Artificial Intelligence

Unit-2 (Heuristic search Techniques)

Artificial Intelligence 01CE0702

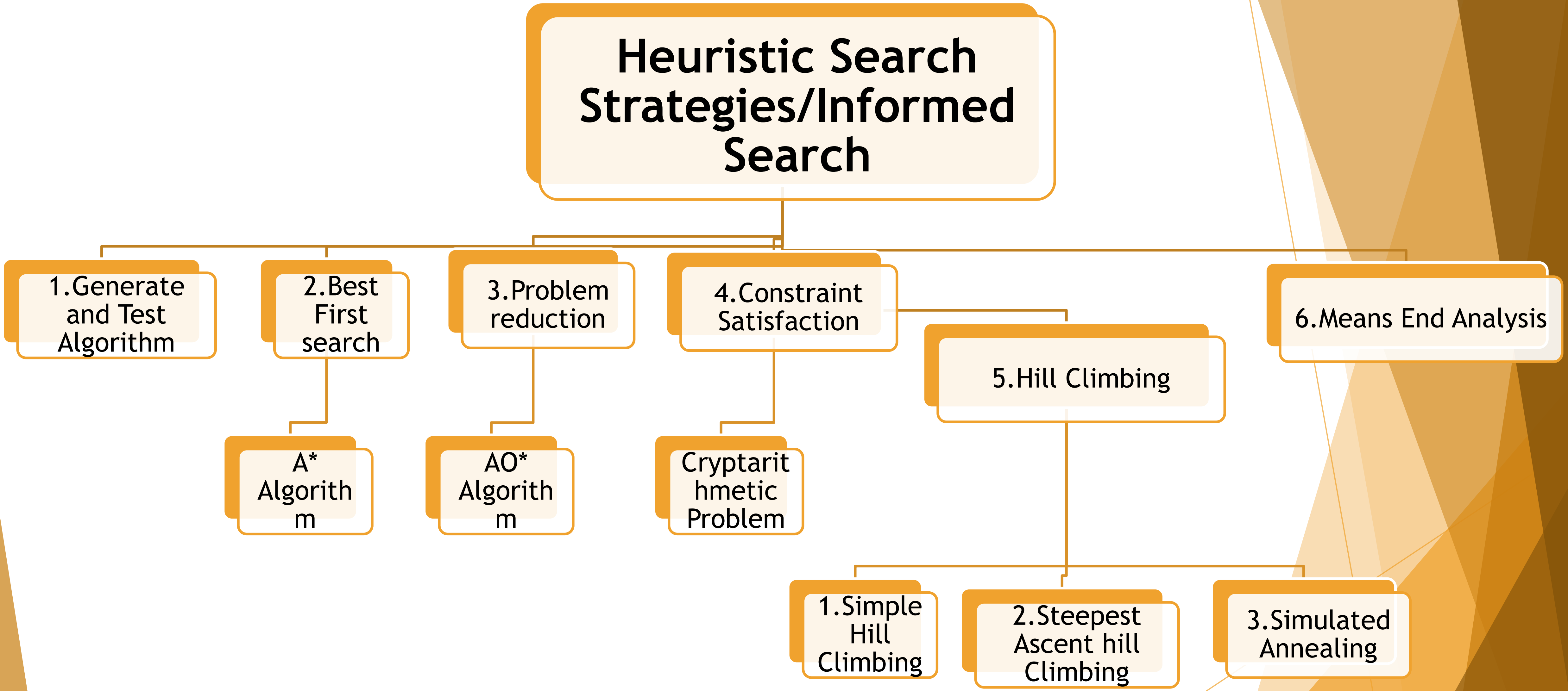
Marwadi University

Department of Computer Engineering

Shilpa Singhal

Topics to be Covered

- Heuristic Search Techniques
- Generate-And-Test, Hill Climbing, Best-First Search, Problem Reduction, Constraint Satisfaction, Means-Ends Analysis


Heuristic Search Strategies/Informed Search
1.Generate and Test Algorithm
2.Best First search
3.Problem reduction
4.Constraint Satisfaction
5.Hill Climbing
6.Means End Analysis
A* Algorith m
AO* Algorith m
Cryptarit hmetic Problem
1.Simple Hill Climbing
2.Steepest Ascent hill Climbing
3.Simulated Annealing

1.Generate-and-Test Strategy

Algorithm

1. Generate a possible solution.
2. Test to see if this is actually a solution.
3. Quit if a solution has been found. Otherwise, return to step 1.

Properties

- Acceptable for simple problems.
- Inefficient for problems with large space.

Types of Generate and test

- **Exhaustive** generate-and-test.
- **Heuristic** generate-and-test: not consider paths that seem unlikely to lead to a solution.
- **Plan** generate-test:
  – Create a list of candidates.
  – Apply generate-and-test to that list.

Example: Generate and test

Example: coloured blocks
“Arrange four 6-sided cubes in a row, with each side of each cube painted one of four colours, such that on all four sides of the row one block face of each colour is showing.”

Heuristic: if there are more red faces than other colours then, when placing a block with several red faces, use few of them as possible as outside faces.

2.Constraint Satisfaction

- Operates in a space of constraint sets.
- Initial state contains the original constraints given in the problem.
- A goal state is any state that has been constrained “enough”.

Two-step process:

1. Constraints are discovered and propagated as far as possible.
2. If there is still not a solution, then search begins, adding new constraints.

Cryptarithmetic puzzle:

Many AI problems can be viewed as problems of constraint satisfaction.

Cryptarithmetic puzzle:

"It is an arithmetic problem which is represented in letters. It involves the decoding of digit represented by a character. It is in the form of some arithmetic equation where digits are distinctly represented by some characters. The problem requires finding of the digit represented by each character. Assign a decimal digit to each of the letters in such a way that the answer to the problem is correct. If the same letter occurs more than once, it must be assigned the same digit each time. No two different letters may be assigned the same digit".

