CS-208:Artificial Intelligence
Lectures-08
A* Algorithm

Evaluation Function

A more flexible way to use heuristic information is to use some criteria to reorder all the nodes on OPEN at every step. In order to apply such an ordering procedure, we need some measure by which to evaluate the promise of a node. Such measures are called evaluation function.

The value of the evaluation function ***f'*** (***n***)at any node '***n***' is the cost of the path from start node '***s***' to the node '***n***' **PLUS** an estimate of the cost of the minimal cost path from the node '***n***' to the goal node.

$$f'(n) = g'(n) + h'(n)$$

f'(***n***) is an estimate of the cost of the minimal cost path constrained to go though '***n'***.

A* Algorithm

Step-1: Add the start node ***s*** to a list called ***OPEN*** and compute $f'(s)=0+h'(s)$. Set ***CLOSED*** as an empty list.

Step-2: If ***OPEN*** is empty then exit with ***Failure*** else continue.

Step-3: Pick the node on ***OPEN*** with the lowest ***f'*** value. Call it as ***Best_Node*** and remove from ***OPEN*** and add into ***CLOSED.***

Step-4: If the ***Best_Node*** is a goal node exit with solution path which is obtained by tracing back through the pointers else continue.

Step-5: Expand the ***Best_Node*** by generating all of its successors. For each successor do the following:

a) Set a pointer back from the successor to the ***Best_Node.*** (this will be useful for recovering the path, once goal node has been found).

b) Compute ***g'(***successor) = ***g'***(***Best_Node***) + the cost of getting from ***Best_Node*** to successor***.***

c) Check if the successor is same as any node on ***OPEN***. If so, call that node as ***OLD*** and add it to the list of successors to the ***Best_Node***. Find whether it is cheaper to get ***OLD*** via its current parent or to get successor via ***Best_Node.*** IF ***OLD*** is cheaper then do nothing. If successor is cheaper then reset the ***OLD***'s parant link to point the ***Best_Node***. Record the new cheaper path by updating ***g'***(***OLD***) and ***f'***(***OLD***).

d) If the successor was not on ***OPEN*** then check if it is on ***CLOSED***. If so, call that node as ***OLD*** and add it to the list of successors to the ***Best_Node***. Find whether it is cheaper to get ***OLD*** via its current parent or to get successor via ***Best_Node.*** IF ***OLD*** is cheaper then do nothing. If successor is cheaper then reset the ***OLD***'s parant link to point the ***Best_Node***

-Perform DFS staring at ***OLD*** and update each node's ***g'*** value that is visited. Terminate each branch when either at node with no successor or a node to which equivalent or better path has been found is reached.

e) If the successor was not already on either ***OPEN*** or ***CLOSED***, then add it to ***OPEN***. Compute ***f'***(successor) = ***g'***(successor)+ ***h'***(successor).

f) Go to the Step-2