**Artificial Intelligence**

**Unit-1**

# Introduction to Artificial Intelligence, AI Problems, AI Techniques, Problems ,Problem Space and Search, Defining the problem as a state space search, Production system, Problem characteristics, Heuristic search Technologies: Generate and Test, Hill Climbing, Best First Search, optimal and A\* algorithm, Problem Reduction, means-end-analysis, AO \* algorithm. AND-OR graph.

**Introduction to Artificial Intelligence**

* Artificial Intelligence is composed of two words **Artificial** and **Intelligence**, where Artificial defines *"man-made,"* and intelligence defines *"thinking power"*, hence AI means *"a man-made thinking power."*
* So, we can define AI as:
* "It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions."

**What is Artificial Intelligence?**

* According to the father of Artificial Intelligence, John McCarthy, it is *“The science and engineering of making intelligent machines, especially intelligent computer programs”.*
* Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.
* AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.
* The exciting new effort to make computers think … *machines with minds,* in the full literal sense. By Haugeland, 1985
* The study of how to make computers do things at which, at the moment, people are better. By Rich & Knight, 1991.
* The study of mental faculties through the use of computational models.

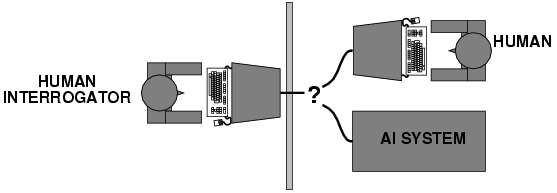
By . Charniak and McDermott, 1985

* A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes. By Schalkoff, 1990

 According to these definitions computer systems can be classified into the following 4 categories to pursue in AI.

**1. Systems that act like humans**

* **Computational models of human behaviour**
  + Programs that behave (externally) like humans.
  + This is the original idea from Turing and the well known **Turing Test** is to use to verify this.
* You enter a room which has a computer terminal. You have a fixed period of time to type what you want into the terminal, and study the replies. At the other end of the line is either a human being or a computer system.
* If it is a computer system, and at the end of the period you cannot reliably determine whether it is a system or a human, then the system is deemed to be intelligent.



**2. Systems that think like humans**

* Computational models of human “thought”
  + Programs that operate (internally) the way humans do
* It associate with human thinking, activities like decision making, problem solving, learning etc...
* Humans as observed from ‘inside’
* How do we know how humans think?
  + Introspection vs. psychological experiments
* Cognitive Science

**3. Systems that think rationally (logically)**

* Computational systems that behave intelligently?
  + But what does it mean to behave intelligently?
* Humans are not always ‘rational’
* Rational - defined in terms of logic?
* Logic can’t express everything (e.g. uncertainty)
* Logical approach is often not feasible in terms of computation time (needs ‘guidance’)
  + Example

if

X is a man

all men are mortal

then

X is a mortal

**4. Systems that act rationally**

* Computational systems that behave rationally
  + More widely accepted view
* Rational behavior: doing the right thing
* The right thing: that which is expected to maximize goal achievement, given the available information
* Giving answers to questions is ‘acting’.
* I don't care whether a system:
  + replicates human thought processes
  + makes the same decisions as humans
  + uses purely logical reasoning
* What means “behave rationally” for a person/system:
  + Take the right/ best action to achieve the goals, based on his/its knowledge and belief
* Example. Assume I don’t like to get wet (my goal), so I bring an umbrella (my action). Do I behave rationally?
  + The answer is dependent on my knowledge and belief
  + If I’ve heard the forecast for rain and I believe it, then bringing the umbrella is rational.
  + If I’ve not heard the forecast for rain and I do not believe that it is going to rain, then bringing the umbrella is not rational

**Why Artificial Intelligence?**

* Before Learning about Artificial Intelligence, we should know that what is the importance of AI and why should we learn it. Following are some main reasons to learn about AI:
  + With the help of AI, you can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.
  + With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.
  + With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.
  + AI opens a path for other new technologies, new devices, and new Opportunities.

