Artificial Intelligence(23CP307T)

Simulated Annealing, AO*, MEA

Course Coordinator: Dr. Sidheswar Routray
Course Faculty: Dr. Shilpa Pandey,Dr. Davinder Singh,Dr. Trishna Paul,Dr. Azriel Henry
Department of Computer Science & Engineering
School of Technology

Simulated Annealing:

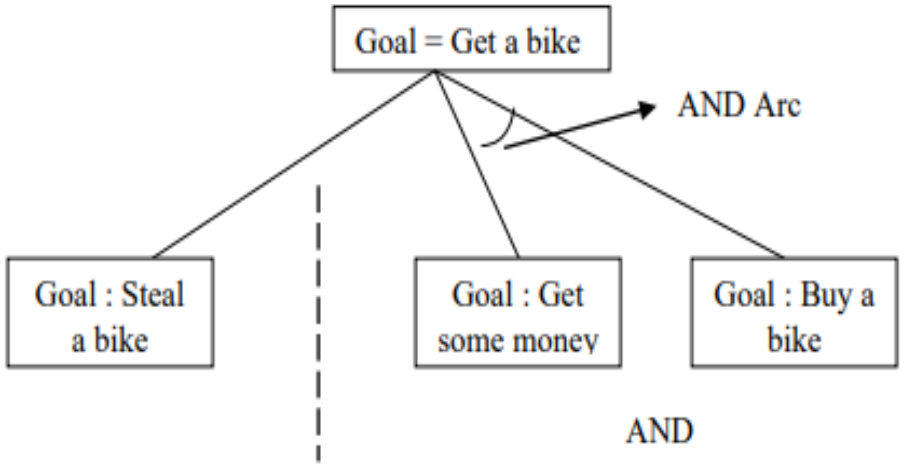
- A hill-climbing algorithm which never makes a move towards a lower value guaranteed to be incomplete because it can get stuck on a local maximum. And if algorithm applies a random walk, by moving a successor, then it may complete but not efficient. **Simulated Annealing** is an algorithm which yields both efficiency and completeness.
- In mechanical term **Annealing** is a process of hardening a metal or glass to a high temperature then cooling gradually, so this allows the metal to reach a low-energy crystalline state.
- The same process is used in simulated annealing in which the **algorithm picks a random move, instead of picking the best move**. If the random move improves the state, then it follows the same path.
- Otherwise, the algorithm follows the path which has a probability of less than 1 or it moves downhill and chooses another path.

PROBLEM REDUCTION SEARCH:

- We already know about the divide and conquer strategy, a solution to a problem can be obtained by decomposing it into smaller sub-problems. Each of this sub-problem can then be solved to get its sub solution. These sub solutions can then recombined to get a solution as a whole.
- That is called is **Problem Reduction**. This method generates arc which is called as **AND** arcs. One AND arc may point to any number of successor nodes, all of which must be solved in order for an arc to point to a solution.

AND-OR GRAPHS/AO* Algorithm

- The AND-OR GRAPH (or tree) is useful for representing the solution of problems that can solved by decomposing them into a set of smaller problems, all of which must then be solved.
- This decomposition, or reduction, generates arcs that we call AND arcs. One AND arc may point to any number of successor nodes, all of which must be solved in order for the arc to point to a solution.
- Just as in an OR graph, several arcs may emerge from a single node, indicating a variety of ways in which the original problem might be solved.
- This is why the structure is called not simply an AND-graph but rather an AND-OR graph (which also happens to be an AND-OR tree)



- In an AND-OR graph, OR node represents a choice between possible decompositions, and an AND node represents given decomposition.
- For example, to Get a bike, we have two options, either: 1. (Steal a bike) OR 2.
- Get some money AND Buy a Bike.
- In this graph we are given two choices, first Steal a bike or get some money AND Buy a Bike.
- When we have more than one choice and we have to pick one, we apply OR condition to choose one.(That's what we did here).
- Basically, the ARC here denotes AND condition. Here we have replicated the arc between the Get some money and buy a bike because by getting some money possibility of buying a bike is more than stealing.
- AO* search algorithm is based on AND-OR graph, so it is called AO* search algorithm. AO* Algorithm basically based on problem decomposition (Breakdown problem into small pieces).

