Unit-1. Problem Solving

1.1. Introduction to AI

Artificial Intelligence deals with making machines perform tasks that are typically associated with human intelligence, such as problem-solving and learning. There are many applications of AI like: • gaming (machines can play games with humans)

- expert systems (machines having knowledge and reasoning capabilities like human experts such as physicians, engineers, and others),
- computer vision (machines with ability to recognize objects/activity in images and videos) • natural language processing (ability to understand human languages like English, Hindi, and others)
- robotics (creating a robot to do specialized or routine physical tasks done by humans), etc.

The first paper recognized as "AI"was published in 1943 (paper on artificial neuron by McCullouch and Pitts). The field of AI research was founded in 1956 in a workshop organized by **John McCarthy**, who is now considered as **Father of AI**. From 1956-1973, active research was conducted in the field of AI but the period from 1974-1980 is known as "First AI winter" as fundings for AI research was hard to get. With the emergence of expert systems in 1981-86, AI again came into forefront, but after failure of expert systems, there was "Second AI winter" from 1987-2000. From 2000 onwards, machine learning came into forefront and has been on rise ever since.

Alan Turing was another AI researcher who proposed a **Turing test** in 1950 to classify whether a machine can be labelled as AI or not. The test was originally called "imitation game". In Turing test, there are 3 isolated rooms, one having a human expert, second having an intelligent machine, and third having a human interrogator. The human expert doesn't know which room has human expert and which room has intelligent machine. The human interrogator asks series of questions and gets the reply from both the rooms. If the interrogator can't decide which room has intelligent machine after asking multiple questions, then the machine is said to have passed the test.

Turing test has received criticism as The Turing test does not test for highly intelligent behaviours, such as the ability to solve difficult problems or come up with original insights. **Chinese room test** is intended to show that, even if the Turing test is a good operational definition of intelligence, it may not indicate that the machine has a mind, consciousness, or intentionality.

Chinese Room Experiment: suppose that artificial intelligence research has succeeded in constructing a computer that behaves as if it understands Chinese. It takes Chinese characters as input and, by following the instructions of a computer program, produces other Chinese characters, which it presents as output. Suppose, that this computer performs its task so convincingly that it comfortably passes the Turing test: it convinces a human Chinese speaker that the program is itself a live Chinese speaker. To all of the questions that the person asks, it makes appropriate responses, such that any Chinese speaker would be convinced that they are talking to another Chinese-speaking human being. Does the machine literally "understand" Chinese? Or is it merely simulating the ability to understand Chinese?

1.2. AI techniques

One of the few hard and fast results to come out of the first three decades of AI research is that

Intelligence requires knowledge. An AI technique is a method that exploits knowledge. Three important AI techniques are:

i. Use of knowledge – Knowledge should capture generalization i.e. situations that share important properties are grouped together.
ii. Search – process of finding a solution for given problem.
iii. Abstraction – separation of important features from unimportant ones.

1.3. Problem Solving

To build a system to solve a particular problem, we need to do 4 things:

- Define the problem precisely as state space search
- Analyze the problem
- Isolate and represent the task knowledge that is necessary to solve the problem. • Choose the best problem-solving technique(s) and apply them to the particular problem.

1.3.1. State Space Search

We must specify the initial/start situation and final/goal situation of the problem. We must then find a mechanism to represent the start, goal and all intermediary states of the problem. Let's take the example of **water-jug problem** which is stated as follows: You are given two jugs, a 4L one and a 3L one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2L water into 4L jug.

The state space for this problem can be described as the set of ordered pairs (x,y) where x represents amount of water in 4L jug, and y represents amount of water in 3L jug. The start state is (0,0) and goal state is (2,n).

1.3.2. Analysing the problem

In order to choose the most appropriate method to solve a particular problem, it is necessary to analyse the problem wrt 7 characteristics:

i. Is the problem decomposable into a set of smaller sub-problems?
ii. Can solution steps be undone?
iii. Is the problem's universe predictable i.e. can we specify next state for each operation? iv. Is the desired solution a state or a path?
v. Is a good solution absolute or relative?
vi. Is a large amount of knowledge required so solve the problem or just to constrain the search?
vii. Will the solution require interaction with a human?