**Goals of Artificial Intelligence**

Following are the main goals of Artificial Intelligence:

* **To Create Expert Systems** − The systems which exhibit intelligent behavior, learn new things by it selfs, demonstrate, explain, and advice its users.
* **To Implement Human Intelligence in Machines** − Creating systems that understand, think as well as see, hear, walk, talk,feel, learn, and behave like humans.
* To make computers more useful which can perform tasks that require human intelligence such as:
  + Proving a theorem
  + Playing chess
  + Plan some surgical operation
  + Driving a car in traffic
* Understand principles of human intelligence
* Solve Knowledge-intensive tasks

An intelligent connection of perception and action

**Advantages of Artificial Intelligence**

Following are some main advantages of Artificial Intelligence:

* **High Accuracy with less errors:** AI machines or systems are prone to less errors and high accuracy as it takes decisions as per pre-experience or information.
* **High-Speed:** AI systems can be of very high-speed and fast-decision making, because of that AI systems can beat a chess champion in the Chess game.
* **High reliability:** AI machines are highly reliable and can perform the same action multiple times with high accuracy.
* **Useful for risky areas:** AI machines can be helpful in situations such as defusing a bomb, exploring the ocean floor, where to employ a human can be risky.
* **Digital Assistant:** AI can be very useful to provide digital assistant to the users such as AI technology is currently used by various E-commerce websites to show the products as per customer requirement.
* **Useful as a public utility:** AI can be very useful for public utilities such as a self-driving car which can make our journey safer and hassle-free, facial recognition for security purpose, Natural language processing to communicate with the human in human-language, etc.

**Disadvantages of Artificial Intelligence**

Every technology has some disadvantages, and thesame goes for Artificial intelligence. Being so advantageous technology still, it has some disadvantages which we need to keep in our mind while creating an AI system. Following are the disadvantages of AI:

* **High Cost:** The hardware and software requirement of AI is very costly as it requires lots of maintenance to meet current world requirements.
* **Can't think out of the box:** Even we are making smarter machines with AI, but still they cannot work out of the box, as the robot will only do that work for which they are trained, or programmed.
* **No feelings and emotions:** AI machines can be an outstanding performer, but still it does not have the feeling so it cannot make any kind of emotional attachment with human, and may sometime be harmful for users if the proper care is not taken.
* **Increase dependency on machines:** With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.
* **No Original Creativity:** As humans are so creative and can imagine some new ideas but still AI machines cannot beat this power of human intelligence and cannot be creative and imaginative.

**Application of AI**

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:

1. AI in Astronomy

* Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

2. AI in Healthcare

* In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.
* Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

3. AI in Gaming

* AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

4. AI in Finance

* AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

5. AI in Data Security

* The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform,are used to determine software bug and cyber-attacks in a better way.

6. AI in Social Media

* Social Media sites such as Facebook, Twitter, and Snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hashtag, and requirement of different users.

7. AI in Travel & Transport

* AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

8. AI in Automotive Industry

* Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.
* Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

9. AI in Robotics:

* Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.
* Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

10. AI in Entertainment

* We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recommendations for programs or shows.

11. AI in Agriculture

* Agriculture is an area which requires various resources, labor, money, and time for best result. Now a day's agriculture is becoming digital, and AI is emerging in this field. Agriculture is applying AI as agriculture robotics, solid and crop monitoring, predictive analysis. AI in agriculture can be very helpful for farmers.

12. AI in E-commerce

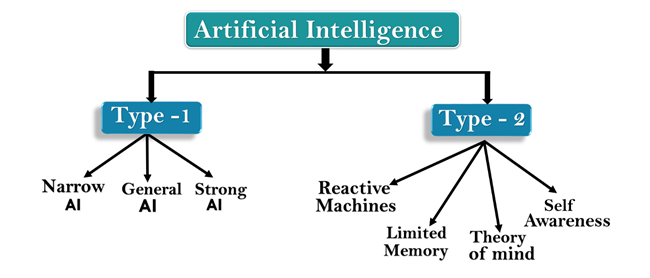
* AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

13. AI in education:

* AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.
* AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

# Types of Artificial Intelligence:

Artificial Intelligence can be divided in various types, there are mainly two types of main categorization which are based on capabilities and based on functionally of AI. Following is flow diagram which explain the types of AI.