Procedure:

Cryptarithmatic problem is an interesting constraint satisfaction problem for which different algorithms have been developed. Cryptarithm is a mathematical puzzle in which digits are replaced by letters of the alphabet or other symbols. Cryptarithmatic is the science and art of creating and solving cryptarithms. The different constraints of defining a cryptarithmatic problem are as follows.

1) Each letter or symbol represented only one and a unique digit throughout the problem.

2) When the digits replace letters or symbols, the resultant arithmetical operation must be correct.

The above two constraints lead to some other restrictions in the problem.

Example:

```
   SEND
+  MORE
_______
  MONEY
```

```
   . EAT
 + THAT
-------------------
 A PP LE
```

Solution

Step 1: In the above problem, M must be 1. You can visualize that, this is an addition problem. The sum of two four digit numbers cannot be more than 10,000. Also M cannot be zero according to the rules, since it is the first letter. So now you have the problem like

```
   S END
 + 1 0 RE
 -----------
 1 0 NEY
```

Step 2: Now in the column s10, s+1 ≥ 10. S must be 8 because there is a 1 carried over from the column EON or 9. O must be 0 (if s=8 and there is a 1 carried or s = 9 and there is no 1 carried) or 1 (if s=9 and there is a 1 carried). But 1 is already taken, so O must be 0.

Step 3: There cannot be carry from column EON because any digit +0 < 10, unless there is a carry from the column NRE, and E=9; But this cannot be the case because then N would be 0 and 0 is already taken. So E < 9 and there is no carry from this column. Therefore S=9 because 9+1=10.

```
   9 E N D
 + 1 O R E
-----------
 1 O N E Y
```

Step 4:

In the column EON, E cannot be equal to N. So there must be carry from the column NRE; E+1=N. We now look at the column NRE, we know that E+1=N. Since we know that carry from this column, N+R=1E (if there is no carry from the column DEY) or N+R+1=1E (if there is a carry from the column DEY).

Let us see both the cases:

No carry: $N + R = 10 + (N - 1) = N + 9$

$R = 9$

But 9 is already taken, so this will not work

Carry: $N + R + 1 = 9$

$R = 9 - 1 = 8$ This must be solution for R

```
   9 E N D
 + 1 0 8 E
-------------------
 1 O N E Y
```

Step 5: Now just think what are the digits we have left? They are 7, 6, 5, 4, 3 and 2. We know there must be a carry from the column DEY. So D E % 10.N E 1, So E cannot be 7 because then N would be 8 which is already taken. D is almost 7, so E cannot be 2 because then D E & 10 and E cannot be 3 because then D E 10 and Y 0, but 0 is already taken. Also E cannot be 4 because if D % 6, D E & 10 and if D 6 or D 7 then Y 0 or Y 1, which are both taken. So E is 5 or 6. If E 6, then D 7 and Y 3. So this part will work but look the column N8E. Point that there is a carry from the column D5Y.N 8 1 16(As there is a carry from this column). But then N=7 and 7 is taken by D therefore E=5.

$$\begin{array}{r} 9\ 5\ N\ D \\ +\ 1\ 0\ 8\ 5 \\ \hline\hline 1\ 0\ N\ 5\ Y \end{array}$$

Step 6:

Now we have gotten this important digit, it gets much simpler from here. N+8+1=15, N=6

```
   9 5 6 D
 + 1 0 8 5
-------------------
 1 0 6 5 Y
```

Step 7:

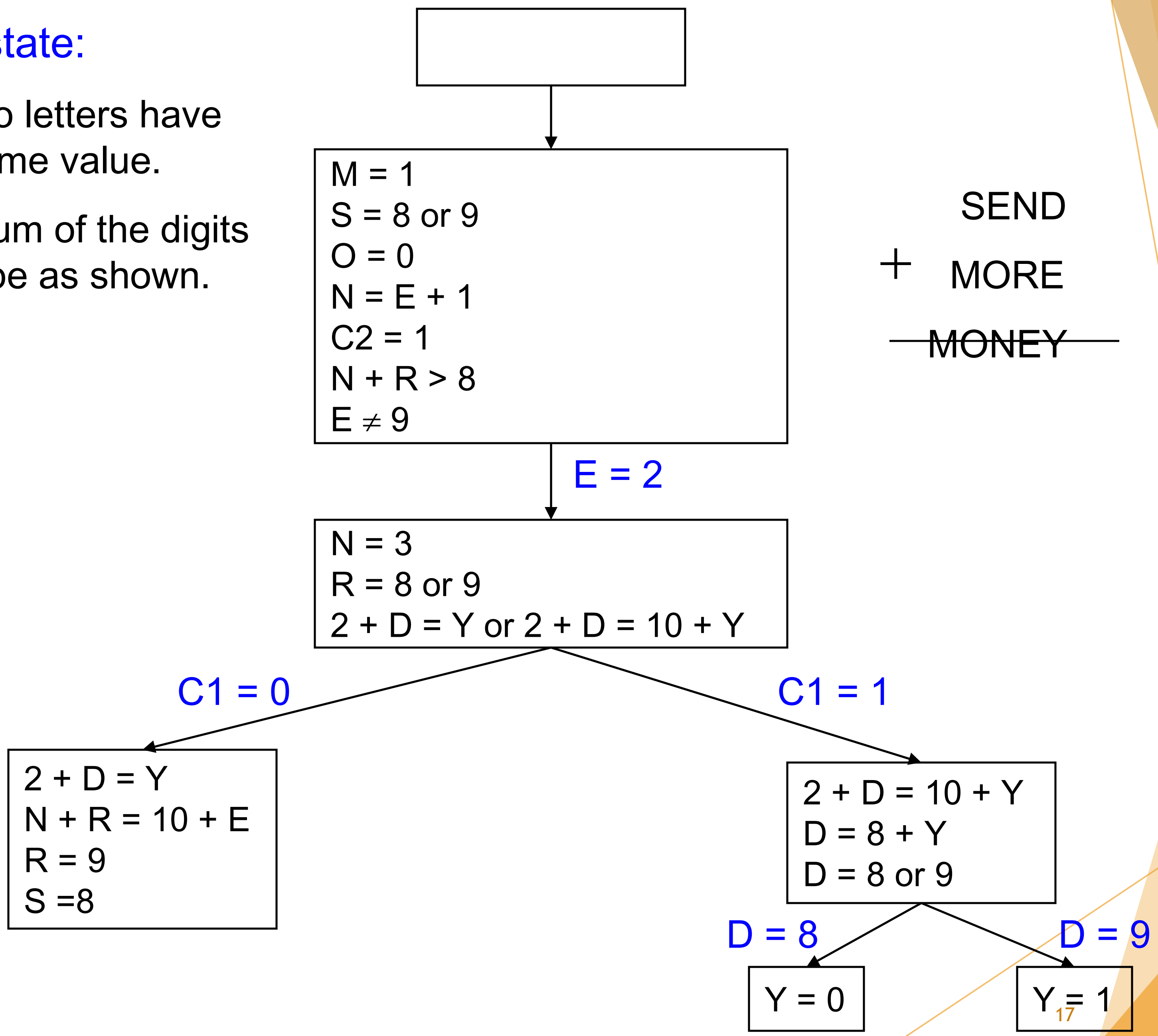
The digits left are 7, 4, 3 and 2. We know there is carry from the column D5Y, so the only pair that works is D=7 and Y= 2.

```
    9 5 6 7
 +  1 0 8 5
-------------------
 1  0 6 5 2
```

