

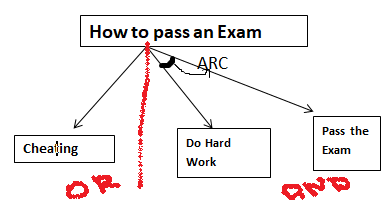
**AO\* Algorithm**

AO\* Algorithm basically based on problem decompositon (Breakdown problem into small pieces)

When a problem can be divided into a set of sub problems, where each sub problem can be solved separately and a combination of these will be a solution, **AND-OR graphs**or **AND - OR trees**are used for representing the solution.

The decomposition of the problem or problem reduction generates AND arcs.

**AND-OR Graph**



**The figure shows an AND-OR graph**

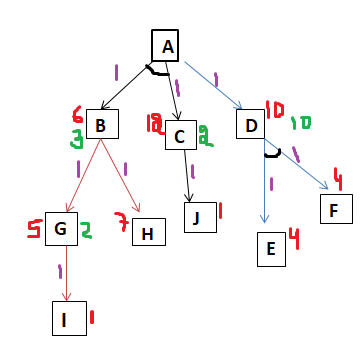
1. To pass any exam, we have two options, either cheating or hard work.
2. In this graph we are given two choices, first do cheating **or (The red line)**work hard and **(The arc)**pass.
3. When we have more than one choice and we have to pick one, we apply **OR condition**to choose one.(That's what we did here).
4. Basically the **ARC**here denote**AND condition**.
5. Here we have replicated the arc between the work hard and the pass because by doing the hard work possibility of passing an exam is more than cheating.

### ****A\* Vs AO\*****

1. Both are part of informed search technique and use heuristic values to solve the problem.
2. The solution is guaranteed in both algorithm.
3. A\*  **always** gives an **optimal solution** (shortest path with low cost) But It is not guaranteed to that**AO\*** always provide **an optimal solutions**.
4. **Reason:** Because AO\* does not explore all the solution path once it got solution.

### ****How AO\* works****

Let's try to understand it with the following diagram



The algorithm always moves towards a **lower cost value.**

Basically, We will calculate the **cost function** here **(F(n)= G (n) + H (n))**

**H:** **heuristic/ estimated** value of the nodes. and**G:**actual cost or edge value (here unit value).

Here we have taken the **edges value 1**, meaning we have to focus solely on the **heuristic value.**

1. The **Purple color**values are **edge values (here all are same that is one).**
2. The **Red color**values are **Heuristic values for nodes.**
3. **The Green color**values are**New Heuristic values for nodes.**

**Procedure:**

1. In the above diagram we have two ways from**A to D**or**A to B-C**(because of and condition). calculate cost to select a path
2. **F(A-D)= 1+10 = 11**            and               **F(A-BC) = 1 + 1 + 6 +12 = 20**
3. As we can see**F(A-D)**is less than **F(A-BC)**then the algorithm choose the path **F(A-D).**
4. Form D we have one choice that is **F-E.**
5. **F(A-D-FE) = 1+1+ 4 +4 =10**
6. Basically **10** is the cost of reaching **FE from D.** And **Heuristic value of node D** also denote the cost of reaching **FE from D**. So, the new Heuristic value of D is 10.
7. And the Cost from A-D remain same that is **11**.

Suppose we have searched this path and we have got the **Goal State**, then we will never explore the other path. (this is what AO\* says  but here we are going to explore other path as well to see what happen)

**Let's Explore the other path:**

1. In the above diagram we have two ways from**A to D**or**A to B-C**(because of and condition). calculate cost to select a path
2. **F(A-D)= 1+10 = 11**            and               **F(A-BC) = 1 + 1 + 6 +12 = 20**
3. As we know the cost is more of **F(A-BC)**but let's take a look
4. Now from B we have two path G and H , let's calculate the cost
5. **F(B-G)= 5+1 =6**  and **F(B-H)= 7 + 1 = 8**
6. So, cost from**F(B-H)** is more than **F(B-G)**we will take the path B-G.
7. The Heuristic value fro**m G to I is 1 but let's calculate the cost form G to I.**
8. **F(G-I) = 1 +1 = 2.**which is less than **Heuristic value 5**. So, the new**Heuristic value form G to I is 2.**
9. If it is a new value, then the cost from**G to B**must also have changed. Let's see the new **cost form (B to G)**
10. F(B-G)= 1+2 =3 . Mean the**New Heuristic value of B is 3.**
11. **But A is associated with both B and C .**
12. As we can see from the diagram**C only have one choice or one node to explore that is J.** The Heuristic value of C is 12.
13. Cost form C to **J= F(C-J) = 1+1= 2** Which is less than Heuristic value
14. No**w the New Heuristic value of C is 2.**
15. **And the New Cost from A- BC that is F(A-BC) = 1+1+2+3 = 7 which is less than F(A-D)=11.**
16. In this case Choosing path A-BC is more cost effective and good than that of A-D.