## AI type-1: Based on Capabilities

### 1. Weak AI or Narrow AI:

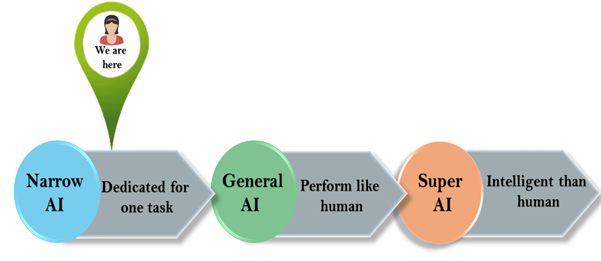
* Narrow AI is a type of AI which is able to perform a dedicated task with intelligence.The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
* Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
* Apple Siriis a good example of Narrow AI, but it operates with a limited pre-defined range of functions.
* IBM's Watson supercomputer also comes under Narrow AI, as it uses an Expert system approach combined with Machine learning and natural language processing.
* Some Examples of Narrow AI are playing chess, purchasing suggestions on e-commerce site, self-driving cars, speech recognition, and image recognition.

### 2. General AI:

* General AI is a type of intelligence which could perform any intellectual task with efficiency like a human.
* The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
* Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
* The worldwide researchers are now focused on developing machines with General AI.
* As systems with general AI are still under research, and it will take lots of efforts and time to develop such systems.

### 3. Super AI:

* Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI.
* Some key characteristics of strong AI include capability include the ability to think, to reason,solve the puzzle, make judgments, plan, learn, and communicate by its own.
* Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.



## Artificial Intelligence type-2: Based on functionality

### 1. Reactive Machines

* Purely reactive machines are the most basic types of Artificial Intelligence.
* Such AI systems do not store memories or past experiences for future actions.
* These machines only focus on current scenarios and react on it as per possible best action.
* IBM's Deep Blue system is an example of reactive machines.
* Google's AlphaGo is also an example of reactive machines.

### 2. Limited Memory

* Limited memory machines can store past experiences or some data for a short period of time.
* These machines can use stored data for a limited time period only.
* Self-driving cars are one of the best examples of Limited Memory systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road.

### 3. Theory of Mind

* Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.
* This type of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.

### 4. Self-Awareness

* Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.
* These machines will be smarter than human mind.
* Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

**Strong AI Vs. Weak AI**

**Strong AI** is artificial intelligence that matches or exceeds human intelligence — the intelligence of a machine that can successfully perform any intellectual task that a human being can.

* + It is a primary goal of artificial intelligence research and an important topic for science fiction writers and futurists.
  + Strong AI is also referred to as "artificial general intelligence" or as the ability to perform "general intelligent action".
  + Science fiction associates strong AI with such human traits as consciousness, sentience, sapience and self-awareness.

**Weak AI** is an artificial intelligence system which is not intended to match or exceed the capabilities of human beings, as opposed to strong AI, which is. Also known as applied AI or narrow AI.

* + The weak AI hypothesis: the philosophical position that machines can demonstrate intelligence, but do not necessarily have a mind, mental states or consciousness. (See philosophy of artificial intelligence or John Searle's definition of Strong AI in Chinese Room)

**AI Problems**

* AI has much more requirements in the problem areas like perception( vision and speech), natural language understanding and problem solving in specialized domains such as medical diagnosis and chemical analysis.
* Perception tasks are difficult because they involves analog signals, the signals are typically very noisy and usually a large no. of things must be perceived at once.
* Natural language understanding is also difficult in order to understand sentence about a topic it is necessary to know not only a lot about the language itself(its vocabulary and grammar) but also a good deal about the topic, so that understand assumption can be recognized.
* The problem areas of AI is classified into
  1. Mundane tasks
  2. Formal tasks
  3. Expert tasks

### Mundane Tasks:

Mundane tasks include:  
(i) Perception of vision and speech  
(ii) Natural Language understanding  
(iii) Common Sense Reasoning  
(iv) Robotics

### Formal Tasks:

Formal Tasks include:  
(i) Mathematics – Geometry, calculus, etc.  
(ii) Game Playing – Chess, 8-queens problems, water-jug problem, etc.

