CHAPTER – 3 HEURISTIC SEARCH **Techniques**

Dr. Goutam Sarker,
CSE, NITD

HEURISTIC SEARCH TECHNIQUES

- If we want to solve a problem within a finite time, (e.g. if the problem solving system is connected with a real time or online systems) we require an efficient and effective control strategy or search technique.

Heuristic (or Informed) Search

A Heuristic is a search technique that improves the efficiency of a search process, sometimes possibly by sacrificing claims of complete search

An Example of Heuristic Application – TSP

A salesman has to start from a particular city and after the visiting all the cities, he has to come back to the starting city. We are to perform a planning for the salesman so that he is visiting all the cities in the shortest route.

A MAP OF A TSP

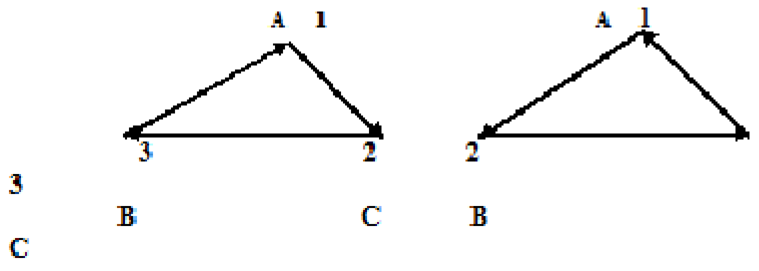


Fig. 1 Map for traveling salesman's problem

Travelling Salesman Problem contd..

1. If there are two cities then there is obviously a single path.
2. If there are there cities then there are two paths as shown in the above map of traveling salesman‟s problem.
3. Similarly we can verify that if there are four cities then there are 6 **[!(4-1)]** different paths.

4 Continuing in this fashion we can verify through induction that if there are “n” different cities then there are **!(n-1)** different paths.

TIME TO FIND OUT THE SHORTEST PATH

Now assume that the time required to examine a particular path is proportional to the total number of cities in that path.

So the time required to examine all the paths (to find the shortest path) is proportional to **n * ! (n- 1) = ! n.**

Now if n is small then the number of paths (**= ! (n-1)**) as well as the time required to examine all those paths (**! n**) are small and manageable.

If n is too large, time taken to examine all the paths to find out the shortest path is also extremely large.

So we should have to find out an alternative mechanism (heuristic) to find out the shortest path within the affordable time.

HEURISTIC - REVIVED

A heuristic is an intelligent searching method that improves the time efficiency or promptitude of a search process to find goal(s), (which sometimes may be inexact/ non- optimal one) from information in which region of the search space the goal(s) is/are lying.

TSP Heuristics Branch and Bound

1. Generate several paths.
2. Find the shortest path out of them.
3. Now give up exploring any path if its partial length is greater than the shortest path ever found.
4. On the other hand if its partial length is shorter than the current shortest path, then update the shortest path by the new one.

So we have to explore some undesirable nodes or paths before getting the shortest one.

TSP Heuristic - Nearest Neighbor

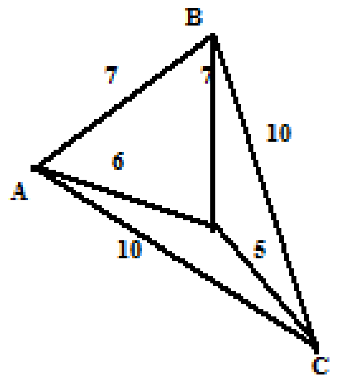
Input: A Map of a Traveling Salesman‟s Problem.

Output: Shortest Route for the Traveling Salesman.

Steps:

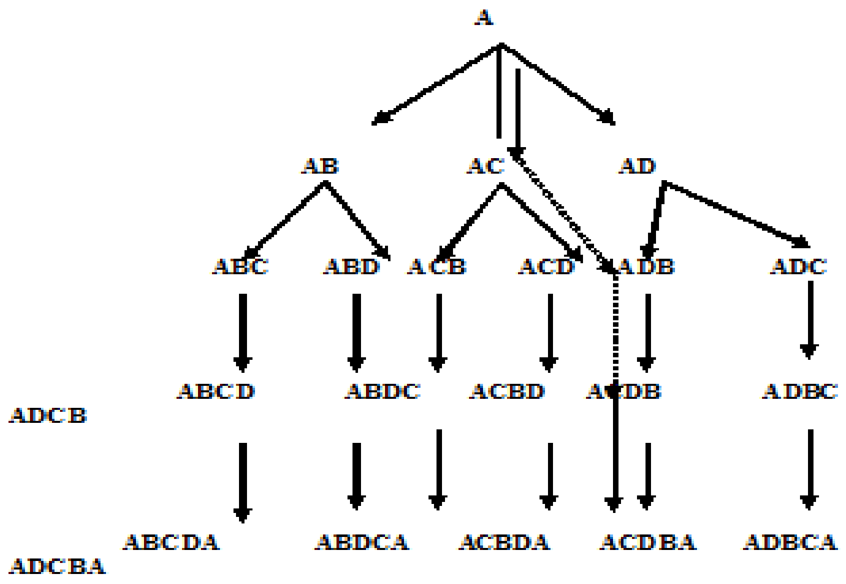
1. Randomly choose a city to start up.
2. Select the city nearest to the present city, which has not yet been visited. Visit that city.
3. Repeat step 2 until all cities have been visited.

Map of One Travelling Salesman Problem with 4 cities



THE BOLD LINE IN FIG. 3 OF THE NEXT SLIDE SHOWS THE DESIRED SHORTEST ROUTE WITH NEAREST NEIGHBOR ALGORITHM.

Fig. 3 State Space Representation of the TSP with Nearest Neighbor Heuristic



UTILITY OF APPLICATION OF HEURISTIC:

If the search is an uninformed or blind one, we would become hopelessly entrapped in a combinatorial explosion of useless nodes.

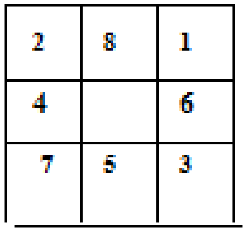
With heuristic, obviously we cannot claim that the search is complete and thus the goal is possibly the inexact/ non optimal rather than the best one.

RARELY DO WE ACTUALLY NEED THE BEST SOLUTION. WE ASSUME THAT THE GOAL PRODUCED BY HEURISTIC (WHICH IS NOT THE BEST BUT AN APPROXIMATE ONE) WILL SERVE OUR PURPOSE IN THE PRESENT CASE.

SOME EXAMPLES OF HEURISTIC SEARCH PROBLEMS

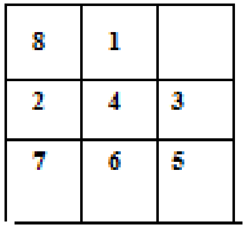
a) The Game of Eight Puzzle: By the application of suitable heuristic we would be able to directly land onto the goal state without wandering off in the irrelevant directions.

Fig. - One Eight Puzzle PROBLEM



2	8	1
4		6
7	5	3

Initial state



8	1	
2	4	3
7	6	5

Goal State

Heuristic vs. Heuristic Function

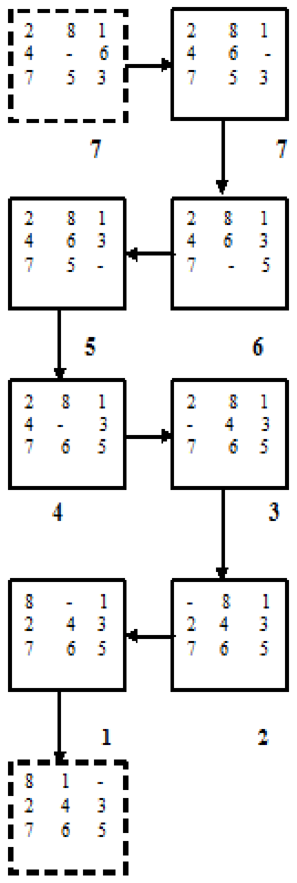
- Heuristic function (also called evaluation function) measures / evaluates how far the present state is away from the desired goal state.
- Heuristic is the informed / intelligent search technique

In this problem, we may think of two different type of heuristic /evaluation functions to evaluate the current state:

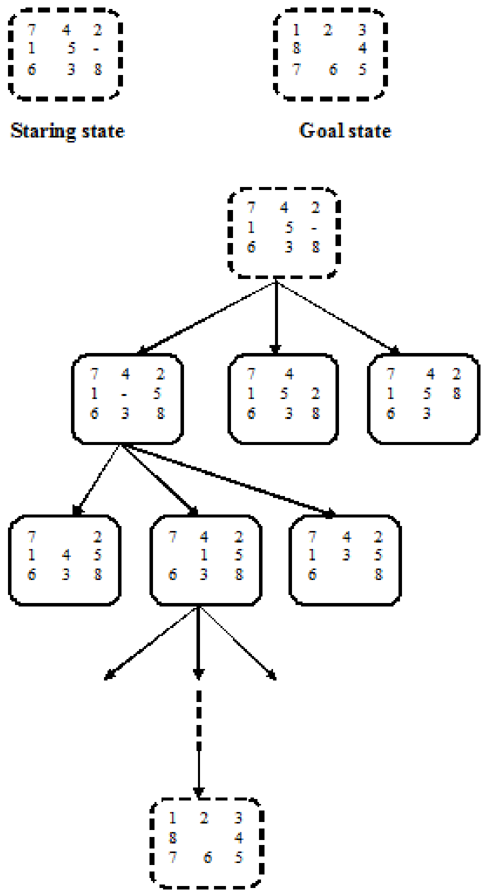
1. The total number of misplaced tiles:
2. The total number of properly placed tiles:

WITH THE ABOVE HEURISTIC AND HEURISTIC FUNCTION, THE PATH TRAVERSED BY THE SEARCH IS AS SHOWN IN THE NEXT SLIDE

PATH TRAVERSED DUE TO THE APPLICATION OF HEURISTIC AND HEURISTIC FUNCTION



VARIOUS POSSIBLE MOVES OF „EIGHT PUZZLE‟ FOR A FIXED INITIAL GOAL STATE



PROBLEM CHARACTERIZATION

It may so happen that one particular type of heuristic is having the best performance for one specific type of problem while the same heuristic produces worst performance for another type of problem.

PROBLEM CHARACTERIZATION CONTD..

1. **Is it Decomposable?** : The whole problem is logically divisible or decomposable into a set of smaller sub problems or not.

2. **Is it Ignorable / Recoverable / Irrecoverable?** : The solution steps of the problem may be ignored or undone, recovered or not to move back to the initial state.

3. **Is it Predictable / Un- predictable?** : The universe of the problem is predictable or un predictable.
4. **Is it Absolute / Relative?** The good solution of the problem is absolute or relative to other possible solutions.
5. **Is it State / Path?** The solution of the problem is a „goal state" or a „path to the goal state form the initial state".

6. **What is the Amount of Knowledge for attaining the goal? :** The role or the amount of knowledge while performing the solution of the problem.

7. . **Is it Interactive or Non Interactive Problem? :** Is intermediate interaction essential for the goal or not.

EXAMPLES

1. Decomposable Problem:

$\int (x^2 + 3x + \sin^2 x \cos^2 x)\, dx$

We can solve the problem by breaking it into three different smaller sub problems and solving them

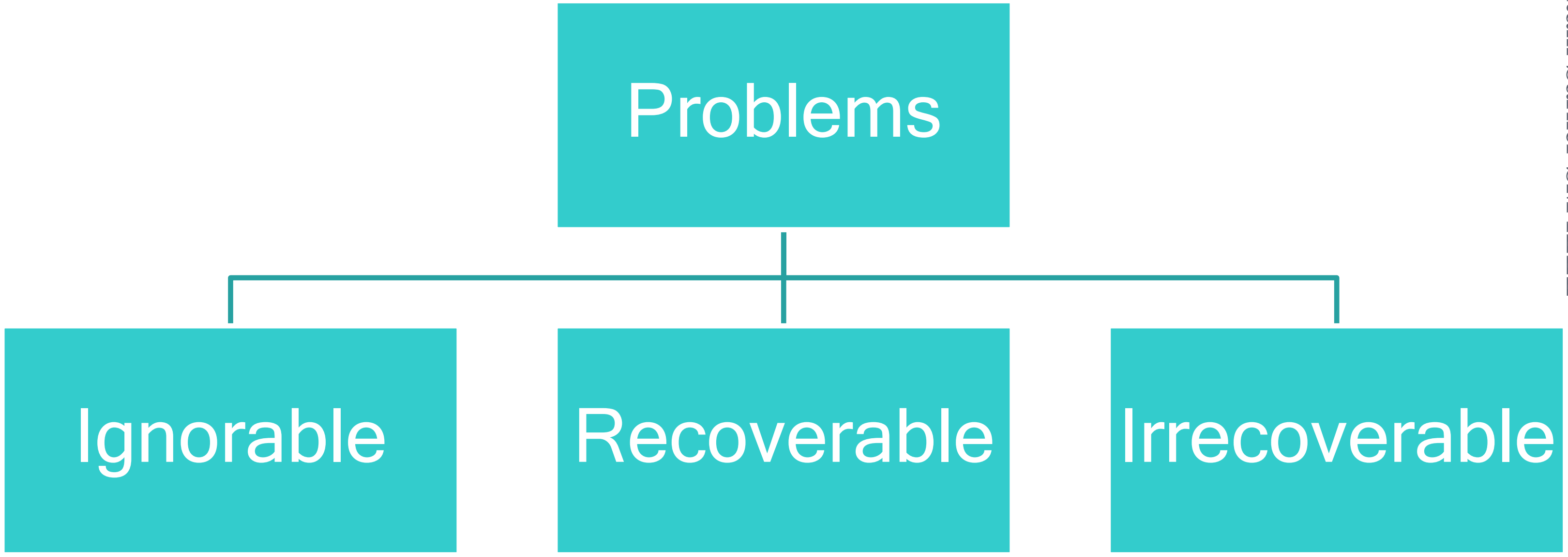
2.Problems for which solution steps are Ignorable/Recoverable/ Irrecoverable

(i) Ignorable: Example of this type of problems is Theorem Proving.

(II) RECOVERABLE: EXAMPLE OF THIS TYPE OF PROBLEM IS THE GAME OF EIGHT PUZZLE, RUBIK CUBE:

(iii) Irrecoverable: An example of this category is the game of Chess:

Problems



3.The Universe is not predictable:

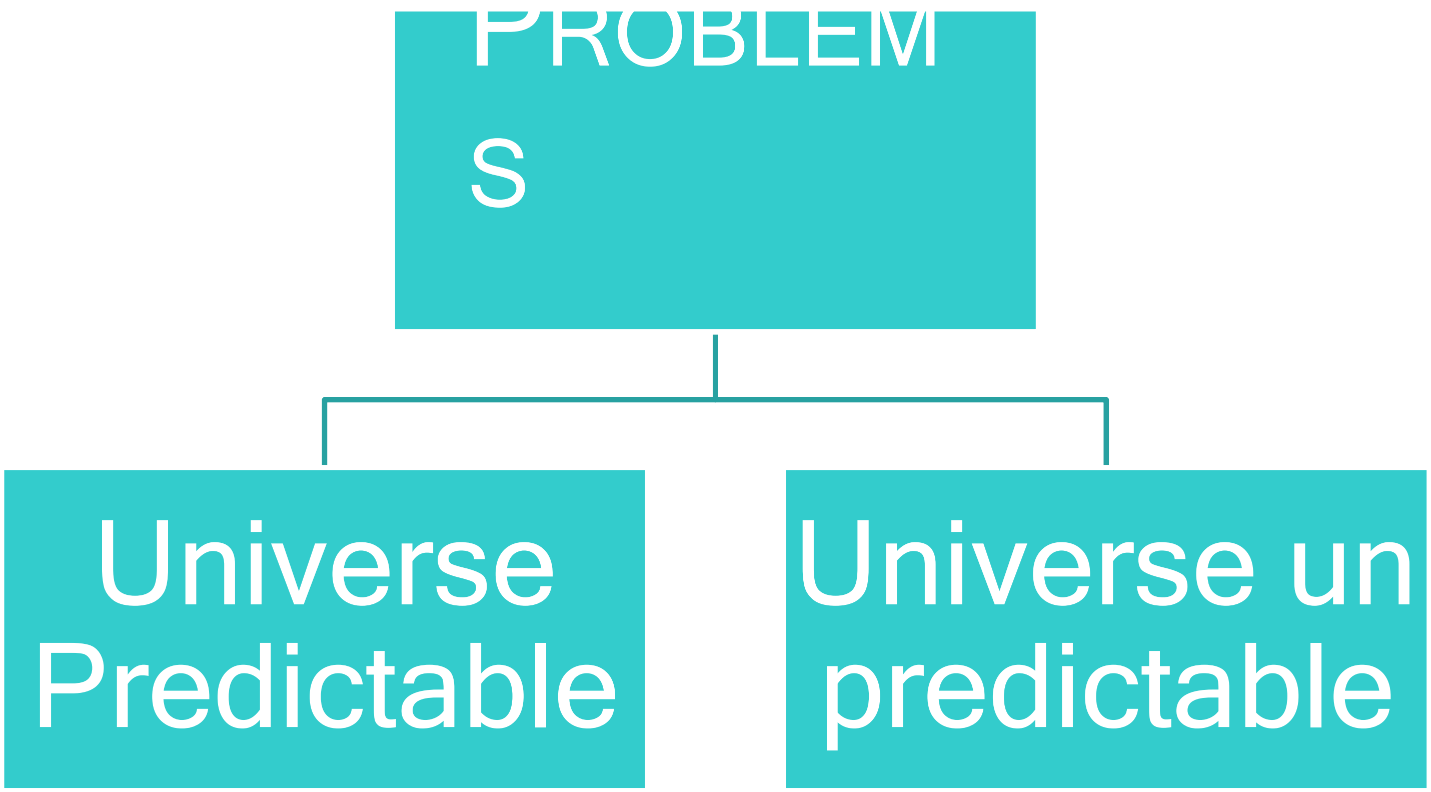
3.The Universe is not predictable:

(i) **Predictable Problems:**

Consider the game of 8 Puzzle, or Rubik Cube. Here we can perform the proper plan for the entire sequence of moves and with that planning we are quite ensured about the landing on the goal state.

(ii) Unpredictable Problems:

Consider the chess- playing program. For a particular board position, we don‟t specifically know what exactly would be the opponent‟s next move.

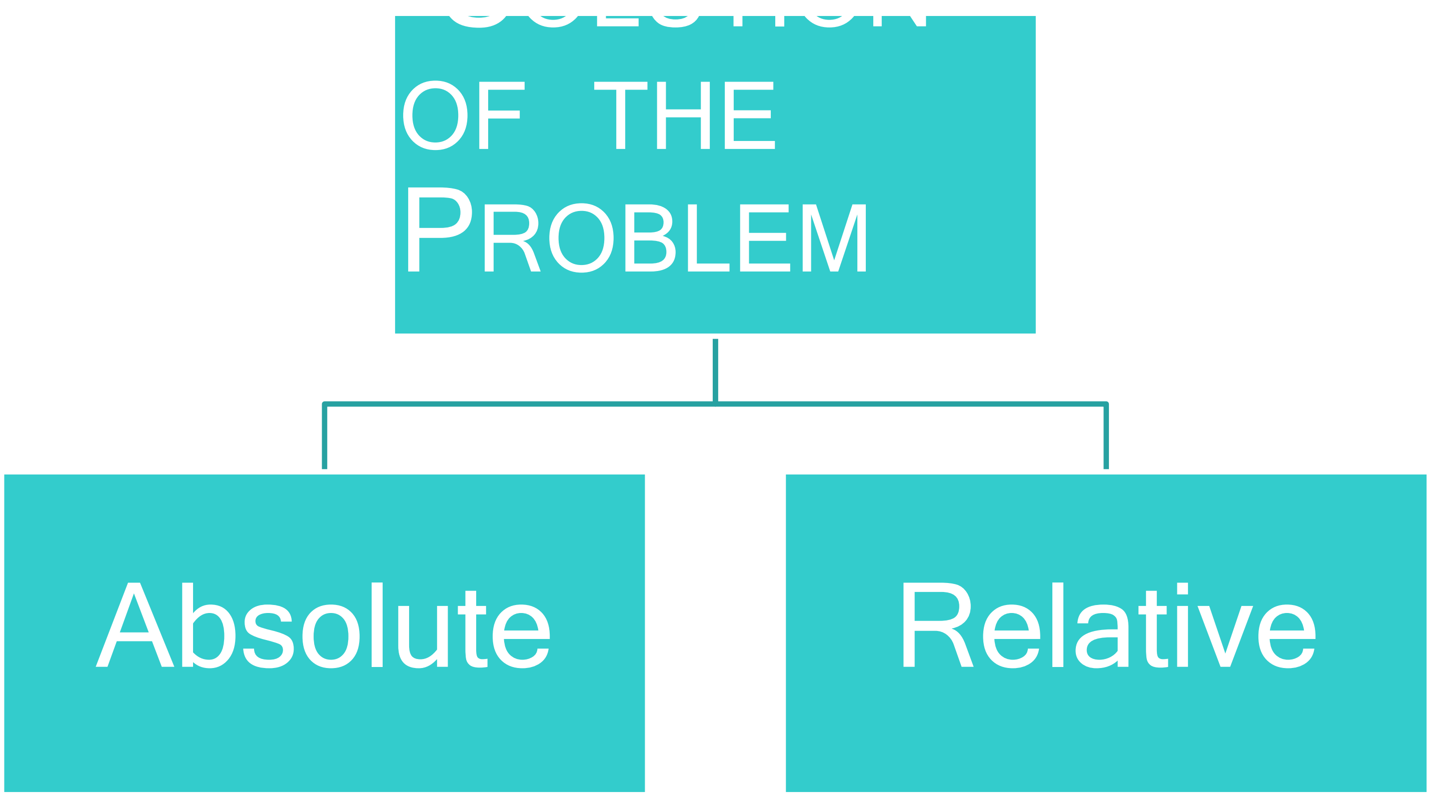

PROBLEM
S
Universe Predictable
Universe un predictable

4. SOLUTION TO THE PROBLEM IS ABSOLUTE OR RELATIVE:

or Relative

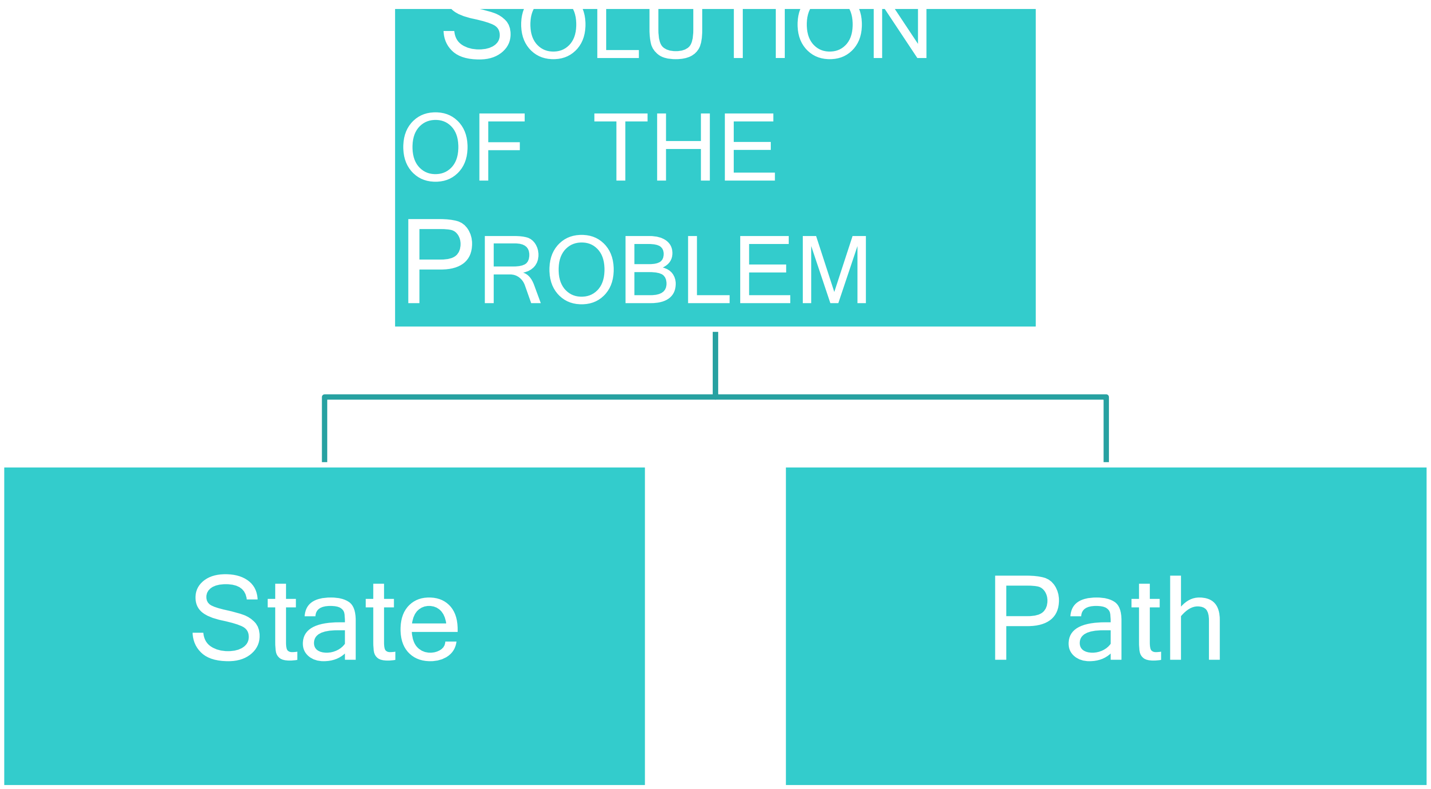
In database or knowledge base query, once a solution is found, that is sufficient and we need not have to compare it with any other possible solution.