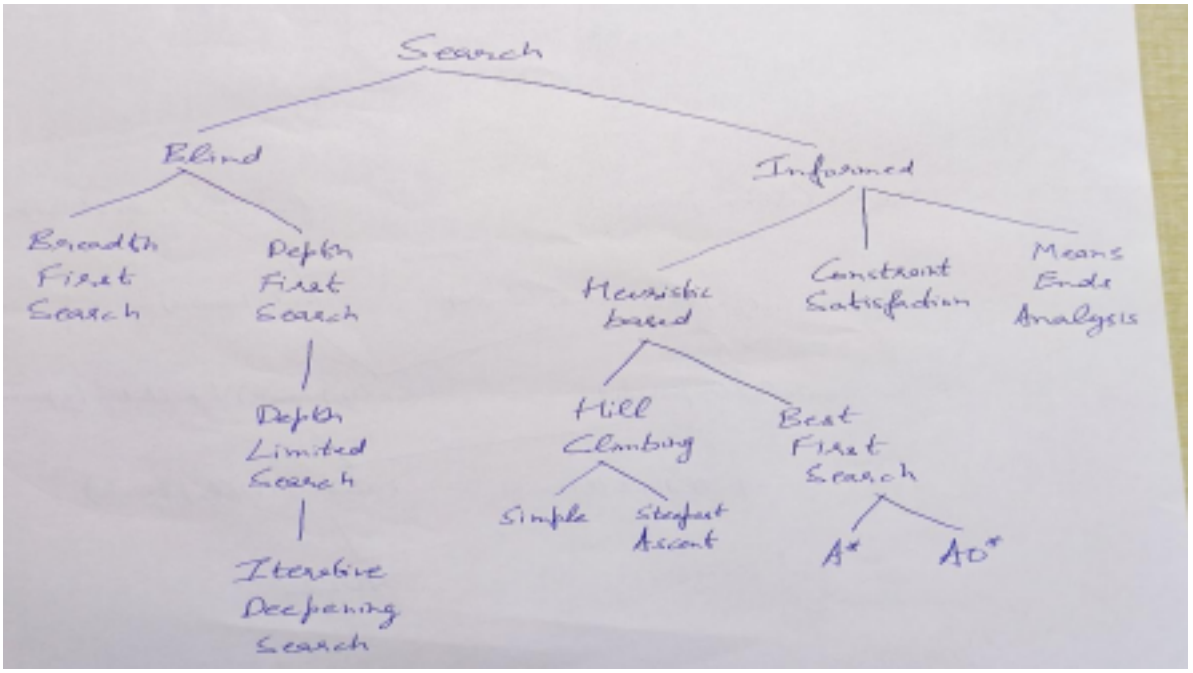
1.3.3. Production System

A production system is a structure to represent task knowledge. It consists of: i. A set of rules: each rule consists of pre-conditions and post-conditions. ii. Databases: one or more databases which may contain some additional information. iii. Control strategy: provide ways of resolving a conflict when several rules match at once. iv. Rule applier: an algorithm to apply rules

Let's take an example of production rules for water-jug problem.

Task	Rule
Fill the 4L jug	(x,y) ⟶ (4,y) if x<4
Fill th3 3L jug	(x,y) ⟶ (x,3) if y<3
Empty the 4L jug on ground	(x,y) ⟶ (0,y) if x>0
Empty the 3L jug on ground	(x,y) ⟶ (x,0) if y>0
Pour all the water from 4L to 3L jug	(x,y) ⟶ (0,x+y) if x+y≤3 and x>0
Pour all the water from 3L to 4L jug	(x,y) ⟶ (x+y,0) if x+y≤4 and y>0
Pour water from the 4L jug into 3L jug until the 3L jug is full	(x,y) ⟶ (x-(3-y),3) if x+y≥3 and x>0
Pour water from the 3L jug into 4L jug until the 4L jug is full	(x,y) ⟶ (4,y-(4-x)) if x+y≥4 and y>0

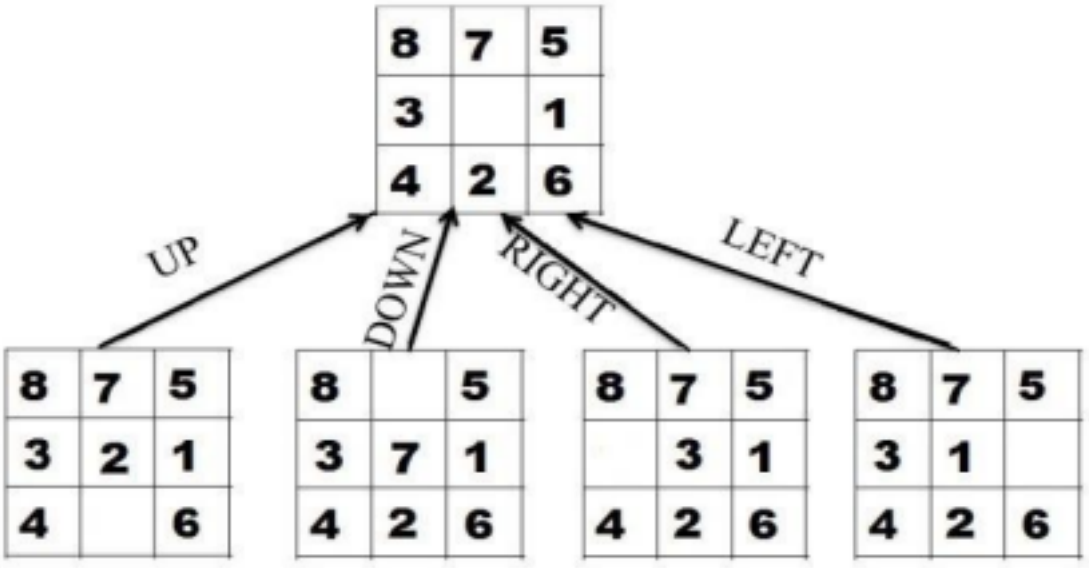
1.3.4. Search Algorithms



1.4. Blind Search

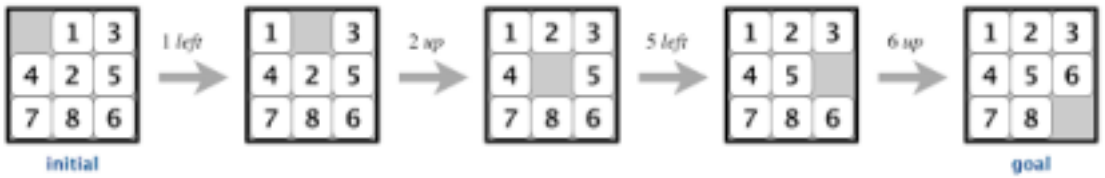
In blind search, we move in the search space with a pre-set criteria without any comparison between states/paths. There are two main blind search algorithms: breadth first search (**BFS**) and depth first search (**DFS**). **In BFS we move horizontally whereas in DFS we move vertically.**

Let's take example of 8-puzzle problem.