### Expert Tasks:

Expert Tasks include:  
(i) Engineering Field  
(ii) Scientific Analysis  
(iii) Medical and Financial Analysis

* Humans learn **mundane (ordinary) tasks** since their birth. They learn by perception, speaking, using language, and locomotives. They learn Formal Tasks and Expert Tasks later, in that order.
* For humans, the mundane tasks are easiest to learn. The same was considered true before trying to implement mundane tasks in machines.
* Earlier, all work of AI was concentrated in the mundane task domain.
* Later, it turned out that the machine requires more knowledge, complex knowledge representation, and complicated algorithms for handling mundane tasks.
* This is the reason **why AI work is more prospering in the Expert Tasks domain** now, as the expert task domain needs expert knowledge without common sense, which can be easier to represent and handle.

**AI (Artificial Intelligence) Techniques**

AI technique is a method that exploits knowledge that should be represented in such a way that:  
1. It captures generalizations.  
2. It can be understood by people who must provide it.  
3. It can be easily modified to correct errors and to incorporate changes.  
4. It can be used in many situations even if it is not accurate or complete.  
5. It can be used to help overcome its sheer bulk by helping to narrow the range of possibilities that must usually be considered.

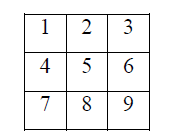
In order to characterize an AI technique let us consider initially OXO or tic-tac-toe and use a series of different approaches to play the game.

* The programs increase in complexity, their use of generalizations, the clarity of their knowledge and the extensibility of their approach.
* In this way they move towards being representations of AI techniques.

**Example-1: Tic-Tac-Toe**

The first approach (simple)

* The Tic-Tac-Toe game consists of a nine element vector called BOARD; it represents the numbers 1 to 9 in three rows.
* An element contains the value 0 for blank, 1 for X and 2 for O. A MOVETABLE vector consists of 19,683 elements (39) and is needed where each element is a nine element vector.
* The contents of the vector are especially chosen to help the algorithm.

****

The algorithm makes moves by pursuing the following:

* + View the vector as a ternary number. Convert it to a decimal number.
  + Use the decimal number as an index in MOVETABLE and access the vector.
  + Set BOARD to this vector indicating how the board looks after the move. This approach is capable in time but it has several disadvantages. It takes more space and requires stunning effort to calculate the decimal numbers. This method is specific to this game and cannot be completed.

**The second approach**

* The structure of the data is as before but we use 2 for a blank, 3 for an X and 5 for an O.
* A variable called TURN indicates 1 for the first move and 9 for the last. The algorithm consists of three actions:
* MAKE2 which returns 5 if the centre square is blank; otherwise it returns any blank noncorner square, i.e. 2, 4, 6 or 8. POSSWIN (p) returns 0 if player p cannot win on the next move and otherwise returns the number of the square that gives a winning move.
* It checks each line using products 3\*3\*2 = 18 gives a win for X, 5\*5\*2=50 gives a win for O, and the winning move is the holder of the blank. GO (n) makes a move to square n setting BOARD[n] to 3 or 5.
* This algorithm is more involved and takes longer but it is more efficient in storage which compensates for its longer time. It depends on the programmer’s skill.

**The final approach**

* The structure of the data consists of BOARD which contains a nine element vector, a list of board positions that could result from the next move and a number representing an estimation of how the board position leads to an ultimate win for the player to move.
* This algorithm looks ahead to make a decision on the next move by deciding which the most promising move or the most suitable move at any stage would be and selects the same.
* Consider all possible moves and replies that the program can make. Continue this process for as long as time permits until a winner emerges, and then choose the move that leads to the computer program winning, if possible in the shortest time.
* Actually this is most difficult to program by a good limit but it is as far that the technique can be extended to in any game. This method makes relatively fewer loads on the programmer in terms of the game technique but the overall game strategy must be known to the adviser.