In traveling salesman‟s problem, there are several solutions and we are to find the shortest path solution.


OF THE PROBLEM
Absolute
Relative

5. Solution is State solution or a Path Solution:

Consider the sentence “They are flying kites”. The main issue is what path it followed to derive the sentence, on which the interpretation of the sentence will depend.


SOLUTION OF THE PROBLEM
State
Path

6. Role of Knowledge to perform the Solution:

In card game or chess playing, we only need to know the rules to determine legal moves and control mechanism for appropriate search procedure.

Contrary to this consider any newspaper or story understanding. Here a vast amount of prior associated knowledge is very essential even to be able to recognize the solution i.e. to interpret a single sentence of the story.

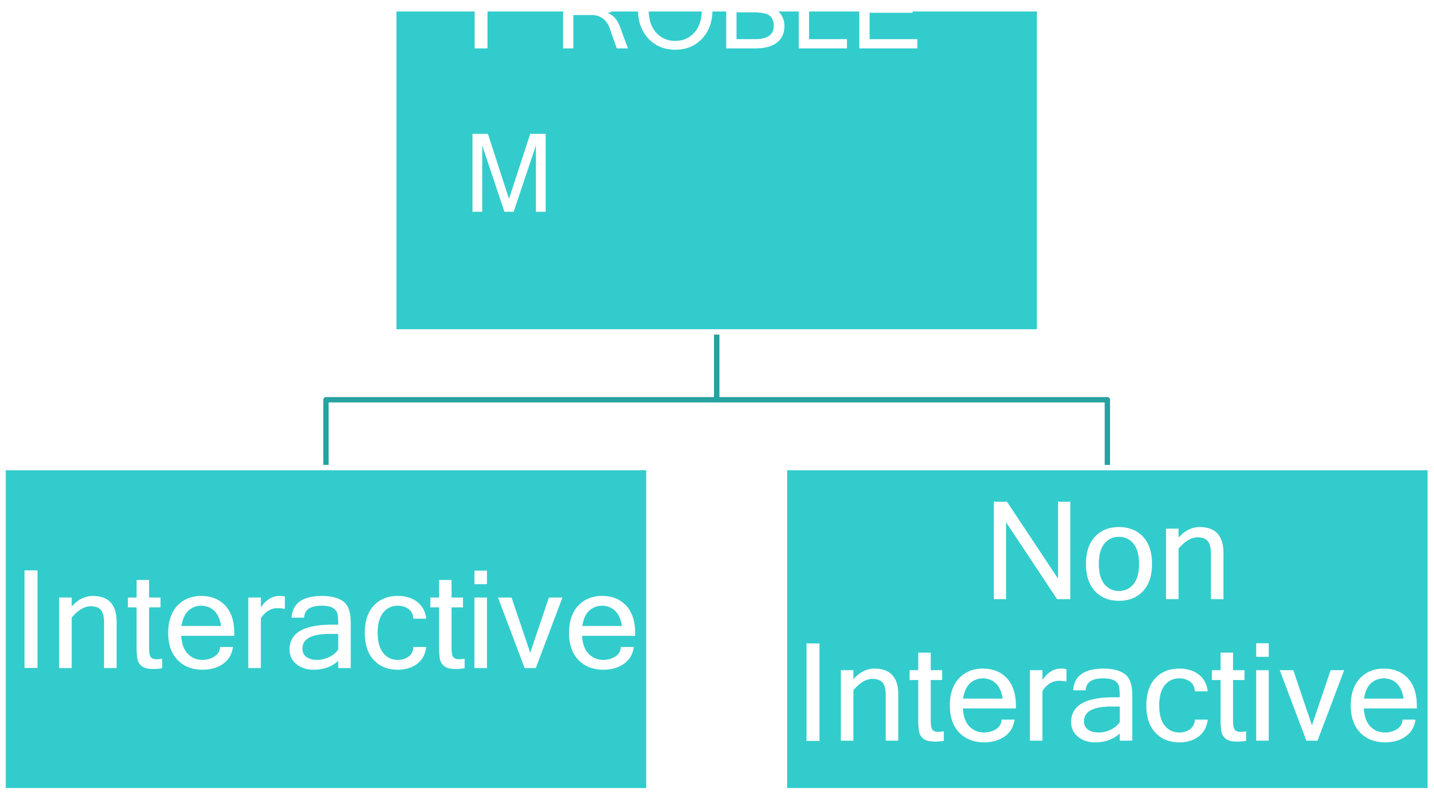
7. Interactive vs. Non interactive Problem:

7. Interactive vs. Non interactive Problem:

(1)Non interactive Problems: where the system performs solutions or answers corresponding to query without any intermediate human interaction.

Example : Database query

(2)Interactive Problems: Where intermediate interaction is essential for feedback to AI system or supply additional information to the user to come to the solution or answer to the problem. Example: Medical Expert System.


PROBLE
M
Interactive
Non Interactive

HEURISTIC SEARCH TECHNIQUES

- **Generate and Test.**
- **Hill Climbing:**
  - **(i) Simple Hill Climbing**
  - **(ii) Steepest Ascent Hill Climbing**
- **Best First Search and A* Heuristics.**
- **Problem Reduction**
  - **(i) AND OR GRAPH**
  - **(ii) AO * Heuristic**
- **Means End Analysis.**

GENERATE AND TEST

Algorithm:

Input: State Space Representation of the Problem.

Output: The goal state „g".

Steps:

1. Generate a possible solution.
2. Test if this is the actual solution.
3. If yes then quit with the solution, else move back to step 1.

Generate and Test contd..

Because of its simplicity, this heuristic is also applicable to simplest type of problem i.e. Ignorable Problems without any complicacy of backtracking.

HILL CLIMBING:

Algorithm:

Input: The state space representation of the problem.

Output: The goal node „g"

Steps:

1. Examine the initial state. If it is a goal state, then report the goal and quit.

Otherwise continue with the new assignment: initial sate = current state.

2. Continue in the loop until a solution is found or until no other new rule is there to be applied to the current state.

Hill Climbing Algo. Contd..