Which is final solution of the problem.

Initial state:

- No two letters have the same value.
- The sum of the digits must be as shown.



Solution:

Initial state:

- Assign values between 0 to 9.
- No two letters have the same value.
- The sum of the digits must be as shown.

$$
\begin{array}{rccccc}
 & & & S & E & N & D \\
+ & & & M & O & R & E \\
\hline
 & & M & O & N & E & Y
\end{array}
$$

$$
\begin{array}{ccccc}
 & & 1 & 1 & \\
 & \boxed{9} & \boxed{5} & \boxed{6} & \boxed{7} \\
 & \boxed{1} & \boxed{0} & \boxed{8} & \boxed{5} \\
\hdashline
\boxed{1} & \boxed{0} & \boxed{6} & \boxed{5} & \boxed{2}
\end{array}
$$

S	9
E	5
N	6
D	7
M	1
O	0
R	8
Y	2

3.Best-first Search Algorithm (Greedy Search)

- Step 1: Place the starting node into the OPEN list.
- Step 2: If the OPEN list is empty, Stop and return failure.
- Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
- Step 4: Expand the node n, and generate the successors of node n.
- Step 5: Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node,then return success and terminate the search, else proceed to Step 6.
- Step 6: For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- Step 7: Return to Step 2.

Advantages

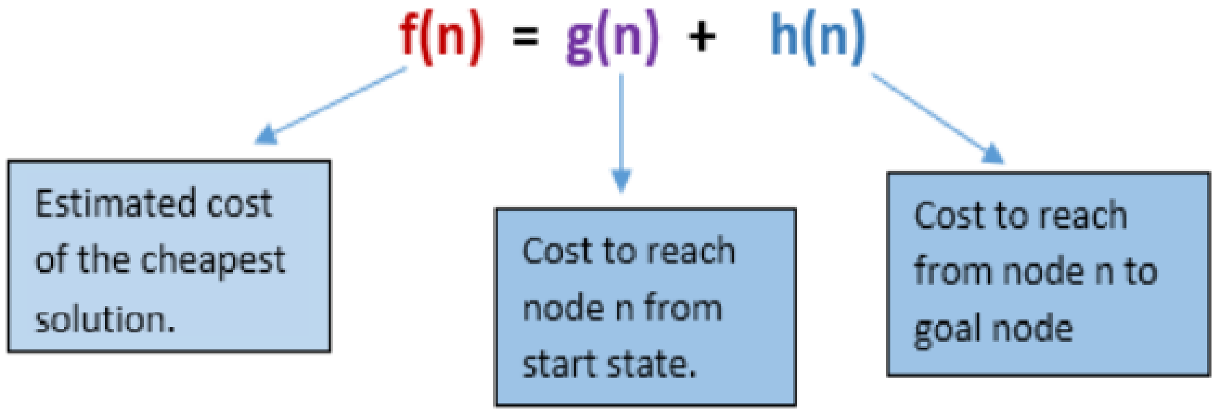
- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.

Disadvantages

- It can behave as an unguided depth-first search in the worst case scenario.
- This algorithm is not optimal.

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).
- 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)


f(n) = g(n) + h(n)
Estimated cost of the cheapest solution.
Cost to reach node n from start state.
Cost to reach from node n to goal node