**Example-2: Question Answering**

* Let us consider Question Answering systems that accept input in English and provide answers also in English. This problem is harder than the previous one as it is more difficult to specify the problem properly. Another area of difficulty concerns deciding whether the answer obtained is correct, or not, and further what is meant by ‘correct’.
* For example, consider the following situation

**Text**

* Rani went shopping for a new Coat. She found a red one she really liked.
* When she got home, she found that it went perfectly with her favorite dress.
* **Question**

1. What did Rani go shopping for?

2. What did Rani find that she liked?

3. Did Rani buy anything?

**Method 1**

**Data Structures**

* A set of templates that match common questions and produce patterns used to match against inputs. Templates and patterns are used so that a template that matches a given question is associated with the corresponding pattern to find the answer in the input text. For example, the template who did **x y generates x y z if a match occurs and z is the answer to the question.** The given text and the question are both stored as strings.

**Algorithm**

Answering a question requires the following four steps to be followed:

* + Compare the template against the questions and store all successful matches to produce a set of text patterns.
  + Pass these text patterns through a substitution process to change the person or voice and produce an expanded set of text patterns.
  + Apply each of these patterns to the text; collect all the answers and then print the answers.

**Example**

* In question 1 we use the template WHAT DID X Y which generates Rani go shopping for **z** and after substitution we get Rani goes shopping for **z** and Rani went shopping for **z** giving **z** [equivalence] a new coat
* In question 2 we need a very large number of templates and also a scheme to allow the insertion of ‘find’ before ‘that she liked’; the insertion of ‘really’ in the text; and the substitution of ‘she’ for ‘Rani’ gives the answer ‘a red one’.
* Question 3 cannot be answered.

**Comments**

* This is a very primitive approach basically not matching the criteria we set for intelligence and worse than that, used in the game. Surprisingly this type of technique was actually used in ELIZA which will be considered later in the course.

**Method 2**

**Data Structures**

* A structure called English consists of a dictionary, grammar and some semantics about the vocabulary we are likely to come across. This data structure provides the knowledge to convert English text into a storable internal form and also to convert the response back into English.
* The structured representation of the text is a processed form and defines the context of the input text by making explicit all references such as pronouns.
* There are three types of such
* *knowledge representation systems: production rules of the form ‘if x then y’, slot and filler* systems and statements in mathematical logic.
* The system used here will be the slot and filler system.

Take, for example sentence:

**‘She found a red one she really liked’.**

* **Event2** Event2
  + instance: finding instance: liking
  + tense: past tense: past
  + agent: Rani modifier: much
  + object: Thing1 object: Thing1
* **Thing1**
  + instance: coat
  + colour: red
* The question is stored in two forms: as input and in the above form.
* **Algorithm**
  + Convert the question to a structured form using English know how, then use a marker to indicate the substring (like ‘who’ or ‘what’) of the structure, that should be returned as an answer. If a slot and filler system is used a special marker can be placed in more than one slot.
  + The answer appears by matching this structured form against the structured text.
  + The structured form is matched against the text and the requested segments of the question are returned.

**Examples**

* Both questions 1 and 2 generate answers via a new coat and a red coat respectively.
* Question 3 cannot be answered, because there is no direct response.

**Comments**

* This approach is more meaningful than the previous one and so is more effective. The extra power given must be paid for by additional search time in the knowledge bases. A warning must be given here: that is – to generate unambiguous English knowledge base is a complex task and must be left until later in the course. The problems of handling pronouns are difficult.

For example:

* **Rani walked up to the salesperson: she asked where the toy department was.**
* **Rani walked up to the salesperson: she asked her if she needed any help.**
* Whereas in the original text the linkage of ‘she’ to ‘Rani’ is easy, linkage of ‘she’ in each of the above sentences to Rani and to the salesperson requires additional knowledge about the context via the people in a shop.

**Problem, Problem spaces and Search**

To solve the problem of building a system you should take the following steps:

1. Define the problem accurately including detailed specifications and what constitutes a

suitable solution.

2. Scrutinize the problem carefully, for some features may have a central affect on the chosen

method of solution.