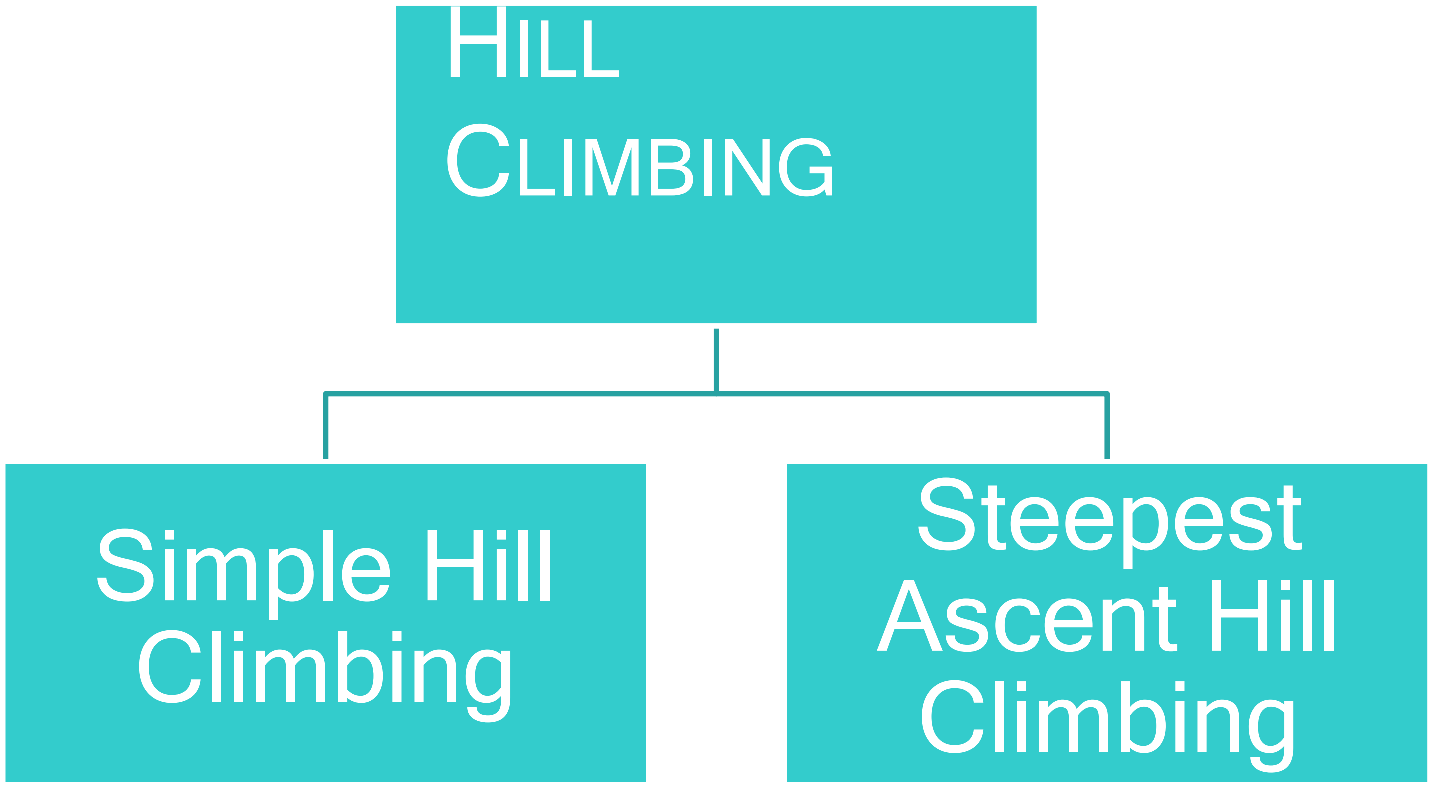
i) Select an untried operator. Produce a new state.

ii) Evaluate the new state:

iii) If it is a goal state, then return it and quit.

iv) If it is not a goal state, but is better than current state, then perform the assignment: "new state = current state".

v) If it is equally good or inferior to the current state, then continue in the loop i.e. switch back to state (2).


HILL CLIMBING
Simple Hill Climbing
Steepest Ascent Hill Climbing

Steepest Ascent Hill Climbing:

Steepest Ascent Hill Climbing:

A major variation of Simple Hill Climbing is achievable if all possible rules are applied in the current state and the best one is selected as the next state.

Both Simple Hill Climbing and Steepest Ascent Hill Climbing is applicable to irrecoverable problems (8puzzle, rubic cube etc.) because the algorithm does not permit to move to any state lying at any upper level, although it may be the best one at that instant out of all terminal nodes.

The Game of Rubik Cube

There are four different sub cubes each of which is having four different colors, namely red, green, yellow and blue on its four sides. These sub cubes may be moved in any direction by a step of ninety degrees.

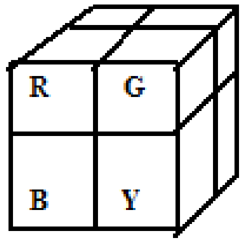
Problem: Get the same color in each face of the cube

Heuristic : Simple or Steepest Ascent Hill Climbing

Heuristic /evaluation Function :The sum of the total number of differences of colors on each face is the suitable heuristic function. This has to be minimized to zero.

The Look of a Rubik Cube

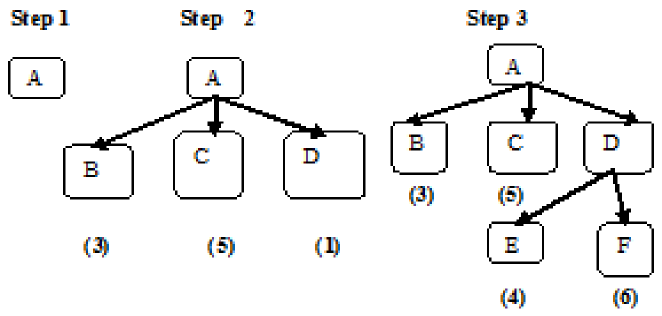
1)



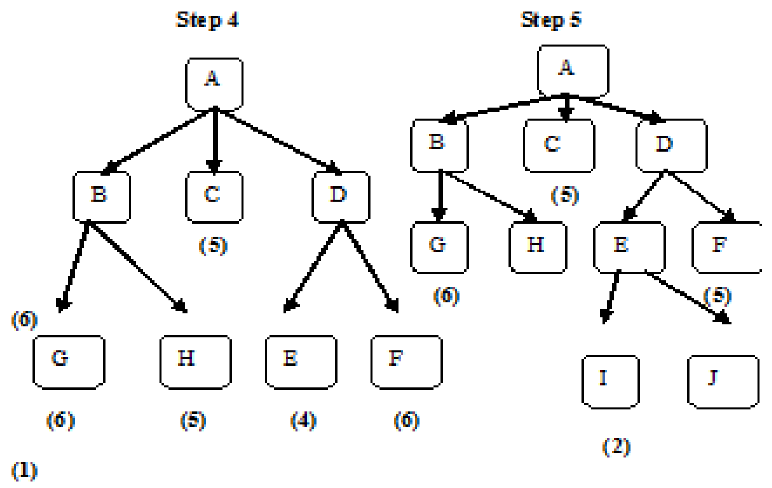
Best First Search and A * Heuristics

- While the Steepest Ascent Hill Climbing restricts itself to only the terminal nodes in the current level only, Best First Search selects the best node by considering all unexplored node lying at any level either previous or current.
- So obviously Best First Search heuristic is mostly suited for recoverable problems.

Moves in a Best First Search



Moves in a Best First Search contd..



- The Best First Search compares all the terminal nodes at different levels to find out the most promising one.
- If two or more nodes are reflecting the same heuristic function measure then Best First Search has to explore the nodes of equal evaluation sequentially from left to right.

Comparison between Hill Climbing and Best First Search

- In both types of Hill Climbing, once a move is deselected at a particular level, it is never reconsidered afterwards although then it might be the most promising move at present. But in Best First Search the previously rejected nodes at lower level may also be reconsidered

- Simple Hill Climbing and its Steepest Ascent version will terminate when the successor states are inferior than the current state. But Best First Search will not stop there. It will continue with the most promising node selected out of all the terminal nodes at that step.

A VARIATION OF BEST FIRST SEARCH – A* HEURISTIC

In the previous Best First Search, the most promising or best node is considered on how far the present node is away from the goal. So here, the Evaluation Function computes, how far the nodes are away from the goal.

