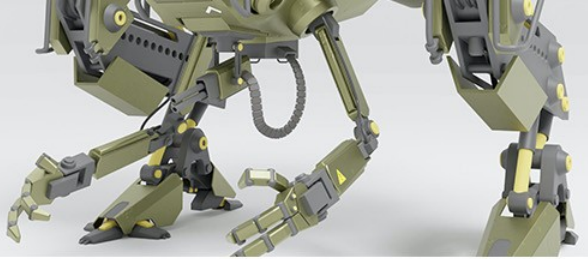
UNIT II

Introduction

Contents

Unit II	Problem-solving	07 Hours
Solving Problems by Searching, Problem-Solving Agents, Example Problems, Search Algorithms, Uninformed Search Strategies, Informed (Heuristic) Search Strategies, Heuristic Functions, Search in Complex Environments, Local Search and Optimization Problems.		
#Exemplar/Case Studies	4th Industrial Revolution Using AI, Big Data And Robotics	
***Mapping of Course Outcomes for Unit II**	CO2, CO4	

Informed Search



- Algorithms have information on the goal state which helps in more efficient searching.
- This information is obtained by a function that estimates how close a state is to the goal state.
- Informed search in AI is a type of search algorithm that uses additional information to guide the search process, allowing for more efficient problem-solving compared to uninformed search algorithms.
- This information can be in the form of heuristics, estimates of cost, or other relevant data to prioritize which states to expand and explore.
- Examples of informed search algorithms include A* search, Best-First search, and Greedy search.

Informed Search

- **Use of Heuristics** – informed search algorithms use heuristics, or additional information, to guide the search process and prioritize which nodes to expand.
- **More efficient** – informed search algorithms are designed to be more efficient than uninformed search algorithms, such as breadth-first search or depth-first search, by avoiding the exploration of unlikely paths and focusing on more promising ones.
- **Goal-directed** – informed search algorithms are goal-directed, meaning that they are designed to find a solution to a specific problem.
- **Cost-based** – informed search algorithms often use cost-based estimates to evaluate nodes, such as the estimated cost to reach the goal or the cost of a particular path.
- **Prioritization** – informed search algorithms prioritize which nodes to expand based on the additional information available, often leading to more efficient problem-solving.
- **Optimality** – informed search algorithms may guarantee an optimal solution if the heuristics used are admissible (never overestimating the actual cost) and consistent (the estimated cost is a lower bound on the actual cost).

Heuristics function

- Heuristic is a function which is used in Informed Search, and it finds the most promising path.
- It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal.
- The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time.
- Heuristic function estimates how close a state is to the goal. It is represented by h(n), and it calculates the cost of an optimal path between the pair of states.
- The value of the heuristic function is always positive.
- Admissibility of the heuristic function is given as:

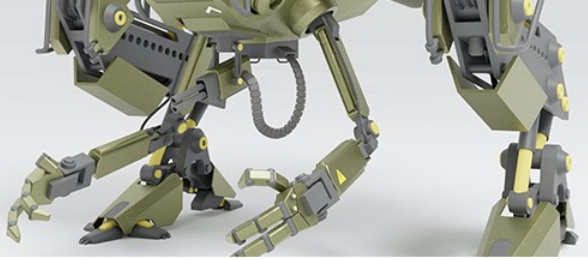
$$h(n) <= h^*(n)$$

- Here h(n) is heuristic cost, and h*(n) is the estimated cost. Hence heuristic cost should be less than or equal to the estimated cost.

Heuristics function

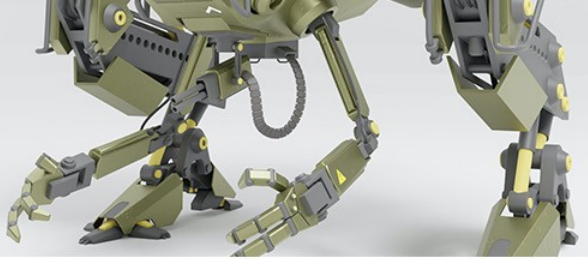
- Pure heuristic search is the simplest form of heuristic search algorithms.
- It expands nodes based on their heuristic value h(n). It maintains two lists, OPEN and CLOSED list. In the CLOSED list, it places those nodes which have already expanded and in the OPEN list, it places nodes which have yet not been expanded.
- On each iteration, each node n with the lowest heuristic value is expanded and generates all its successors and n is placed to the closed list. The algorithm continues unit a goal state is found.