3. Segregate and represent the background knowledge needed in the solution of the problem.

4. Choose the best solving techniques for the problem to solve a solution.

***Problem solving* is a process** of generating solutions from observed data.

• a *‘problem’* is characterized by a set of *goals*,

• a set of *objects*, and

• a set of *operations*.

These could be ill-defined and may evolve during problem solving.

• A **‘*problem space*’** is an abstract space.

 A problem space encompasses all *valid states* that can be generated by the application of any

combination of *operators* on any combination of *objects*.

 The problem space may contain one or more *solutions*. A solution is a combination of

*operations* and *objects* that achieve the *goals*.

• A ‘***search***’ refers to the search for a solution in a problem space.

 Search proceeds with different types of ‘*search control strategies*’.

 The *depth-first search and breadth-first search* are the two common *search strategies.*

**2.1 AI - General Problem Solving**

*Problem solving* has been the key area of concern for Artificial Intelligence.

Problem solving is a process of generating solutions from observed or given data. It is however

not always possible to use direct methods (i.e. go directly from data to solution). Instead,

problem solving often needs to use indirect or modelbased methods.

***General Problem Solver (GPS)*** was a computer program created in 1957 by Simon and Newell

to build a universal problem solver machine. *GPS* was based on Simon and Newell’s theoretical

work on logic machines. *GPS* in principle can solve any formalized symbolic problem, such as

theorems proof and geometric problems and chess playing. *GPS* solved many simple problems,

such as the Towers of Hanoi, that could be sufficiently formalized, but ***GPS could not solve any***

***real-world problems***.

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

 Define the problem precisely – find input situations as well as final situations for an acceptable solution to the problem

 Analyze the problem – find few important features that may have impact on the

oppropriateness of various possible techniques for solving the problem

 Isolate and represent task knowledge necessary to solve the problem

 Choose the best problem-solving technique(s) and apply to the particular problem

**Problem definitions**

A problem is defined by its ‘*elements*’ and their ‘*relations*’. To provide a formal description of a

problem, we need to do the following:

a. Define a *state space* that contains all the possible configurations of the relevant objects,

including some impossible ons.

b. Specify one or more states that describe possible situations, from which the problem solving

process may start. These states are called *initial states*.

c. Specify one or more states that would be acceptable solution to the problem. These states are

called *goal states*.

d. Specify a set of *rules* that describe the actions (*operators*) available.

The problem can then be solved by using the *rules*, in combination with an appropriate *control*

*strategy*, to move through the *problem space* until a *path* from an *initial state* to a *goal state* is

found. This process is known as ***‘search’****.* Thus:

 *Search* is fundamental to the problem-solving process.

 *Search* is a general mechanism that can be used when a more direct method is not known.

 *Search* provides the framework into which more direct methods for solving subparts of a problem can be embedded. A very large number of AI problems are formulated as search problems.

 Problem space

A *problem space* is represented by a directed graph, where *nodes* represent search state and *paths*

represent the operators applied to change the *state*.

To simplify search algorithms, it is often convenient to logically and programmatically represent

a problem space as a **tree**. A *tree* usually decreases the complexity of a search at a cost. Here, the

cost is due to duplicating some nodes on the tree that were linked numerous times in the graph,

e.g. node ***B*** and node ***D.***

A *tree is a graph* in which any two vertices are connected by exactly one path. Alternatively, any

connected *graph with no cycles is a tree*.

**DEFINING PROBLEM AS A STATE SPACE SEARCH**

To solve the problem of playing a game, we require the rules of the game and targets for winning

as well as representing positions in the game. The opening position can be defined as the initial state and a winning position as a goal state. Moves from initial state to other states leading to the goal state follow legally. However, the rules are far too abundant in most games— especially in chess, where they exceed the number of particles in the universe. Thus, the rules cannot be supplied accurately and computer programs cannot handle easily. The storage also presents another problem but searching can be achieved by hashing.

The number of rules that are used must be minimized and the set can be created by expressing each rule in a form as possible. The representation of games leads to a state space representation and it is common for well-organized games with some structure. This representation allows for the formal definition of a problem that needs the movement from a set of initial positions to one of a set of target positions. It means that the solution involves using known techniques and a systematic search. This is quite a common method in Artificial Intelligence.