There is another variation of Best First Search. Here the best node is having the shortest path length (or cost) from the root node up to the goal node via the current node

This variation of Best First Search is called A * heuristic

The A* Heuristic uses

f‟ = g+h‟

where

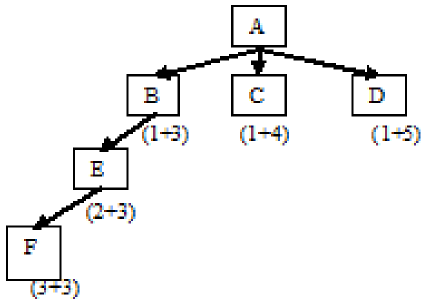
g = path length in terms of number of steps or otherwise or the total cost of movement form initial state to the current state which is going to be evaluated with this function.

h‟ = estimate of the additional cost or additional path length in terms of number of steps or otherwise of moving from the current node to the desired goal state.

A Few Discussions on A * Heuristic

- Both Ordinary Best First Search and A* Heuristics are Basically Best First Search only - only differing in Evaluation Function
- To derive a path involving the smallest number of steps, we set 1 as the level of the arcs connecting one state to another.
- On the contrary, to find the cheapest path solution, we attach the cost of moving from one state to another to the label of the arc connecting one state to another and find out the cheapest path.

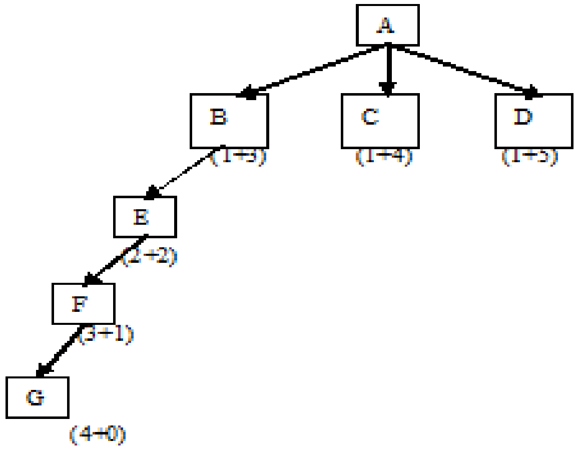
CASE I: THE VALUE OF H‟ AT A PARTICULAR NODE UNDERESTIMATES THE ACTUAL VALUE OF H AT THAT NODE: H‟(B) < H(B).



H"(D) < H(D)

At node F (3+3) we are to switch back to pervious level node C (4+1). This is due to the underestimation of the value of h”. So the conclusion is that by underestimating h(B) we have wasted a lot of our efforts.

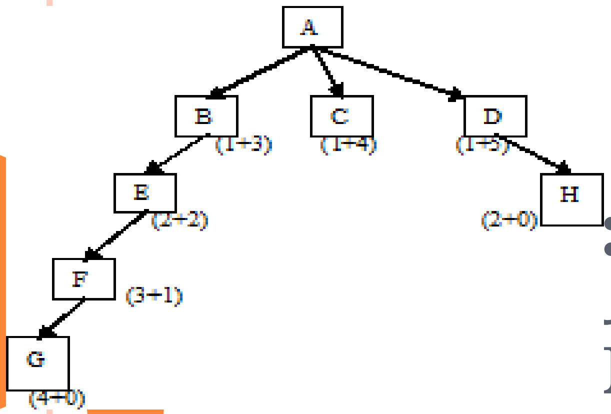
THE VALUE OF H" AT A PARTICULAR NODE IS THE EXACT VALUE OF H AT THAT NODE: H"(B) = H(B).



H"(D) = H(D)

Since there is no approximation in the h" value of B and B is the most promising node having the heuristic value of (3+1) the A* heuristic will converge rapidly towards the goal without any search.

h‟(D) > h(D)



: THE VALUE OF H‟ AT
JLAR NODE
MATES THE ACTUAL
VALUE H AT THAT NODE:

H"(D) > H(D)

Thus f h"(D) overestimates h(D) [actual value of h at the node D] , cheapest path or shortest length solution is not ensured, if the entire graph is not examined.

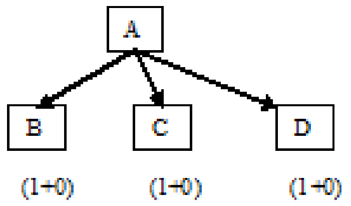
- Both overestimation and underestimation in the value of h has the adverse effect.
- Overestimation cannot guarantee the shortest length or cheapest path solution
- Underestimation leads to wastage of a lot of effort during searching.

- One solution of this problem of overestimation and underestimation might be solved by „don‟t estimate anything‟.

  i.e. f” = g + h‟ = g + 0 = g.

- So we are again back to the Breadth First Search which is nothing but one kind of unguided or routine search. We may allow this search but this is once again an inefficient search.

Setting h”=0 in the heuristic function of A* leads to Breadth First Search



PROBLEM REDUCTION:

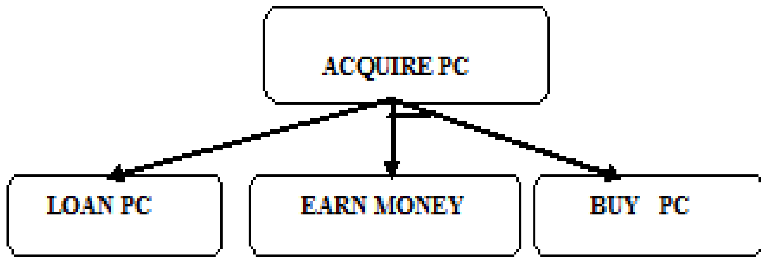
- The problem reduction heuristics we are now going to discuss are only applicable to decomposable type of problems
- There are two different forms of Problem Reduction namely

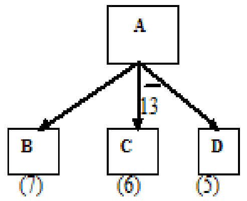
1. **AND OR GRAPH**
2. **AO * Heuristic.**

AND OR GRAPH

- The problem here is to acquire a PC. We can get it in two different ways: either we can loan a PC or we can earn money and buy a PC.

SIMPLE EXAMPLE OF AND OR GRAPH

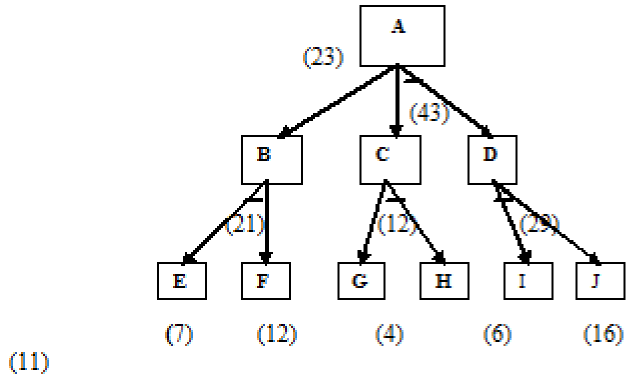




- Consider the AND OR Graph as shown.

  One level expansion of AND OR GRAPH of a problem is shown

SECOND LEVEL EXPANSION OF AND OR GRAPH OF A PROBLEM



In the above figure we should not expand G(4) but rather examine either E(7) or F(12). This is because of the fact that node G(4) is not the part of the current best path since the evaluation of A through the path containing node G is becoming equal to 43. Also the evaluation of A through path with the node E(7) or F(12) is 23. So the next node to be expanded is either E(7) and F(12) although with f" = g+h they are inferior than G(4).

Thus the general rule is that, the selection of which node to expand next must depend not only on f „ (= g + h‟) value of that node but also on whether that node is a part of the current best path form the initial node.

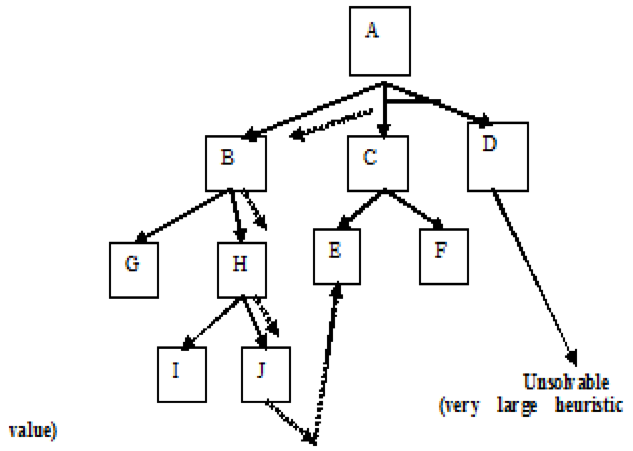
THE AO * HEURISTIC:

If the example is really a graph there is one problem with the heuristic function f „= g + h‟. In this g is the number of steps or cost of moving from the initial state to the current state. But in a graph, there may be many paths to the same state. So we cannot get a unique value of g for a particular node.