As in the given state, we can apply 4 move. In BFS, we apply all the 4 move to generate 4 child states. We do this for each state one-by-one till we achieve the goal state.

In DFS, we only generate one successor for each state, and move to the successor.



Algorithm: Depth-First Search

1. If the initial state is a goal state, quit and return success.
2. Otherwise do the following until success or failure is signalled:

(a) Generate a successor, E of the initial state. If there are no more successors, signal failure.
(b) Call Depth-First Search with E as the initial state.
(c) If success is returned, signal success. Otherwise continue in this loop.

Algorithm: Breadth-First Search

1. Create a variable called LSIT and set it to the initial state.
2. Until a goal state is found or LIST is empty:

(a) Remove first element from LIST and call it E. If LIST was empty, quit.
(b) If new state is a goal state, quit and return this state.
(c) Otherwise, add the new state to the end of LIST.

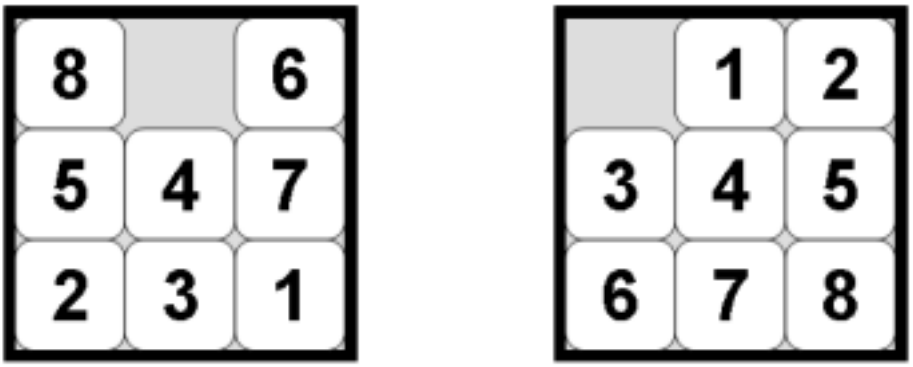
Advantages & Limitations of BFS/DFS

- Depth First Search requires less memory (as it stores the nodes of only the current path) but may fail to find solution (as it may get trapped exploring a blind alley).
- Breadth First Search always finds the optimal solution but requires more memory compared to DFS as it stores all the nodes.

As DFS can get trapped exploring blind alley (i.e. a long path without any certainty of finding the goal), this limitation can be handled by applying a limit to the depth till which a path can be explored. This variant of DFS is known as **Depth Limited Search (DLS)**.
A limitation of DLS is that it may give up exploring a path when the solution is nearby due to depth limit constraint. To overcome this limitation, instead of one fixed depth limit, the limit may be increased iteratively till the solution is found. This variant of blind search algorithm is called as **Iterative Deepening Search (IDS)**.

1.5. Heuristic Search

A **heuristic** is a function which helps us to evaluate each state to guide the search process. For example, in 8-puzzle problem one of the heuristics can be "number of tiles in correct position". The goal/final state has the best heuristic value (maxima/minima).

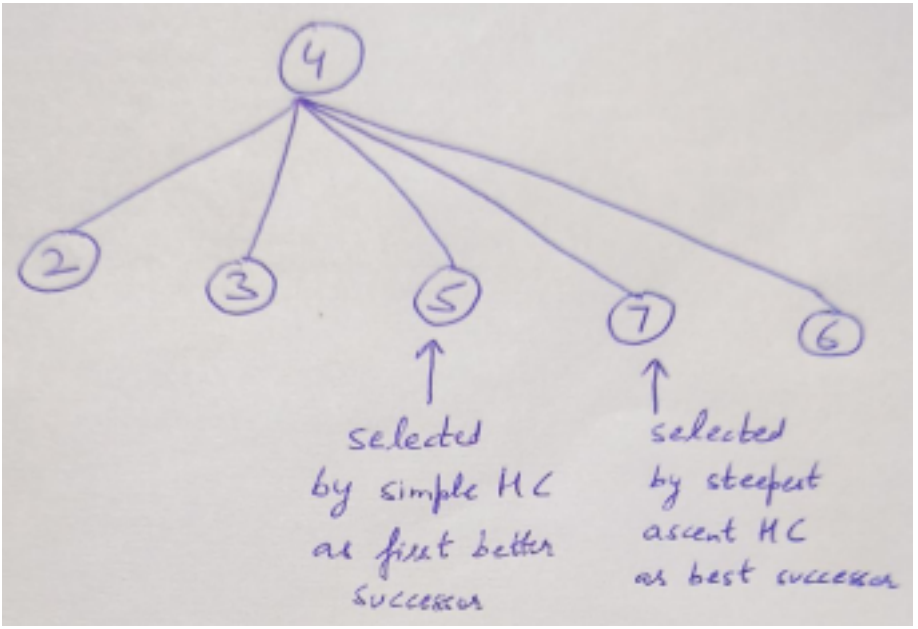


h(start)=1 h(goal)=8

Uniform Cost Search is used to find the shortest path in a graph from source to destination. In this algorithm, the heuristic is the cumulative cost of each path from source node to current node. In **branch-and-bound** technique (used for finding shortest path), we begin generating complete paths, keeping track of the shortest path found so far. We give up exploring any path as soon as its partial length becomes greater than the shortest path found so far.
In **generate-and-test** method, we generate solutions randomly and check whether it satisfies the criteria of the goal state. This method is useful for problems with **combinatorial explosion** i.e. number of solutions grow with extremely high rates with minor increase in state parameters. For example, Traveling Salesman Problem has (N-1)! solutions for N cities i.e. for 6 cities there are 120 solutions but for 11 cities there are 3628800 solutions [Traveling Salesman Problem: A salesman has a list of cities, each of which he must visit exactly once. There are direct roads between each pair of cities on the list. Find the route the salesman should follow for the shortest possible round trip that both starts and finishes at any one of the cities].