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

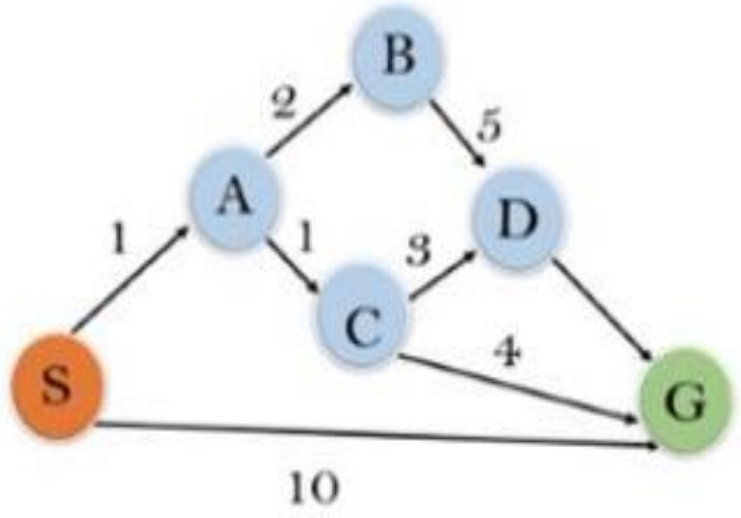
Advantages

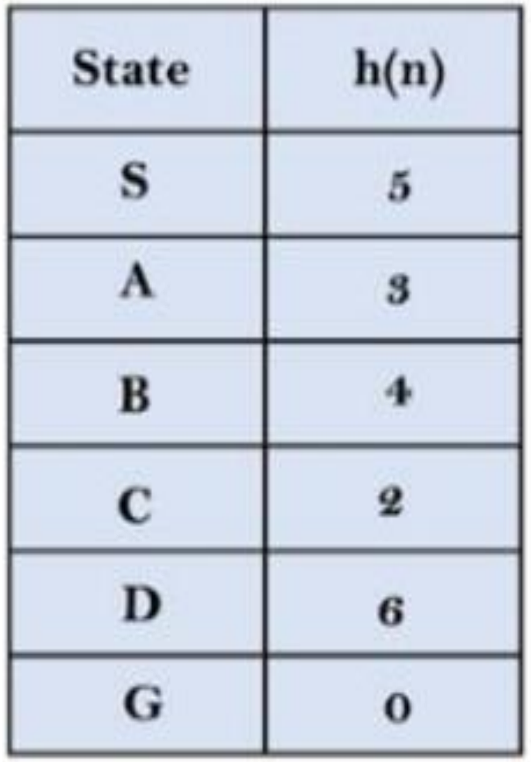
- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

Disadvantages

- It does not always produce the shortest path as it mostly based on heuristics and approximation.
- A* search algorithm has some complexity issues.
- 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.

Example: Find the Path from S to G using A* Algorithm.

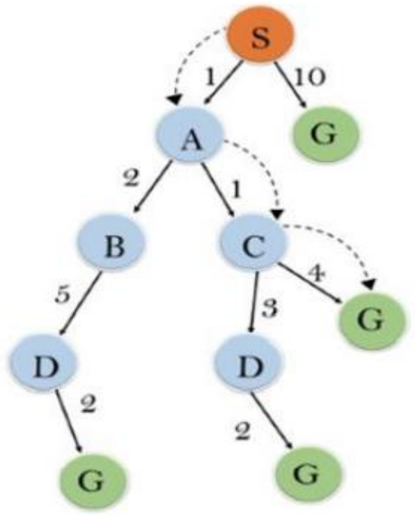




State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

- Initialization: {(S, 5)}
- Iteration1: {(S--> A, 4), (S-->G, 10)}
- Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}
- Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S- -> A-->B, 7), (S-->G, 10)}
- Iteration 4 will give the final result, as S--->A--->C--->G it provides the optimal path with cost 6

Solution



4.Problem Reduction (AND - OR graphs - AO * Algorithm)

- 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. **That is called is Problem Reduction.**
- The decomposition of the problem or problem reduction generates AND arcs. One AND are may point to any number of successor nodes.