One way to solve the problem is to set g = 0. Thus, we will not store even a single value of g. In this case h‟(= f „) serves as the estimate of the goodness of a node.

Here the modified heuristic function f „ = h‟, this AO * heuristic is applicable for state problem or absolute problem, unlike And OR Graph which is applicable for state problem.

Application of AO *Heuristic in a problem to find the best path



- Suppose we are to solve A. To solve A, we are to solve B. To solve B we are to solve H. Similarly to solve H we are to solve J and to solve J we are to solve E. So if E is solvable then through the path E-J-H-B-A we can solve A.
- Now there is another way to solve A. To solve A, we are to solve C. To solve C we are to solve E. So the path is E-C-A. But through that path we can solve A by solving E if D is solvable, because the paths E-C-A and D-A are “AND” ed. But D is practically unsolvable.

LONGER PATH MAY SOMETIMES BE MORE PREFERABLE

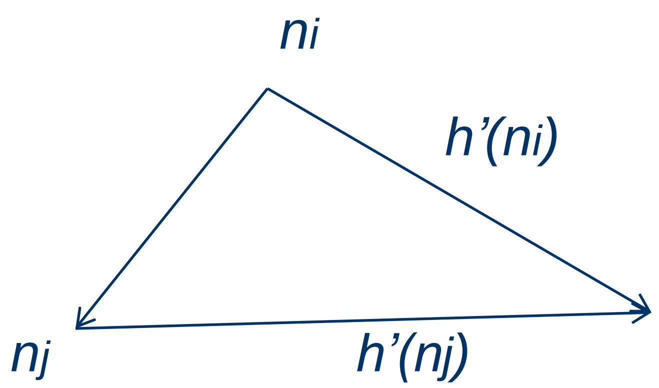
So to get a solution for A we are to solve E, but we are to move to E via the longer path A-B-H-J-E rather than the shorter path A-C-E. So the longer path in this case is better than the shorter path.

The consistency (Monotone) Problem

Consider a pair of nodes such that nj is a successor of ni. We say that h‟ obeys the consistency condition if for all such pairs in the search graph:

$$h‟(ni) - h‟(nj) \leq c(ni,nj)$$

Consistency Condition



Consistency Condition

CONTD.

.

The consistency condition states that along any path in the search graph, our estimate of the remaining optimal cost towards the goal (i.e. h‟) cannot decrease by more than the arc cost along the path.

Means End Analysis:

- Here we are dividing the original problem into series of smaller sub problems and reducing the difference between the source and destination of the new sub problems.
- The heuristic we apply here is called Means End Analysis

MEANS END ANALYSIS ALGO

Input: Start State „S", Goal State „G", and a Set of rules or operators.

Output: The start state „S" and the goal state „G" are merged together.

Steps:

1. Find an operator / rule that would mostly minimize the difference (say d1) between S and G.

ALGORITHM CONTD..

2. In case no such operator is found that could be directly applicable to the current state S, set a sub problem of getting to a state (say S'') in which it can be applied. The difference between S and S'' is say d2 where d2 < d1.

3. In case the operator does not produce exactly the goal state set a second sub problem of getting from the state (say G'') it produces the goal G. The

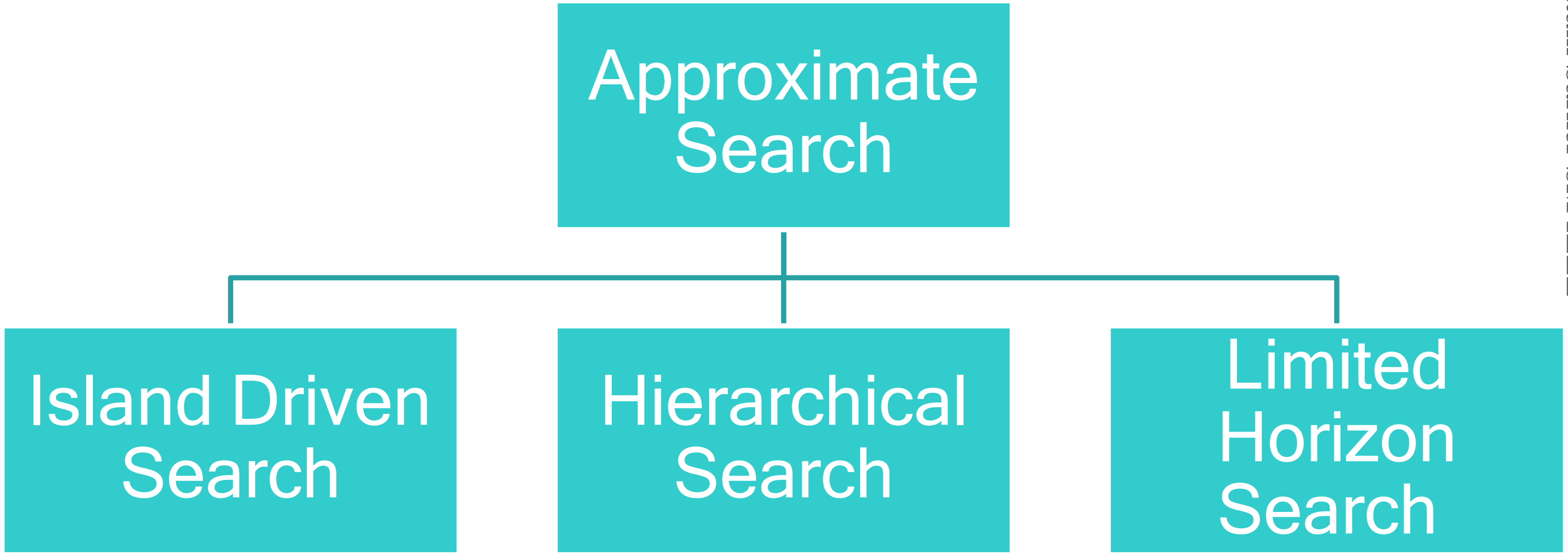
difference between G'' and G is say d3 where d3 < d1.

- The two sub problems should be easier to solve than the original problem. The Means End Analysis process can then be applied recursively.
- It is clear that the above Means End Analysis heuristic is applicable for irrecoverable, predictable, state, absolute problems.

Approximate Search

- Suboptimal / Inexact (or even not a goal) might be produced by some search processes.
- These search processes address the problem of limited computational and / or time resources.