1.5.1. Hill Climbing

Hill climbing is a heuristic-based search algorithm having two variants: **simple hill climbing** and **steepest ascent hill climbing**. In simple hill climbing, successors of current state are generated one by one until a successor which has better heuristic value than current state is generated. In steepest ascent hill climbing, the best among all the possible successors of the current state is selected.



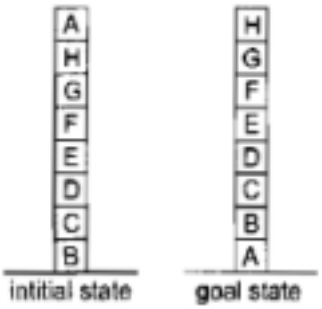
Algorithm: Simple Hill Climbing

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until there are no operators left to be applied in the current state:
   (a) Select an operator that has not yet been applied to the current state and apply it to produce a new state.
   (b) Evaluate the new state.
      (i) If it is a goal state, then return it and quit.
      (ii) If it is not a goal state but it is better than the current state, then make it the current state.
      (iii) If it is not better than the current state, then continue in the loop.

Algorithm: Steepest Ascent Hill Climbing

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until a complete iteration produces no change to current state: (a) Let SUCC be a state such that any possible successor of the current state will be better than SUCC.
   (b) For each operator that applies to the current state do:
      (i) Apply the operator and generate a new state.
      (ii) Evaluate the new state. If it is a goal state, then return it and quit. If not, compare it to SUCC. If it is better, then set SUCC to this state. If not better, leave SUCC alone. (c) If the SUCC is better than current state, then set current state to SUCC.

Let's try to apply simple hill climbing to blocks world problem.



Here, only one block can be move at a time and multiple stacks can be made.

One of the heuristic function which can be used for this is “add one point for every block that is resting on the thing it is supposed to be resting on. Subtract one point for every block that is sitting on the wrong thing”.
Using this heuristic function, the goal state will have heuristic value 8 (+1 for each block). For initial state, B=-1 (as B is resting on ground instead of A), A=-1 (as A is resting on H instead of ground), and +1 for rest i.e. heuristic value 4.
Only one successor is possible for initial state, where we move A and keep it on the ground. For this state, B=-1 and + for rest i.e. heuristic value 6. As h(initial)=4 and h(goal)=8, we have to maximize the heuristic value, hence 6 is better than 4. Hence, this state will be selected by hill climbing.

Limitations of Hill Climbing

Hill climbing may fail to find a solution due to following cases:

i. Local maxima: we may reach a state where no successor is better for the current node. ii. Plateau: current state and all successors have the same heuristic value. iii. Ridge: special kind of local maxima in which we can come out by making two or few moves combined.

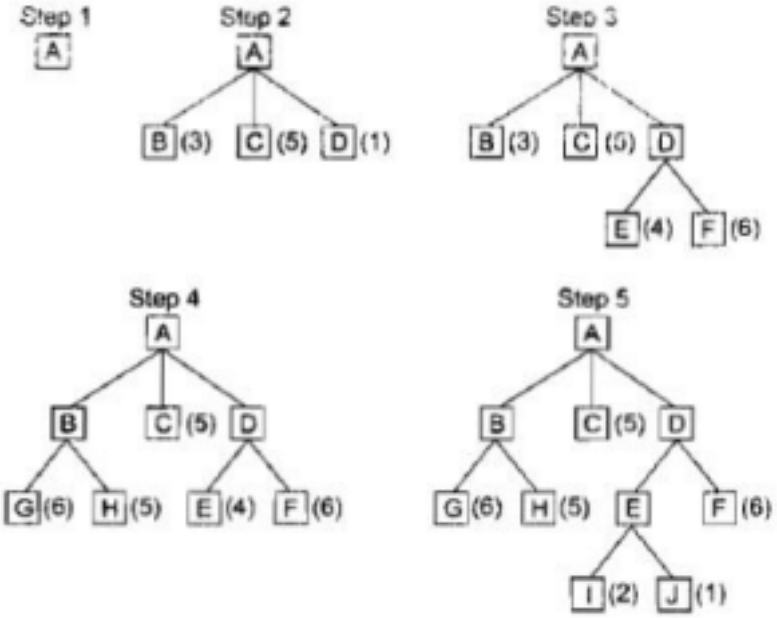
Handling problems of hill climbing

i. Backtrack to earlier node and try going in a different direction.

ii. Make a big jump in some direction to try to get to a new section of the search space. iii. Apply two or more rules before doing the test (specially for dealing with ridges).

1.5.2. Best First Search/A*

In hill climbing, we select a node/path and discard rest of its siblings permanently. In best first search, the nodes are never discarded and are always considered for processing. Hence, in hill climbing we cannot change the path during search process, but in best first search we can change the path during the search process. In the example below, node will lower heuristic value is better.