A* Search Algorithm

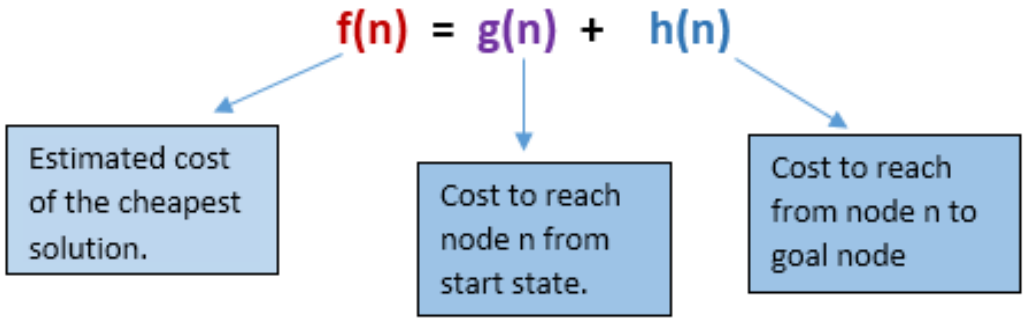


- A* search is the most commonly known form of best-first search.
- It uses heuristic function h(n), and cost to reach the node n from the start state g(n).
- It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently.
- A* search algorithm finds the shortest path through the search space using the heuristic function.
- This search algorithm expands less search tree and provides optimal result faster.
- A* algorithm is similar to UCS except that it uses **g(n)+h(n)** instead of g(n)

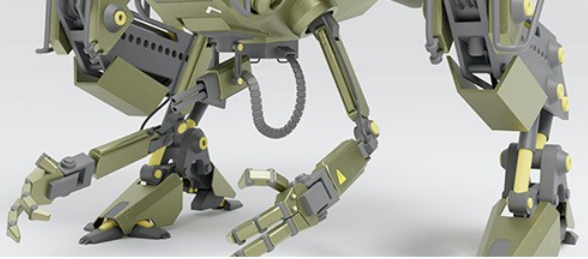
A* Search Algorithm



- In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a fitness number.

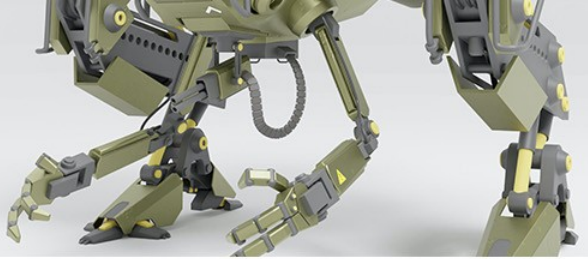


A* Search Algorithm

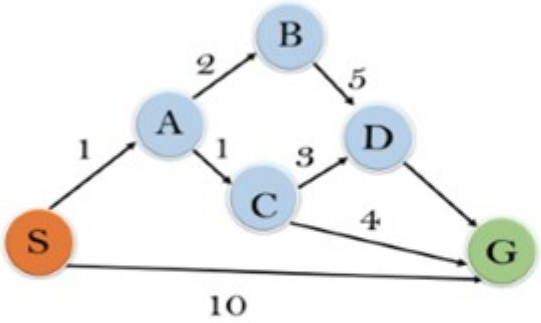


- Algorithm of A* search
  Step1: Place the starting node in the OPEN list.
  Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.
  Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise
  Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.
  Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.
  Step 6: Return to Step 2.

A* Search Algorithm



- Example



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

A* Search Algorithm

- Advantages:

1) A* search algorithm is the best algorithm than other search algorithms.
2) A* search algorithm is optimal and complete.
3) This algorithm can solve very complex problems.

- Disadvantages:

1) It does not always produce the shortest path as it mostly based on heuristics and approximation.
2) A* search algorithm has some complexity issues.
3) The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

A* Search Algorithm

- **Complete:** A* algorithm is complete as long as:
  Branching factor is finite.
  Cost at every action is fixed.
- **Optimal:** A* search algorithm is optimal if it follows below two conditions:
  Admissible: the first condition requires for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
  Consistency: Second required condition is consistency for only A* graph-search.
- If the heuristic function is admissible, then A* tree search will always find the least cost path.
- **Time Complexity:** The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.
- Space Complexity: The space complexity of A* search algorithm is O(b^d)

A* Search Algorithm

- **Complete:** A* algorithm is complete as long as:
  Branching factor is finite.
  Cost at every action is fixed.
- **Optimal:** A* search algorithm is optimal if it follows below two conditions:
  Admissible: the first condition requires for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
  Consistency: Second required condition is consistency for only A* graph-search.
- If the heuristic function is admissible, then A* tree search will always find the least cost path.
- **Time Complexity:** The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.
- Space Complexity: The space complexity of A* search algorithm is O(b^d)