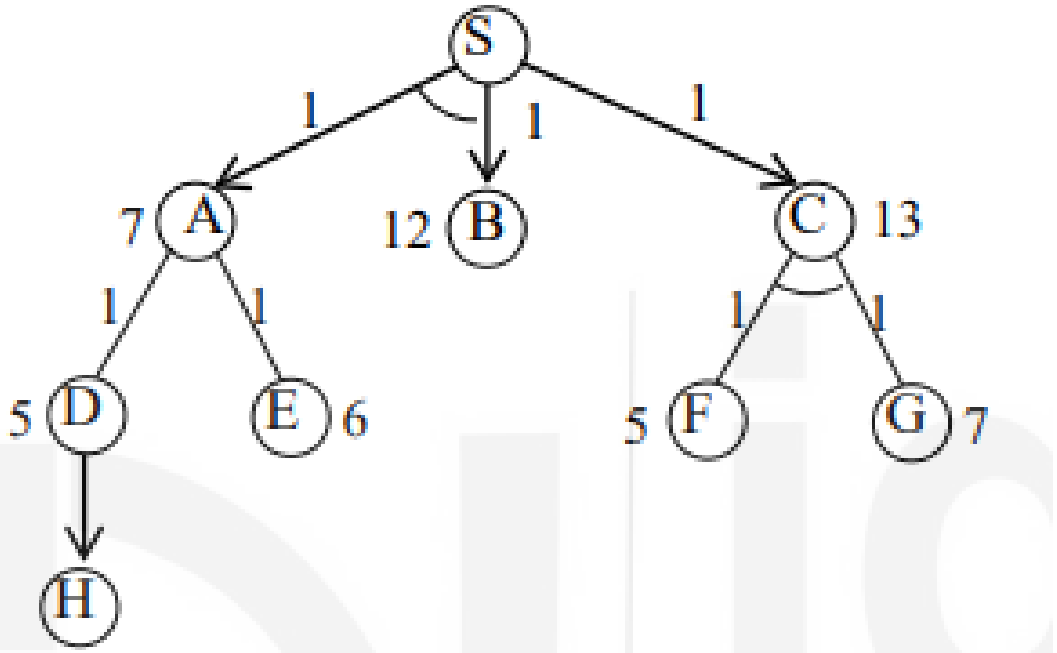
How AO* works:

- AO* explores a search graph with **AND** and **OR nodes**.
- At AND nodes, all child nodes must be solved to proceed.
- At OR nodes, only one child node needs to be solved to continue.
- This algorithm is useful for tasks such as **planning**, **game strategy development**, and **automated reasoning**.

- Use **A*** if you are working with pathfinding or graph traversal problems, especially when you need the shortest path between two points.
- Use **AO*** if the problem involves dependencies between sub-goals, such as in game trees, planning systems, or decision-making scenarios.

Example1:

- Let us see one example with the presence of heuristic value at every node. The estimated heuristic value is given at each node. The heuristic value h(n) at any node indicates “from this node at least h(n) value (or cost) is required to find solution”. Here we assume the edge cost value (i.e., g(n) value) for each edge is 1. Remember in OR node we always mark that successor node which indicates best path for solution.



Solution:

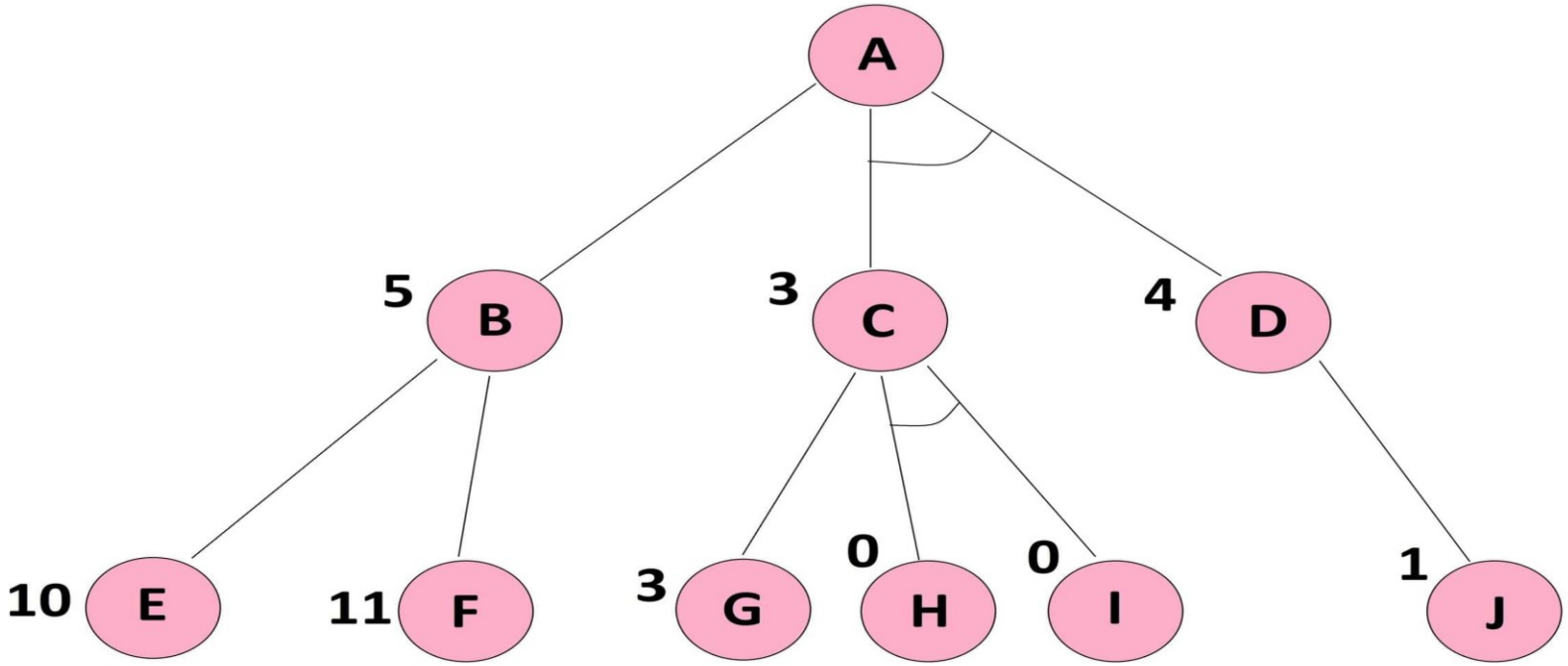
- Note that, the graph given in fig, there are two paths for solution from start state S: either S-A-B or S-C.
- To calculate the cost of the path we use the formula f(n)=g(n)+h(n) [note that here g(n) value is 1 for every edge].
- Path1: f(S-A-B)=1+1+7+12=21
- Path2: f(S-C)=1+13=14
- Since min(21,14)=14; so, we select successor node C, as its cost is minimum, so it indicates best path for solution.
- Note that C is a AND node; so, we consider both the successor node of C.
- The cost of node C is f(C-F-G)=1+1+5+7=14; so, the revised cost of node C is 14 and now the revised cost of node S is f(S-C)=1+14=15 (revised).

Solution:

- Note that once the cost (that is f value) of any node is revised, we propagate this change backward through the graph to decide the current best path.
- Now let us explore another path and check whether we are getting lessor cost as compared to this cost or not.
- F(A-D)=1+5=6 and f(A-E)=1+6=7; since A is an OR node so best successor node is D since min(6,7)=6.
- So revised cost of Node A will be 6 instead of 7, that is f(A)=6 (revised).
- Now next selected node is D and D is having only one node H so f(D-H)=1+2=3, so the revised cost of node D is 3, so now the revised cost of node A, that is f(A-D-H)=4.
- This path is better than f(A-E)=7. So, the final revised cost of node A is 4.
- Now the final revised cost of f(S-A-B)=1+1+4+12=18 (revised)
- Thus, the final revised cost for Path1: f(S-A-B)=18 and Path2: f(S-C)=15
- So optimal cost is 15.

Example2:

- It should be noted that the cost of each edge is the same as 1, and the heuristic cost to reach the goal node from each node of the graph is shown beside it. Apply AO* algorithm and find the optimal cost path using AO* algorithm.



Solution:

Forward Propagation

- First, we begin from node A and calculate each of the OR side and AND side paths.
- The OR side path P(A-B) = G(B) + H(B) = 1 + 5 = 6, where 1 is the cost of the edge between A and B, and 5 is the estimated cost from B to the goal node.
- The AND side path P(A-C-D) = G(C) + H(C) + G(D) + H(D) = 1 + 3 + 1 + 4 = 9,
- where the first 1 is the cost of the edge between A and C, 3 is the estimated cost from C to the goal node, the second 1 is the cost of the edge between A and D, and 4 is the estimated cost from D to the goal node.

Solution:

Reaching the Last Level and Back Propagation

- In this step we continue on the P(A-B) from B to its successor nodes i.e., E and F, where P(B-E) = 1 + 10 = 11 and P(B-F) = 1 + 11 = 12.
- Here, P(B-E) has a lower cost and would be chosen.

- Now, we have reached the bottom of the graph where no more level is given to add to our information.
- Therefore, we can do the backpropagation and correct the heuristics of upper levels.
- In this vein, the updated H(B) = P(B-E) = 11, and as a consequence the updated P(A-B) = G(B) + updated H(B) = 1 + 11 = 12.