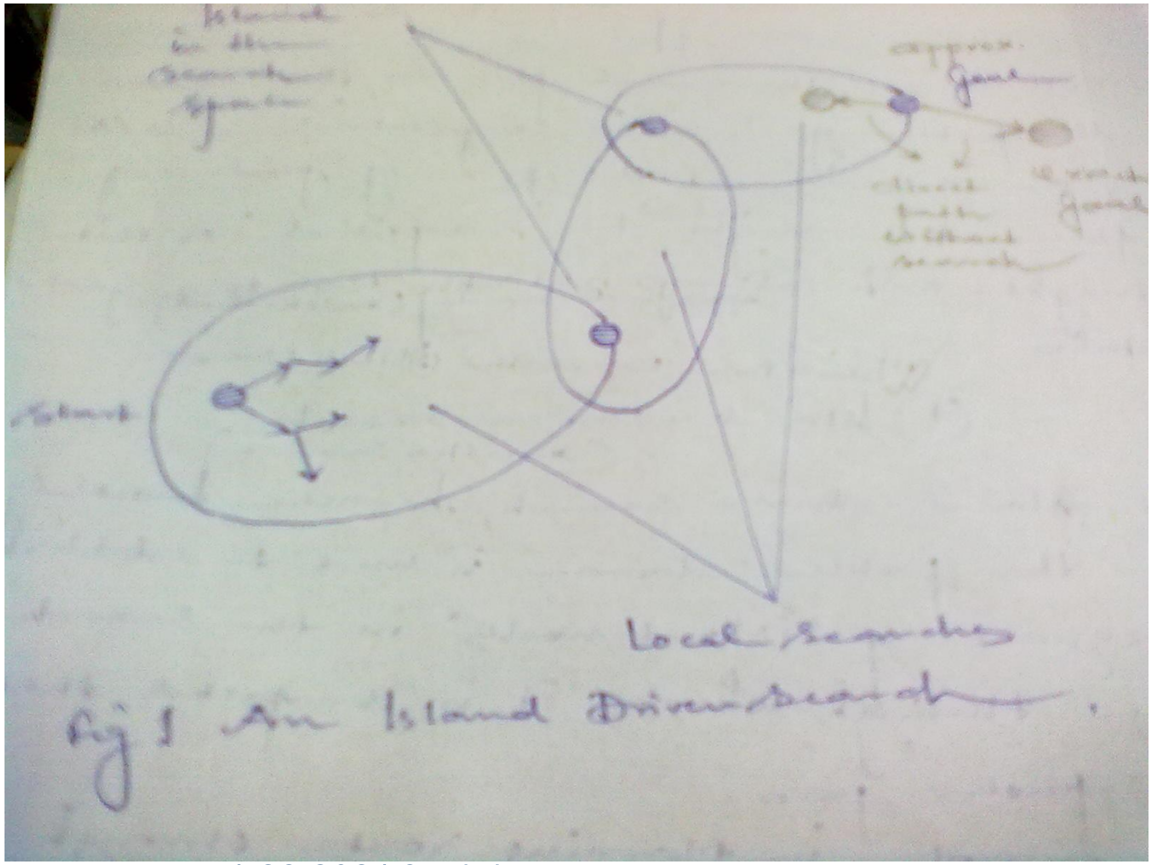
Types of Approximate Search



Island Driven Search

- In Island Driven Search, heuristic knowledge from the problem domain is used to establish a sequence of “island nodes” in the search space through which it is suspected that goal path passes.
- Suppose n0 is the start node and, ng is the goal node, and (n1,n2, … … . … . , nk) is a sequence of such islands. We initiate a heuristic search with n0 as the start node and n1 as the goal node. We start another search with n1 as the start node and n2 as the goal node, and so on until we find a path to ng.

Fig. An Island Driven Search

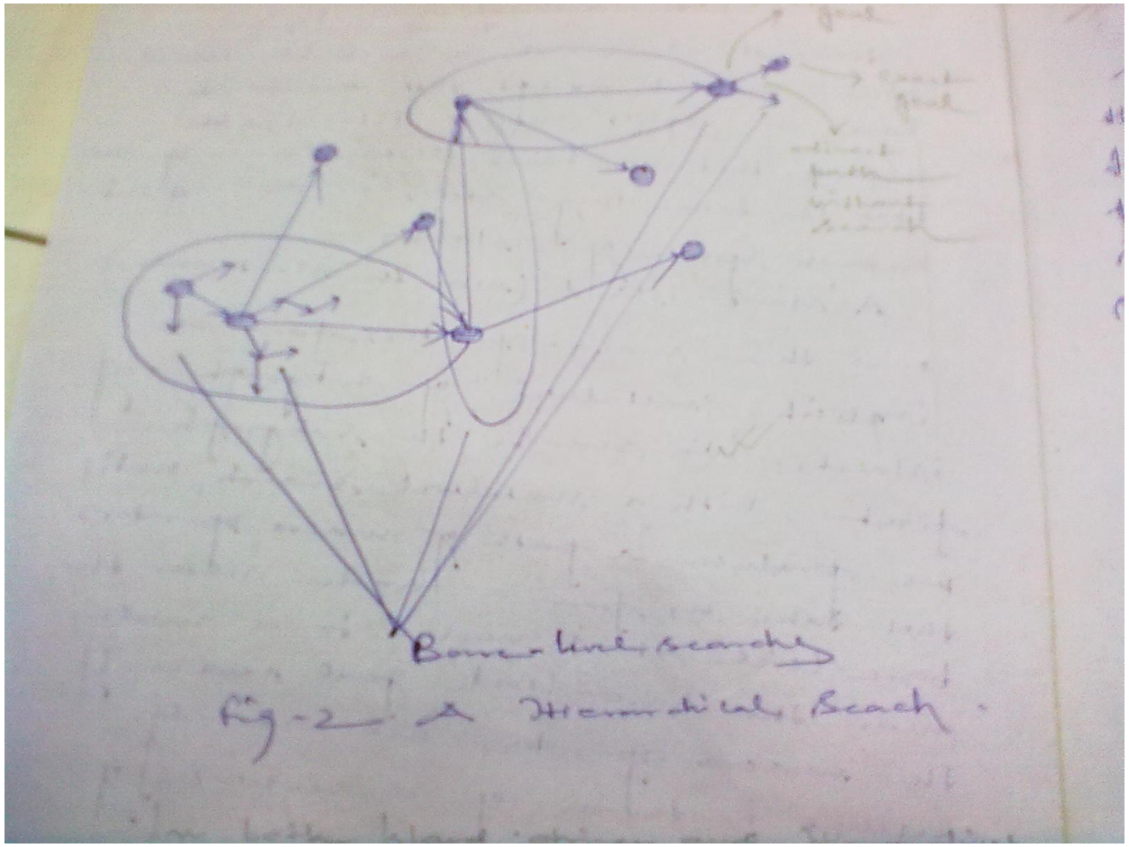


1/28/2021 3:54:47 PM

Hierarchical Search

- Hierarchical Search is much like Island Driven Search except that we do not have an explicit set of islands.
- We assume that we have certain “macro operators” that can take large steps in an implicit search space of islands.
- A start island (near the start node) and these macro operators form an implicit “meta level” of sub graph of islands. We search the sub graph first, with a meta level search, until we produce a path of macro operators that take us from a node near the base level start node to a node near the base level goal node.
- The process is continued until a base level goal node is achieved which is closest to the optimal goal node

A Hierarchical Search



1/28/2021 3:54:47 PM

LIMITED HORIZON SEARCH

- In some problems, it is computationally infeasible for any search method to find a path to the goal. In other words, an action might be taken within the time limit that does not permit the search all the way to a goal.

Limited Horizon Search

- Suppose the time available for search (before an action must be selected) allows the search to a depth “d”. Thus all paths of length “d” or less can be searched. Nodes lying at that depth denoted by “H” is called the horizon nodes.
- Our search process will be to search to depth “d” and then select n* such that:

$$n^* = \underset{n \in H}{\textit{argmin}} \quad f'(n)$$

MEASURE OF PERFORMANCE OF HEURISTICS

- The performance measures are useful in comparing various heuristics. One such measure is called *Penetrance*.

- **Definition:** The *Penetrance* „P‟ of a search process (either routine or heuristic) is the capability with which the search can concentrate towards the goal(s), rather than wandering off in some irrelevant directions.

PENETRANCE contd..

Expressed symbolically:

$$P = N1 / N2$$

$$1 >= P > 0$$

Where:

N1 = Total number of nodes to be explored to directly land onto the goal, excluding the start node.

N2 = Total number of nodes actually explored during the search, excluding the start node.

PENETRANCE CONTD..

- If the heuristic is so powerful to understand in which direction in the state space representation of the problem, the goal(s) is/are lying, then the search would directly land on the goal state(s) without wandering off in the irrelevant directions and accordingly the „Penetrance‟ P will achieve its maximum value of unity (1).
- All routine searches or unguided searches are having the value of „Penetrance‟ P close to zero.

Problems and corresponding Heuristics

Problems and corresponding Heuristics

- Generate and Test: Ignorable and Absolute.
- Hill Climbing: Irrecoverable and Absolute.
- Steepest Ascent Hill Climbing: Irrecoverable and Absolute.
- Best First Search: Recoverable, Absolute and State.
- A * Heuristic: Recoverable and Relative and State.
- AND / OR: Decomposable and Relative.
- AO * Decomposable and Absolute and State.
- Means End Analysis: Absolute, Decomposable and State.