To implement best first search, two lists are used: OPEN and CLOSED. OPEN list contains the unprocessed nodes. CLOSED list contains the processed nodes.

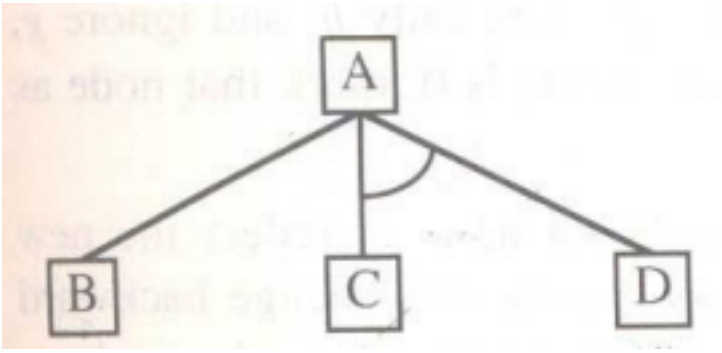
The heuristic function used by best first search is **f'=g+h'** where g is the cost of reaching current node from start node, h' is the estimated cost of reaching goal node from current node, and f' is the estimated cost of reaching goal node from start node.

Algorithm: Best-First Search
1. Start with OPEN containing just the initial state.
2. Until a goal is found or there are no nodes left on OPEN do:
(a) Pick the best node on OPEN.
(b) Generate its successors and add it to CLOSED.
(c) For each successor do:
(i) If it has not been generated before, evaluate it, add it to OPEN, and record its parent. (ii) If it has been generated before, change the parent if this new path is better than the previous one. In that case, update the cost of getting to this node and to any successors that this node may already have.

The best first search algorithm presented above is the simplification of A* algorithm. If we can guarantee that h' never over-estimates h then A* becomes **admissible** i.e. guaranteed to find optimal path.

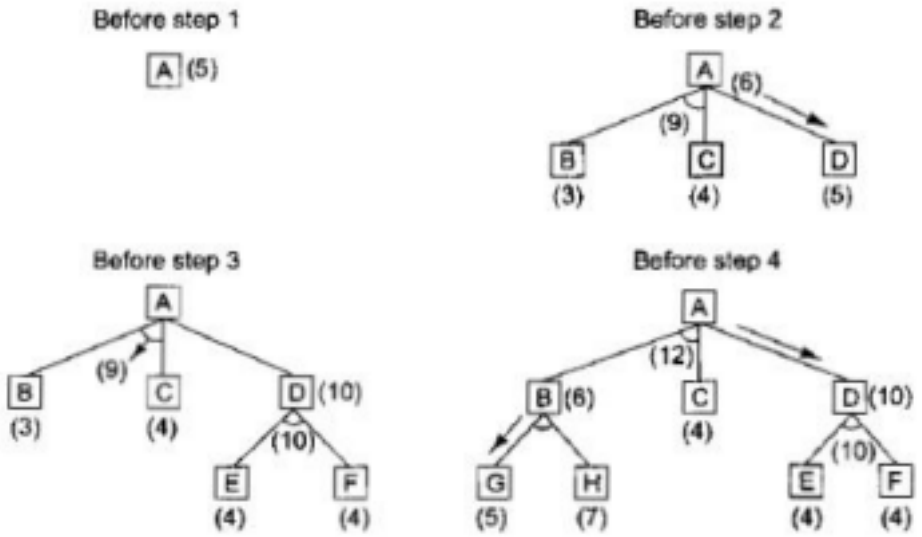
1.5.3. Problem Reduction/AO*

Problem reduction is a method in which we decompose a problem into smaller sub-problems. This decomposition usually results in and AND-OR graph.



The above graph describes that to solve A, we can either do task B or do both C and D tasks.

AO* algorithm is a search algorithm for AND-OR graph (analogous to A* on OR graphs). In this algorithm only a single structure GRAPH is utilized which stores the part of search graph that has been explicitly generated so far. The heuristic function used is h' as it is not possible to compute a single value of "g" as there may be multiple paths from start node to current node.



In an AND arc, the value of parent node is the sum of heuristic values of the child nodes and number of nodes in the arc.

1.6. Constraint Satisfaction

Many problems in AI can be viewed as problems of constraint satisfaction in which the goal is to discover some problem state that satisfies a given set of constraints. Constraint satisfaction is a two step process. First constraints are discovered and propagated as far possible throughout the system. Then, if there is still not a solution, search begins. A guess about something is made and added as a new constraint. Propagation can then occur with this new constraint.

N-queens problem where we have to place N-queens on a NXN chessboard such that no two queens are in same row, column, and diagonal can be solved by constraint satisfaction. We place first queen randomly (guess), then we exclude the row, column and diagonal of the first queen and place the second queen randomly in permitted positions. We repeat this process for all queens.

Another domain where constraint satisfaction is used is solving crypt-arithmetic puzzles like SEND+MORE=MONEY, where each letter represents a unique digit between 0 to 9.

	C3	C2	C1	
	S	E	N	D
+	M	O	R	E
M	O	N	E	Y

M=1 [as sum of two 4 digits number are generating a 5 digit number]

So, C3+S+M≥10 i.e. C3+S≥9
If C3=1 and S=9 then C3+S+M=11 i.e. O=1 which is not possible as M=1.
If C3=1 and S=8 then C3+S+M=10 i.e. O=1.
If C3=0 and S=9 then C3+S+M=10 i.e. O=1.
So, **O=0** and **S = 8 (C3=1) or 9 (C3=0)**.