Solution:

- Now, we can see that P(A-C-D) with a cost of 9 is lower than the updated P(A-B) with a cost of 12.
- Therefore, we need to proceed on this path to find the minimum cost path from A to the goal node.

- It is worth mentioning that if the updated P(A-B) had a lower cost than P(A-C-D), then we were done, and no more calculations would be required.

Solution:

Correcting the Path from Start Node

- In this step, we do the calculations for the AND side path, i.e., P(A-C-D), and first explore the paths attached to node C.
- In this node again we have an OR side where P(C-G) = 1 + 3 = 4, and an AND side where P(C-H-I) = 1 + 0 + 1 + 0 = 2, and as a consequence the updated H(C) = 2.
- Also, the updated H(D) = 2, since P(D-J) = 1 + 1 = 2.
- By these updated values for H(C) and H(D), the updated P(A-C-D) = 1 + 2 + 1 + 2 = 6.
- This updated P(A-C-D) with the cost of 6 is still less than the updated P(A-B) with the cost of 12, and therefore, the minimum cost path from A to the goal node goes from P(A-C-D) by the cost of 6. We are done.

Means-Ends Analysis in Artificial Intelligence

- We have studied the strategies which can reason either in forward or backward, but a mixture of the two directions is appropriate for solving a complex and large problem.
- Such a mixed strategy, make it possible that first to solve the major part of a problem and then go back and solve the small problems arise during combining the big parts of the problem. Such a technique is called **Means-Ends Analysis**.
- Means-Ends Analysis is problem-solving techniques used in Artificial intelligence for limiting search in AI programs.
- It is a mixture of Backward and forward search technique.
- The MEA analysis process centered on the evaluation of the difference between the current state and goal state.

How means-ends analysis Works:

- The means-ends analysis process can be applied recursively for a problem. It is a strategy to control search in problem-solving. Following are the main Steps which describes the working of MEA technique for solving a problem.
- First, evaluate the difference between Initial State and final State.
- Select the various operators which can be applied for each difference.
- Apply the operator at each difference, which reduces the difference between the current state and goal state.

Operator Subgoaling

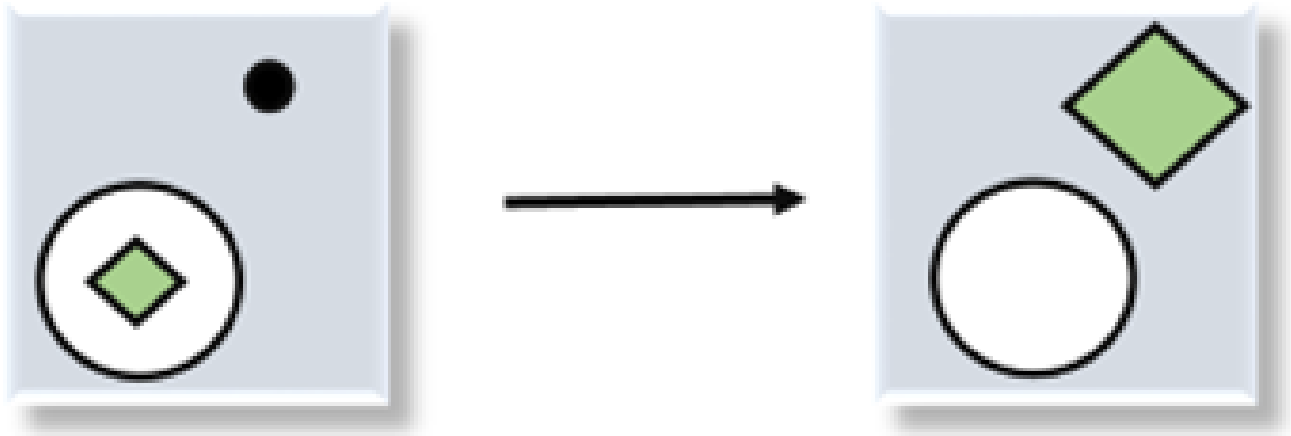
- In the MEA process, we detect the differences between the current state and goal state. Once these differences occur, then we can apply an operator to reduce the differences.
- But sometimes it is possible that an operator cannot be applied to the current state. So we create the subproblem of the current state, in which operator can be applied, such type of backward chaining in which operators are selected, and then sub goals are set up to establish the preconditions of the operator is called **Operator Subgoaling**.