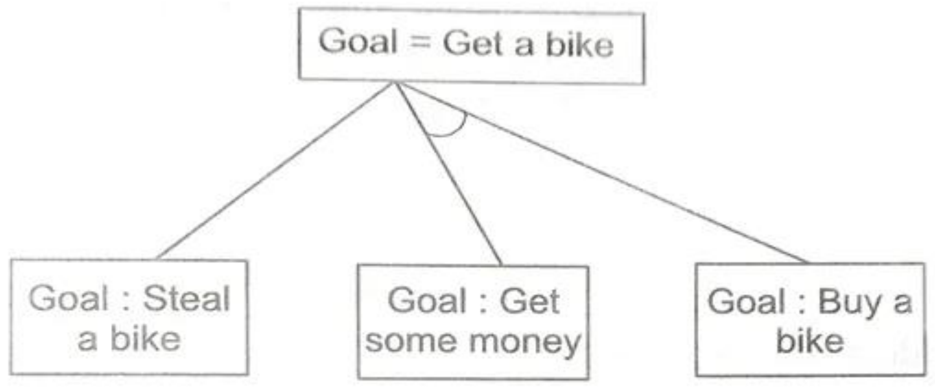


Fig: AND / OR Graph

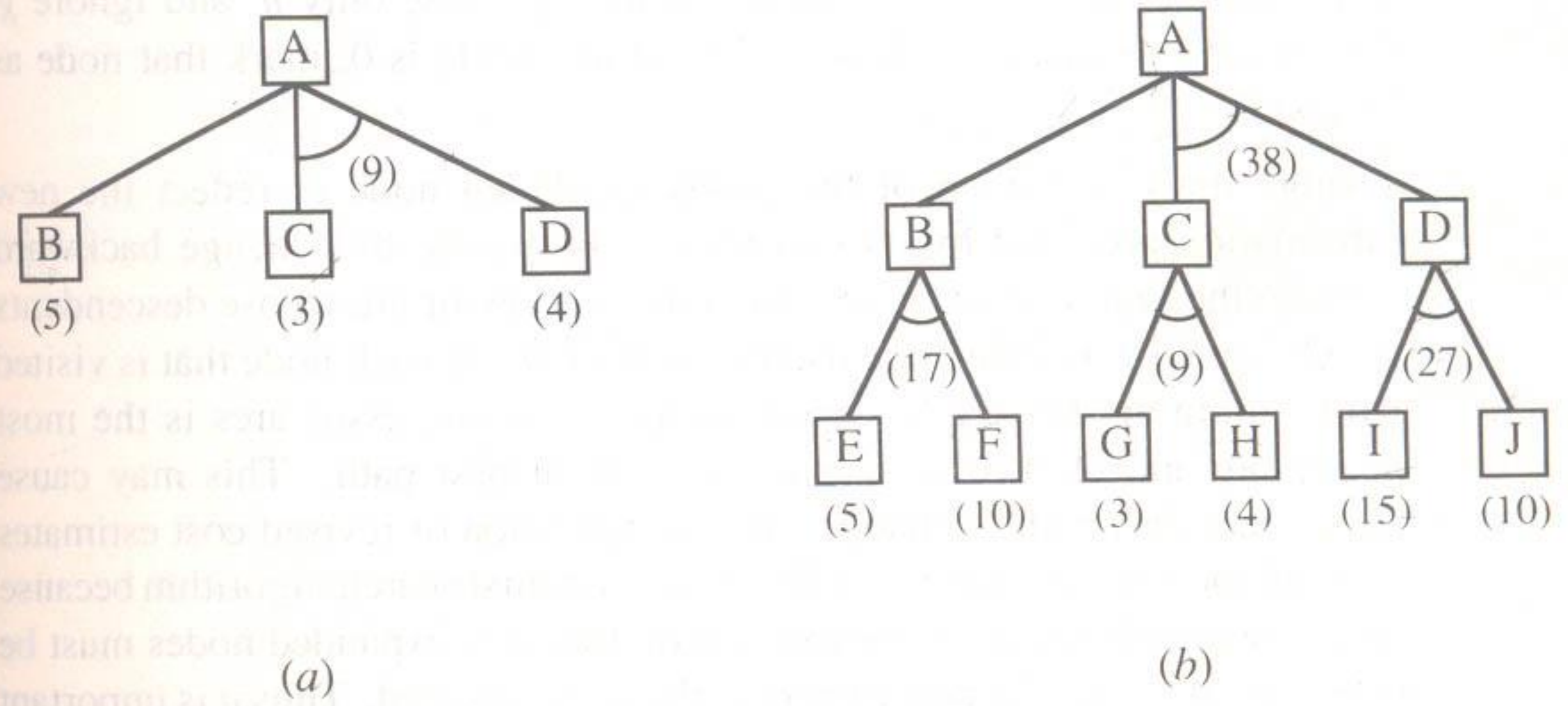


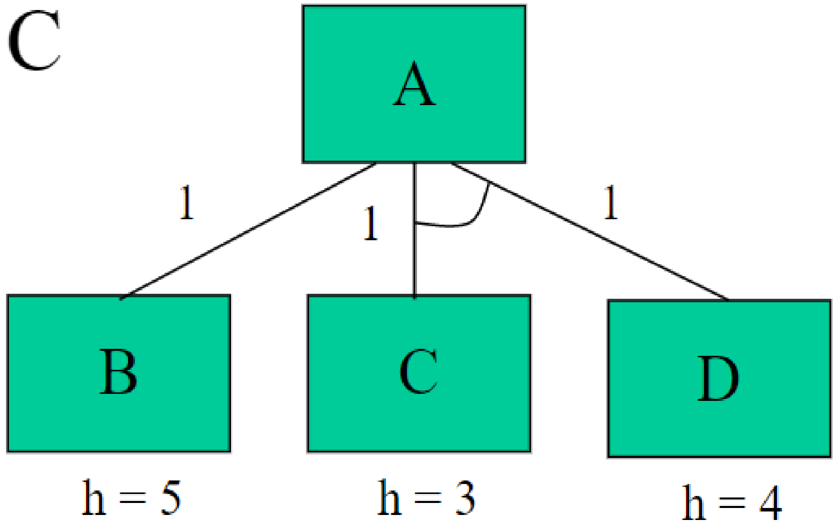
Figure 3.7: AND-OR Graphs

- How are AND-OR graphs searched with a best-first strategy?
  - A* algorithm is not suitable
  - Should we choose C to expand next?

No, because total cost along C would be 3+4+1+1=9;

Along B, 5+1=6

B is the best choice.



Algorithm

Step-1: Create an initial graph with a single node (start node).
Step-2: Transverse the graph following the current path, accumulating node that has not yet been expanded or solved.
Step-3: Select any of these nodes and explore it. If it has no successors then call this value-FUTILITY else calculate f'(n) for each of the successors.
Step-4: If **f'(n)=0**, then mark the node as **SOLVED**.
Step-5: Change the value of f'(n) for the newly created node to reflect its successors by backpropagation.
Step-6: Whenever possible use the most promising routes, If a node is marked as SOLVED then mark the parent node as SOLVED.
Step-7: If the starting node is SOLVED or value is greater than **FUTILITY** then stop else repeat from Step-2.

5.Hill Climbing

Hill climbing algorithm is a technique which is used for optimizing the mathematical problems.

- 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

1. 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.

2. Greedy approach

• Hill-climbing algorithm search moves in the direction which optimizes the cost.

3. No backtracking

• It does not backtrack the search space, as it does not remember the previous states