As, C2+E+O=N and O=0, so C2+E=N.
If C2=0 then E=N which is not possible, so **C2=1** and **N=E+1**.

Constraints can't be propagated any further, so we have to make a guess on for a letter which makes the maximum impact i.e. letter with highest occurrences among the remaining letters.

So, taking **E=5**.
N=6, C3=0, S=9.
N+R+C1=15
If C1=0, R=9 which is not possible as S=9.
If **C1=1, R=8**.

D+E=10+Y
D=7, Y=2.

1.7. Means Ends Analysis

The means-ends analysis process centres on the detection of differences between the current state and the goal state. Once such a difference is isolated, an operator that can reduce the difference must be found. If the selected operator cannot be applied directly to the current state, then achieving the state on which the selected operator can be applied becomes our new goal. This process is known as **operator sub-goaling**. To solve a problem by MEA, we need to define a set of operations and a difference table (which indexes the operations by the differences that they can be used to reduce.

Let's take the example of household robot whose task is to move a table with a box on top of it from one room to another with the condition that table can't be moved when the box is on top of it. We must identify the differences between the various states while solving the problem, and the operators which can be used to reduce these differences. The differences for this problem are: clear the table, move the table, move the robot, place box on table, pick up the box from table, empty the robot arms. The operators that can be used for this problem are:

Operator	Pre-conditions	Post-condition

PUSH	1. Robot should be at object. 2. Object should be large. 3. Object should be clear. 4. Robot arms must be empty	1. Object is at desired location.
CARRY	1. Robot should be at object. 2. Object should be small. 3. Robot should be holding the object.	1. Object is at desired location.
PICKUP	1. Robot should be at object. 2. Object should be small. 3. Robot arms must be empty.	1. Robot is holding the object. 2. Robot arm is not empty.
PUTDOWN	1. Robot is holding the object.	1. Robot is not holding the object. 2. Robot arm is empty.
WALK	-	1. Robot is at desired location.
PLACE	1. Robot should be at large object. 2. Robot should be holding an object.	1. Small object is on large object.

A difference table is used to show the mapping between the differences and operators.

	PUSH	CARRY	WALK	PICKUP	PUTDOWN	PLACE
Clear table				*		
Move object	*	*				
Move robot			*			
Place box on table						*
Hold an object in robot hands				*		
Empty the robot hands					*	*

To solve a problem by MEA, we first identify the main/major difference between the start state and the goal state. The main difference in household robot problem is moving the table from one room to another. To reduce this difference, we can use PUSH operator. But we can't directly apply PUSH operator in start state as it has a pre-condition that the table should be clear. So, satisfying the pre conditions of PUSH becomes our new sub-goal. Hence, it is an iterative process, where we keep dividing a goal into sub-goals.

Algorithm: MEA(CURRENT, GOAL)

1. Compare CURRENT to GOAL. If there are no differences between them then return. 2. Otherwise, select the most important difference and reduce it by doing the following until success or failure is signalled:

(a) Select an as yet untried operator O that is applicable to the current difference. If there are no such operators, then signal failure.

(b) Attempt to apply O to CURRENT. Generate descriptions of two states: O-START, a state in which O's pre-conditions are satisfied and 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 concatenating FIRST-PART, O, and LAST-PART.

The steps of solving household robot problem are as follows, assuming that initially robot is not near the table and its hands are empty.

1. Select the major difference "move object" where the object is table. The operator which can be used to reduce this difference is PUSH. But we can't apply this operator as table is not clear initially.
2. So, our new goal is to "clear table". To "clear table", we select PICKUP operator. But we can't apply it as its pre-condition that "robot should be at object" is not satisfied.
3. So, our new sub-goal is that "robot should be at object". To reduce this difference, we select and apply WALK operator as it has no pre-conditions. After applying, this operator, robot reaches the table/box. Now PICKUP operator can be applied as its pre-conditions are met.
4. But we can't still apply PUSH operator, as robot arms are not empty as it is holding the box. So, the difference now which we need to reduce is "empty the robot hands". To reduce this difference, we use the operator PUTDOWN.
5. Now, all the pre-conditions of PUSH are satisfied. So, we can apply PUSH operator to move the table to the next room.
6. Now current state is that box is in room A whereas table and robot are in room B. So, the major difference now is to move box from room A to room B. For this, we can use CARRY operator, but to apply CARRY operator robot needs to PICKUP the object, and for PICKUP, robot should be at box. So, we apply WALK operator to reach the box.
7. Next, we apply operators PICKUP, CARRY, and PLACE to complete the task.

EXERCISE-1

1. When would best-first search be worse than simple breadth-first search?

   ANSWER

   If we select a bad heuristic function for best-first search.

2. Show how MEA could be used to solve the problem of getting from one place to another. Assume that the available operators are WALK, DRIVE, TAKE THE BUS, TAKE A CAB, FLY.

   ANSWER

   As we are given the operators, we identify the differences: Move from one city to another, Go to/from airport, Reach the vehicle. Next we generate a difference table.

	WALK	DRIVE	TAKE THE BUS	TAKE A CAB	FLY
Move from one city to another					*
Go to airport		*	*	*	

Go from airport			*	*	
Reach the vehicle	*				

Now we must specify the pre-conditions and post-conditions of each operator.