But this will only happen when the algorithm explores this path as well. But according to the algorithm, algorithm will not accelerate this path **(here we have just did  it to see how the other path can also be correct).**

But it is not the case in all the cases that it will happen in some cases that the algorithm will get optimal solution.

# **Hill Climbing Algorithm in Artificial Intelligence**

* Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
* Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
* It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
* A node of hill climbing algorithm has two components which are state and value.
* Hill Climbing is mostly used when a good heuristic is available.
* In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

## **Features of Hill Climbing:**

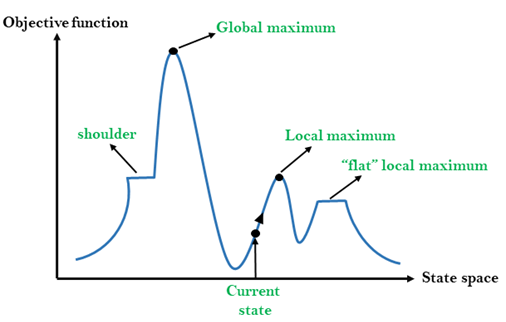
Following are some main features of Hill Climbing Algorithm:

* **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
* **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
* **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

## **State-space Diagram for Hill Climbing:**

The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.

On Y-axis we have taken the function which can be an objective function or cost function, and state-space on the x-axis. If the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum. If the function of Y-axis is Objective function, then the goal of the search is to find the global maximum and local maximum.



## **Different regions in the state space landscape:**

**Local Maximum:** Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.

**Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.

**Current state:** It is a state in a landscape diagram where an agent is currently present.

**Flat local maximum:** It is a flat space in the landscape where all the neighbor states of current states have the same value.

**Shoulder:** It is a plateau region which has an uphill edge.

## **Types of Hill Climbing Algorithm:**

* Simple hill Climbing:
* Steepest-Ascent hill-climbing:
* Stochastic hill Climbing:

### 1. Simple Hill Climbing:

Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state**. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state. This algorithm has the following features:

* Less time consuming
* Less optimal solution and the solution is not guaranteed

### Algorithm for Simple Hill Climbing:

* **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
* **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
* **Step 3:** Select and apply an operator to the current state.
* **Step 4:** Check new state:
  1. If it is goal state, then return success and quit.
  2. Else if it is better than the current state then assign new state as a current state.
  3. Else if not better than the current state, then return to step2.
* **Step 5:** Exit.

### 2. Steepest-Ascent hill climbing:

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors

### Algorithm for Steepest-Ascent hill climbing:

* **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
* **Step 2:** Loop until a solution is found or the current state does not change.
  1. Let SUCC be a state such that any successor of the current state will be better than it.
  2. For each operator that applies to the current state:
     1. Apply the new operator and generate a new state.
     2. Evaluate the new state.
     3. If it is goal state, then return it and quit, else compare it to the SUCC.
     4. If it is better than SUCC, then set new state as SUCC.
     5. If the SUCC is better than the current state, then set current state to SUCC.
* **Step 5:** Exit.

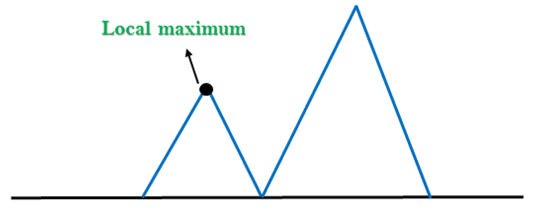
### 3. Stochastic hill climbing:

Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

## **Problems in Hill Climbing Algorithm:**

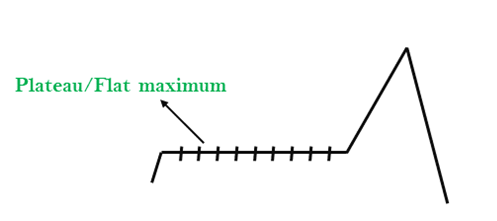
**1. Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

**Solution:** Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



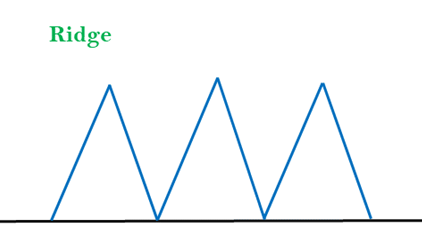
**2. Plateau:** A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

**Solution:** The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



**3. Ridges:** A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

**Solution:** With the use of bidirectional search, or by moving in different directions, we can improve this problem.



### 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.