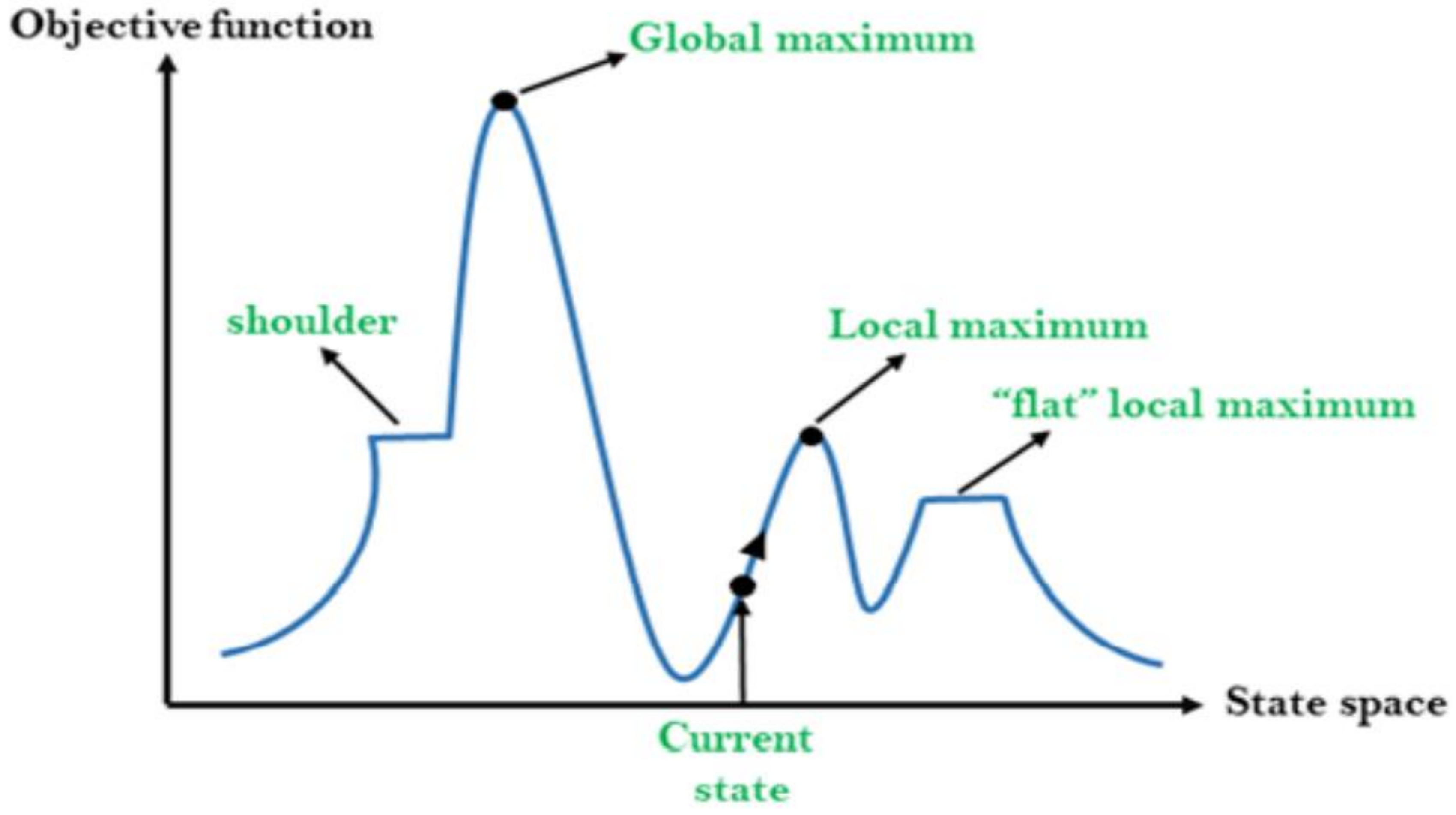
State Space Diagram for Hill Climbing

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.

1. **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.
2. **Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.
3. **Current state:** It is a state in a landscape diagram where an agent is currently present.
4. **Flat local maximum:** It is a flat space in the landscape where all the neighbor states of current states have the same value


Objective function
Global maximum
shoulder
Local maximum
"flat" local maximum
State space
Current state

Types of Hill Climbing Algorithm

1. Simple hill Climbing
2. Steepest-Ascent 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 optimal solution and the solution is not guaranteed
- Less time consuming

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:

- If it is goal state, then return success and quit.
- Else if it is better than the current state then assign new state as a current state.
- 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

1.Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.

2. Loop until a solution is found or the current state does not change.

¤ Let SUCC be a state such that any successor of the current state will be better than it.

¤ For each operator that applies to the current state:

a. Apply the new operator and generate a new state.
b. Evaluate the new state.
c. If it is goal state, then return it and quit, else compare it to the SUCC.
d. If it is better than SUCC, then set new state as SUCC.
e. If the SUCC is better than the current state, then set current state to SUCC.

3. Exit.

Simulated Annealing

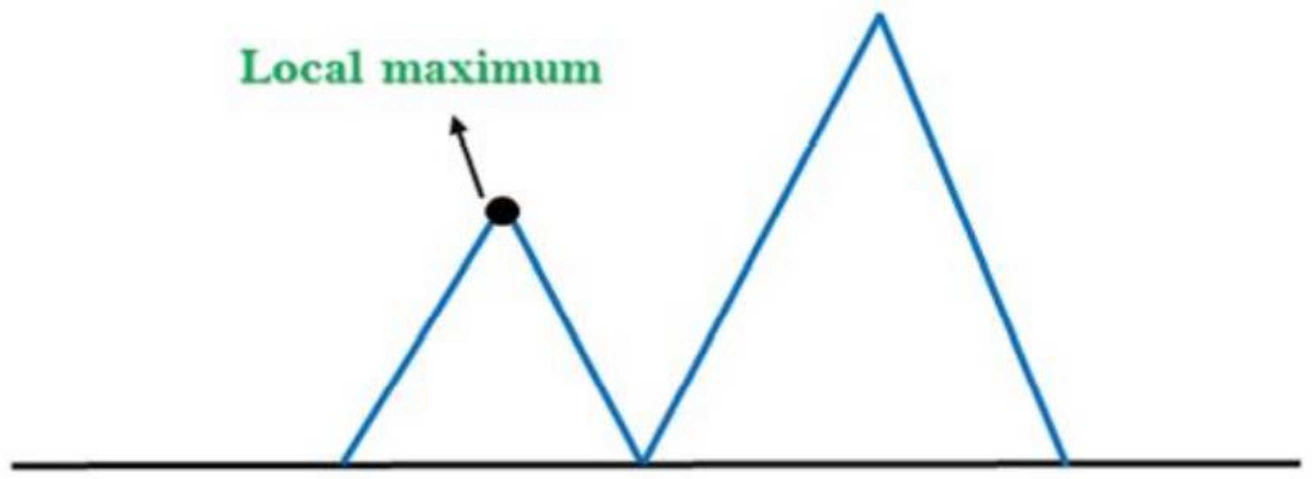
A hill-climbing algorithm which never makes a move towards a lower value guaranteed to be incomplete because it can get stuck on a local maximum.