Algorithm for Means-Ends Analysis:

- Let's we take Current state as CURRENT and Goal State as GOAL, then following are the steps for the MEA algorithm.
- **Step 1:** Compare CURRENT to GOAL, if there are no differences between both then return Success and Exit.
- **Step 2:** Else, select the most significant difference and reduce it by doing the following steps until the success or failure occurs.
  - Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.
  - Attempt to apply operator O to CURRENT. Make a description of two states.
    i) O-Start, a state in which O?s preconditions are satisfied.
    ii) O-Result, the state that would result if O were applied In O-start.
  - If
    (First-Part <------ MEA (CURRENT, O-START)
    And
    (LAST-Part <----- MEA (O-Result, GOAL), are successful, then signal Success and return the result of combining FIRST-PART, O, and LAST-PART.

Example of Mean-Ends Analysis:

- Let's take an example where we know the initial state and goal state as given below. In this problem, we need to get the goal state by finding differences between the initial state and goal state and applying operators.



Solution:

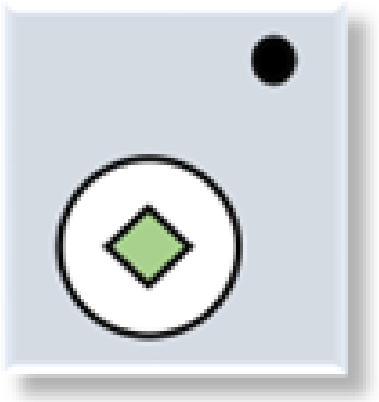
To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

Move

Delete

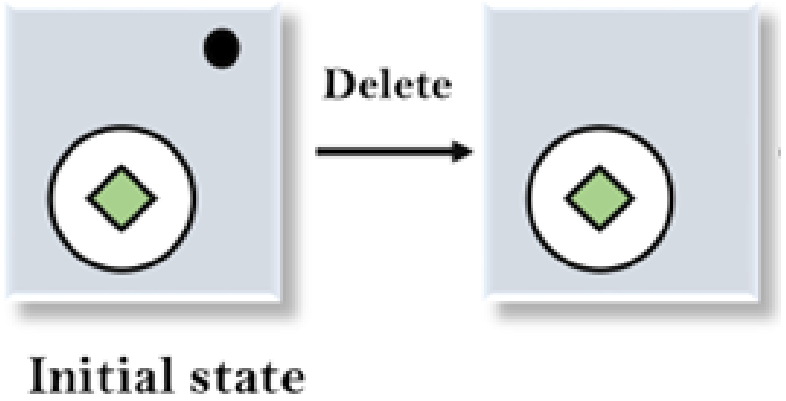
Expand

1. Evaluating the initial state: In the first step, we will evaluate the initial state and will compare the initial and Goal state to find the differences between both states.

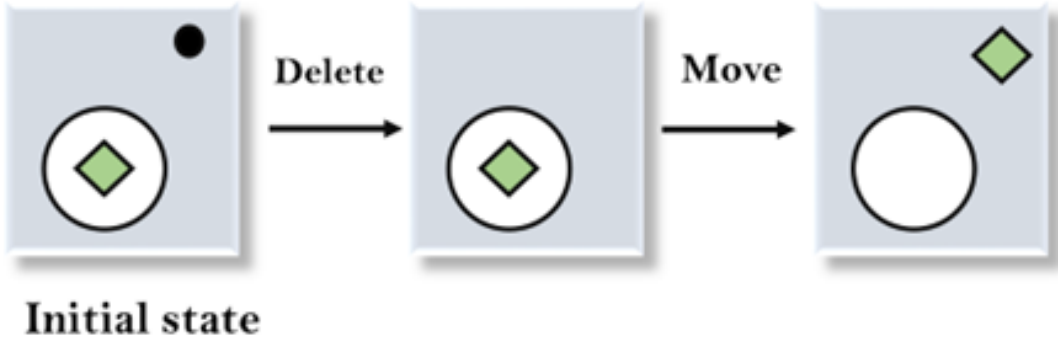


Initial state

2. Applying Delete operator: As we can check the first difference is that in goal state there is no dot symbol which is present in the initial state, so, first we will apply the **Delete operator** to remove this dot.



3. Applying Move Operator: After applying the Delete operator, the new state occurs which we will again compare with goal state. After comparing these states, there is another difference that is the square is outside the circle, so, we will apply the **Move Operator**.



4. Applying Expand Operator: Now a new state is generated in the third step, and we will compare this state with the goal state. After comparing the states there is still one difference which is the size of the square, so, we will apply **Expand operator**, and finally, it will generate the goal state.