        - So now we choose successors based on the one which has minimum value for the function f

He said we should look at the video he put on moodle about A*