- 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.

Problems in Hill Climbing Algorithm

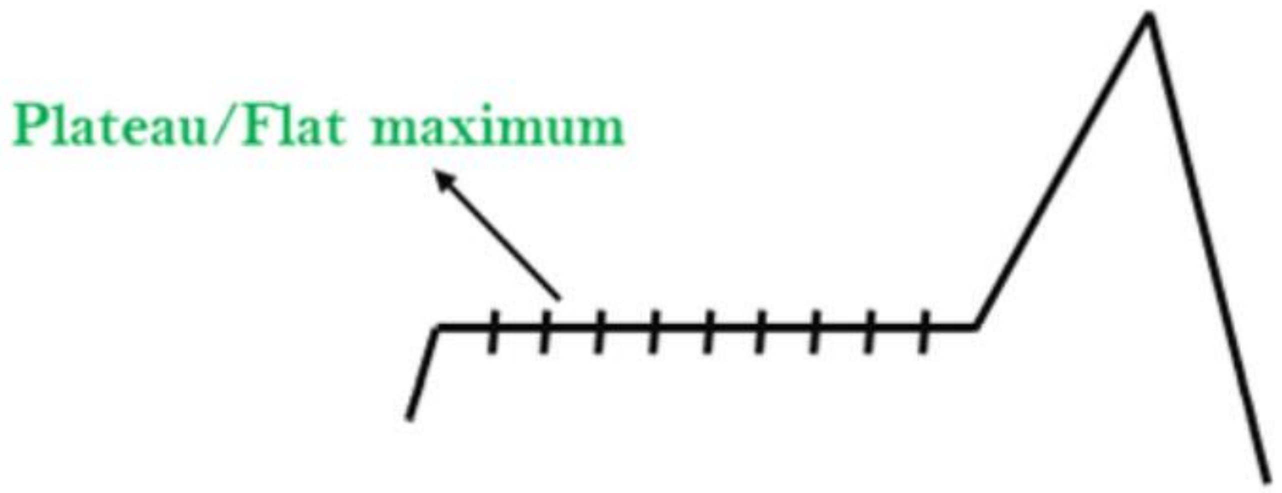
1. Local Maximum

- 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.