	Pre-conditions	Post-conditions
WALK	-	At the vehicle
DRIVE	At the car	At the airport
TAKE THE BUS	At the bus	At the destination
TAKE A CAB	At the cab	At the destination
FLY	At the airport-1	At the airport-2

The most important difference is FLY. To apply it, we must satisfy its pre-condition of being "at airport-1". To satisfy this pre-conditions, we select operator TAKE THE BUS (or DRIVE or TAKE A CAB). To apply this operator, we need to satisfy the pre-condition of being "at the vehicle". For this, we apply WALK operator. Now, we can apply TAKE THE BUS operator and then FLY operator. After applying FLY operator, we are "at airport-2", so we need to "go from the airport", for this we select the operator TAKE A CAB. But to apply this we need to fulfil it's pre-condition by applying WALK. Now, can apply TAKE A CAB and then WALK to reach the final destination.

3. Solve the crypt-arithmetic puzzle CROSS+ROADS=DANGER.

ANSWER

D=1, R=2S

S=3 (guess), R=6, C1=0, E=4, C2=0

O=2 (guess)

C3+R+O=C3+6+2=C3+8 => C4=0

C4+C+R=10+A => 0+C+6=10+A => C=A+4

As the available values are 0,5,7,8,9 we take A=5, C=9.

C2+O+A=0+2+5=7=G

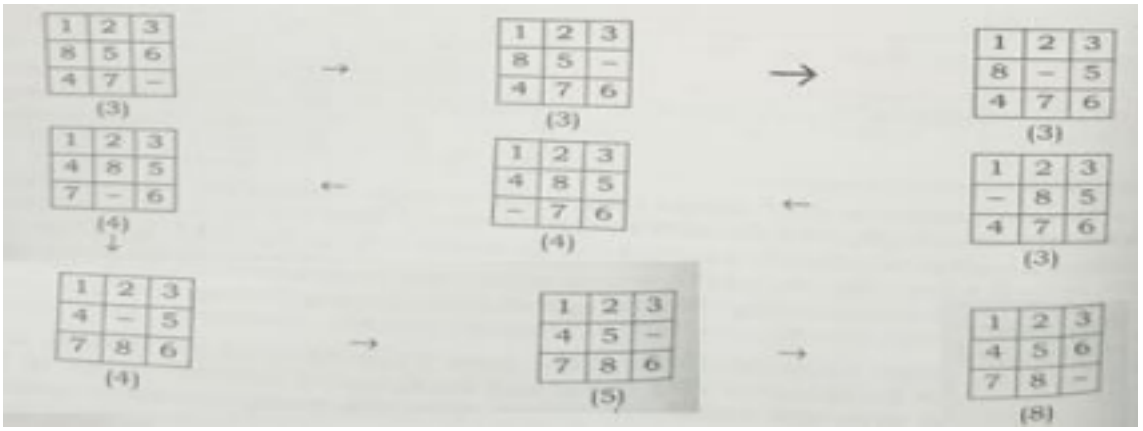
C3=0

C3+R+O=0+6+2=8=N

4. Solve the following 8-puzzle problem using hill climbing.

ANSWER

Choosing the heuristic “no. of tiles in correct position” will result in local maxima after few steps. So, we take the heuristic ”no. of tiles in sequence from start” i.e. in start state first 3 tiles are in sequence, so its heuristic value is 3.

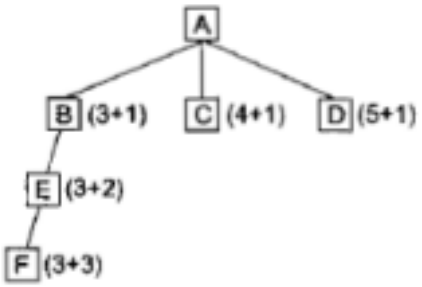


5. Prove admissibility of A*.

ANSWER

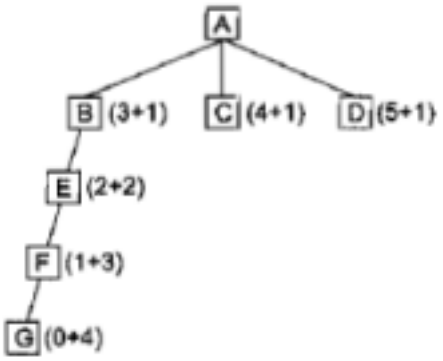
A* is admissible (returns optimal solution) when we guarantee that h’ never over-estimates h.

Case-1: h’ under-estimates h



In the above figure, heuristic is f’=h’+g, where g is number of branches traversed to reach the current node, and h’ is the estimated number of branches needed to be traversed to reach the goal node. Node B with h’=3 has lowest f’ value compared to C and D, so B is expanded. But after processing B, we find E also has h’=3 i.e. h’ was under-estimated at B to be 3 instead of 4. Now, f’(E)=5, f’(C)=5, f’(D)=6, so between E and C, we select E to continue in same path. Let E generate node F which also has h’=3 i.e. we under-estimated h’(B)=3 instead of 5. Now f’(F)=6 but f’(C)=5 and f’(D)=6, so now we select node C. Hence, even though we under-estimated h’(B) to be 3 instead of 5 but still it has no major impact on finding an optimal path as we can change the path on finding its actual value.

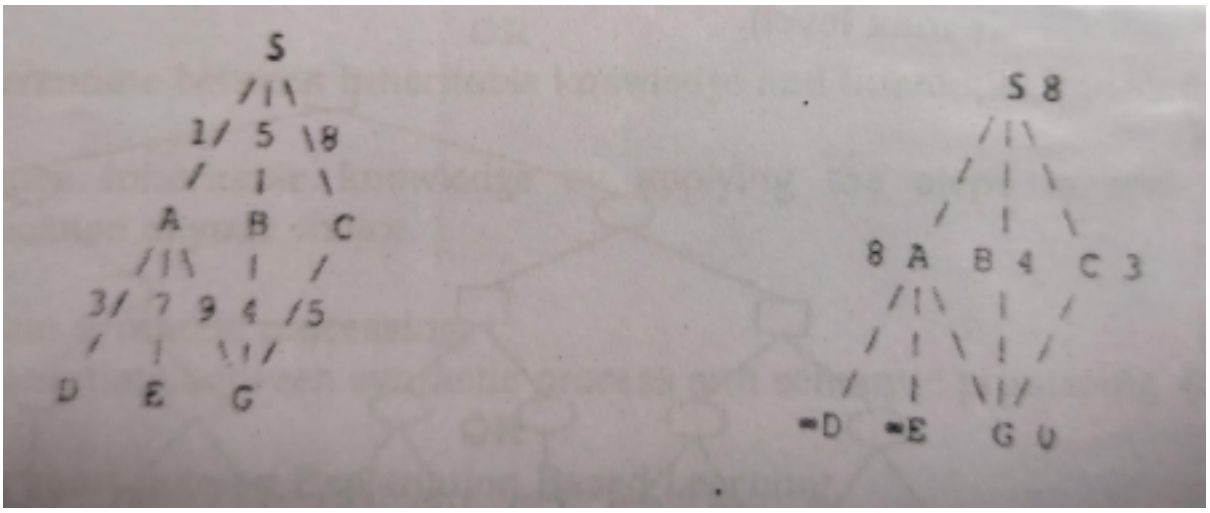
Case-2. H’ over-estimates h



In this figure, suppose h(D)=1 i.e. f(D)=2 but we over-estimate h'(D)=5 so that f'(D)=6, so as we can see above node B is selected as f'(B)=f'(E)=f'(F)=f'(G)=4. Although cost of reaching goal G via D is 2 but we selected path A-B-E-F-G with cost 4 as we over-estimated h'(B). Hence, over-estimating h' may not give the optimal solution.

EXERCISE-2

1. Given the following graphs, where the first graph contains edge cost and second graph contains the heuristic cost. The start state is S and goal state is G .



List the order of nodes generated and expanded using BFS, DFS, IDS and A*.

ANSWER

(a) BFS: S, A, B, C, D, E, G

(b) DFS: S, A, D, E, G, B, C (traversal: S, A, D, A, E, ,A, G, B, G, C)

(c) IDS-1: S, A, B, C

IDS-2: S, A, D,E, G, B, C (traversal: S, A, D, A, E, ,A, G, A, S, B, S, C)

IDS-3: S, A, D,E, G, B, C (traversal: S, A, D, A, E, ,A, G, B, G, C)

(d) A*

OPEN: S

CLOSED: {}

CLOSED: S

OPEN: A (1+8=9), B(5+4=9), C (8+3=11)

Parent of each node in OPEN list is S.

As A and B both have the minimum value, we randomly select A.

CLOSED: S, A
OPEN: B(9), C(11), D(4+INF=INF), E (8+INF=INF), G(10+0=10)
Parent of B and C is S. Parent of D, E and G is A.
As B has minimum value, we expand B.

CLOSED: S, A, B,
OPEN: B(9), C(11), D (INF), E (INF), G(9+0=9).
Parent of B and C is S. Parent of D and E is A. Parent of G is B.
As, G has minimum value, we select G which is the goal node.

CLOSED: S, A, B, G.

As final parent of G is B, parent of B is S. So, the path selected is S, B, G.

2. Solve the following crypt-arithmetic puzzle: SOME+TIME=SPENT.

ANSWER

S=1

P=0

T=8, C3=1, E-4, C1=0

N=6, M=3, C2=0

O=5, I=9

3. Solve the following crypt-arithmetic puzzle: FOUR+MICE=FOUND.

ANSWER

F=1, O=0

I=7 (guess)

U=8, M=9, R=2, E=3, D=5, C=6, N=4

4. Solve monkey banana problem using means ends analysis.

ANSWER

A hungry monkey finds himself in a room in which a bunch of bananas is hanging from the ceiling. The monkey, unfortunately, cannot reach the bananas. However, in the room there are also a chair and a stick. The ceiling is just the right height so that a monkey standing on a chair could knock the bananas down with the stick. The monkey knows how to move around, carry other things around, reach for the bananas, and wave a stick in the air. What is the best sequence of actions for the monkey to take to acquire lunch?

Operator Table

Operator	*Pre-conditions*	*Post-conditions*
WALK	None	Monkey at location.
PUSH	Monkey at chair	Chair at location. Monkey at chair.
CLIMB	Monkey at chair. Chair at location.	Monkey on chair.

GRASP	Monkey near stick.	Monkey holding stick.
WAVE	Monkey holding stick. Monkey on chair.	Bananas on ground.

Difference Table

	WALK	PUSH	CLIMB	GRASP	WAVE
Hit the bananas					*
Climb chair			*		
Hold stick				*	
Move the chair below the bananas		*			
Reach the chair	*				

The most important difference is hitting the bananas for which we can use WAVE operator. But to apply this operator the monkey should be on chair and must be holding the stick. So, now the most important difference is that monkey should be on chair, for which we can use CLIMB operator. But to apply climb operator, chair should be at desired location, for which we can use PUSH operator. But to apply PUSH operator, monkey should be at chair, so for this we can use MOVE operator.