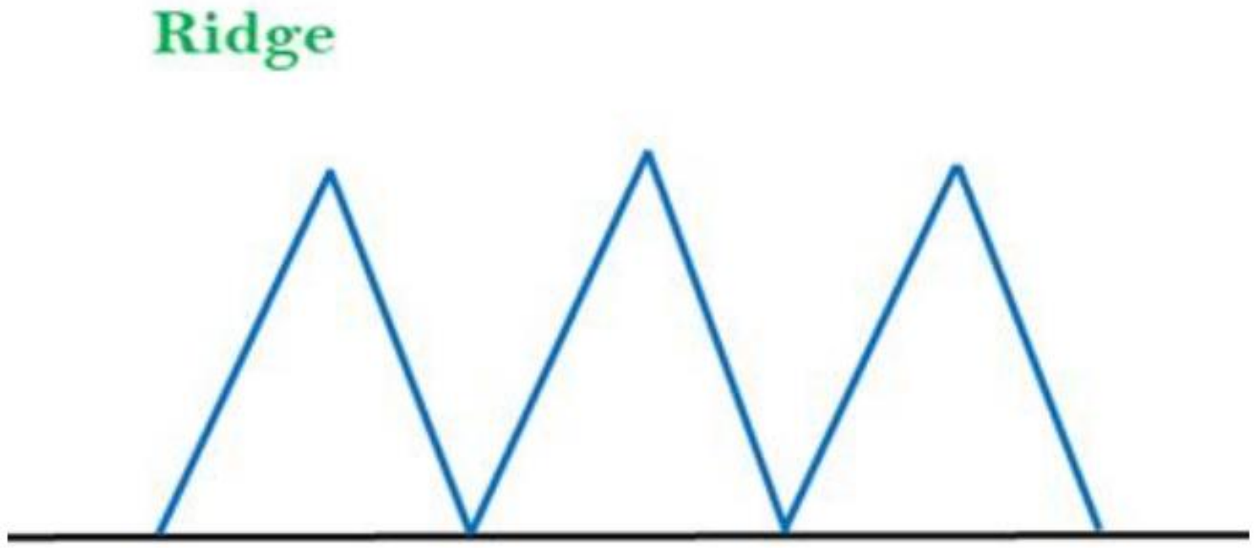
2. Plateau

- 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



3. Ridges

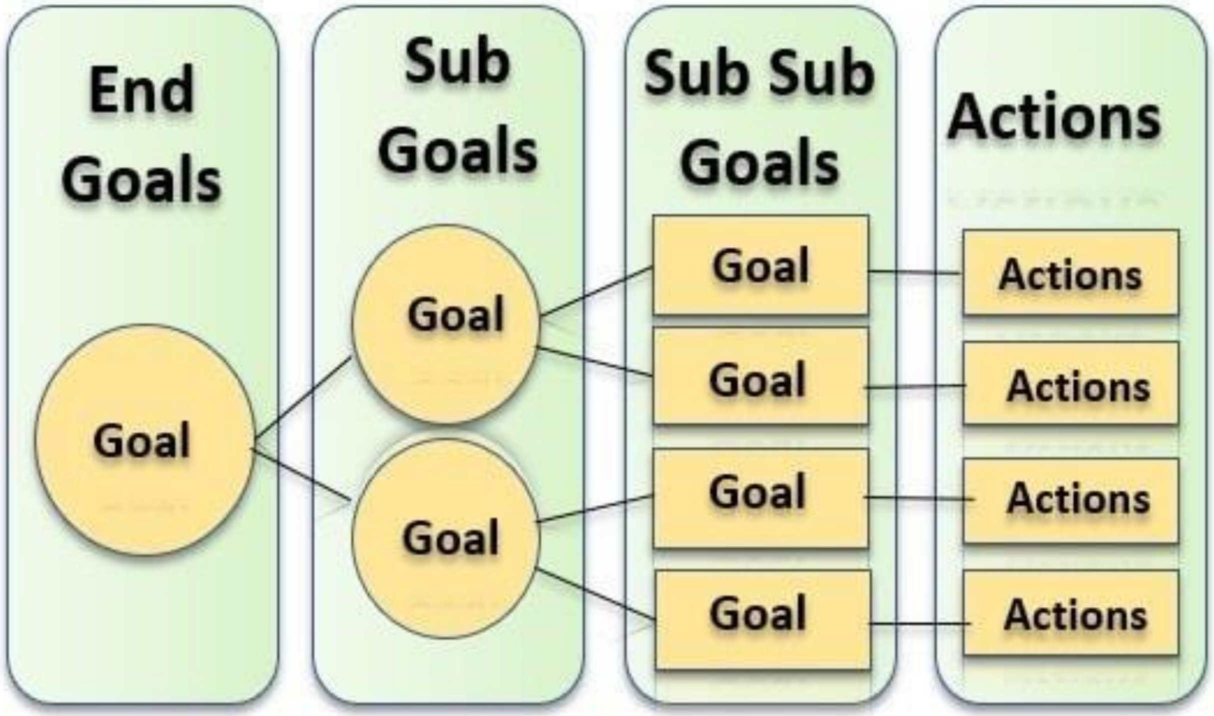
- A ridge is a special form of the local maximum.
- It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move



6. Means-Ends Analysis

- 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 technique was first introduced in 1961 by Allen Newell, and Herbert A. Simon in their problem-solving computer program, which was named as General Problem Solver (GPS).
- The MEA analysis process centered on the evaluation of the difference between the current state and goal state.

The target goal is divided into sub-goals, that are then linked with executable actions.



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.

1. First, evaluate the difference between Initial State and final State.
2. Select the various operators which can be applied for each difference.
3. Apply the operator at each difference, which reduces the difference between the current state and goal state.

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.

a) Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.

b) 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.

c) If

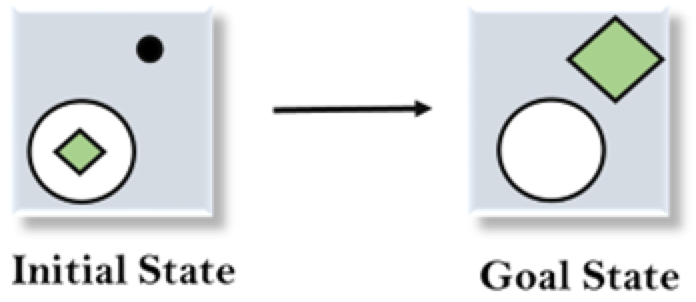
(First-Part <------ MEA (CURRENT, O-START)

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.

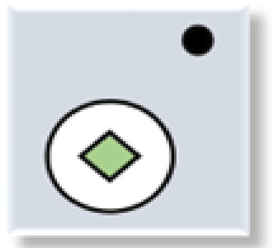


Solution:

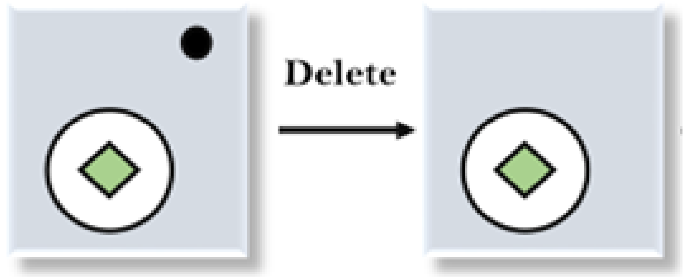
To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

- **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.

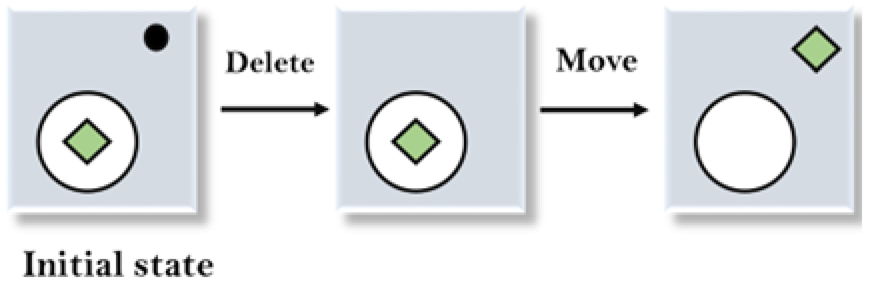

Initial state

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.

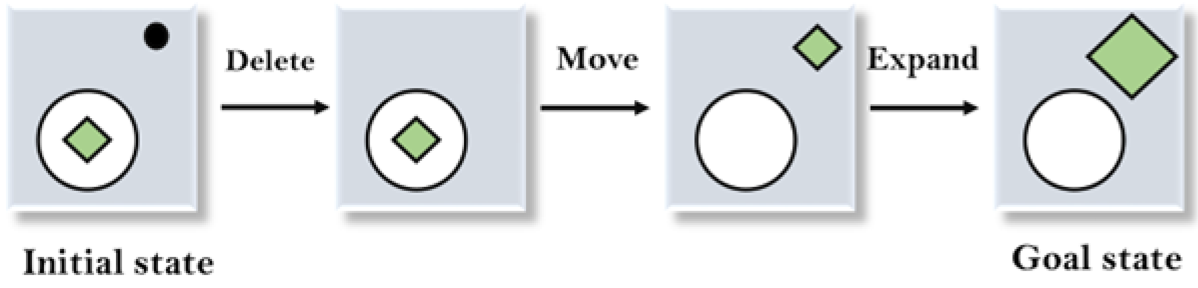


Initial state

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.



References

(Chapter 3 – AI Rich & Knight)