**State Space Search**

A *state space* represents a problem in terms of *states* and *operators* that change states.A state space consists of:

A representation of the *states* the system can be in. For example, in a board game, the board

represents the current state of the game.

A set of *operators* that can change one state into another state. In a board game, the operators

are the legal moves from any given state. Often the operators are represented as programs that change a state representation to represent the new state.

An *initial state*.

A set of *final states*; some of these may be desirable, others undesirable.

This set is often represented implicitly by a program that detects terminal states.

**The Water Jug Problem**

In this problem, we use two jugs called **four** and **three;** four holds a maximum of four gallons of

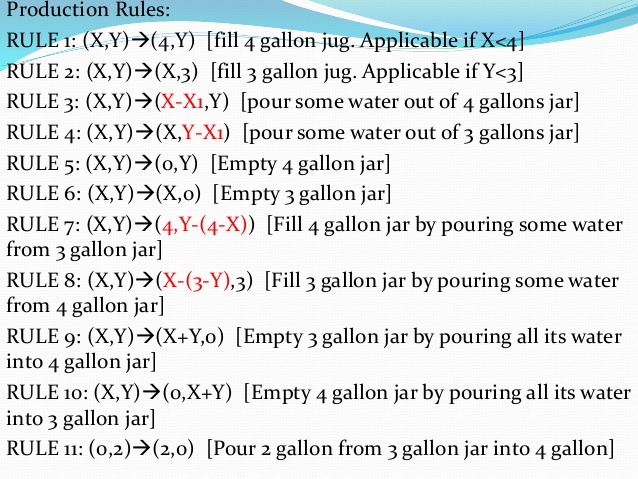
water and **three** a maximum of three gallons of water. How can we get two gallons of water in the **four** jug?

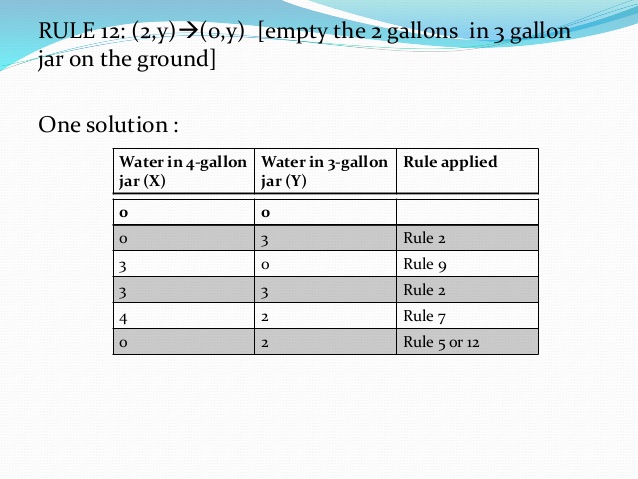
The state space is a set of prearranged pairs giving the number of gallons of water in the pair of

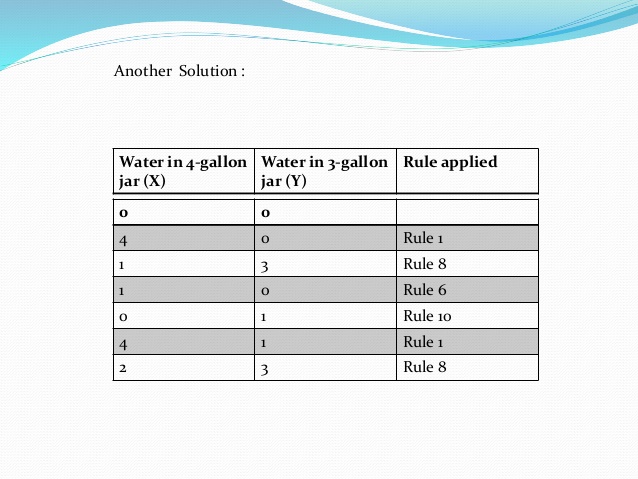
jugs at any time, i.e., (**four, three**) where **fou**r = 0, 1, 2, 3 or 4 and **three** = 0, 1, 2 or 3.

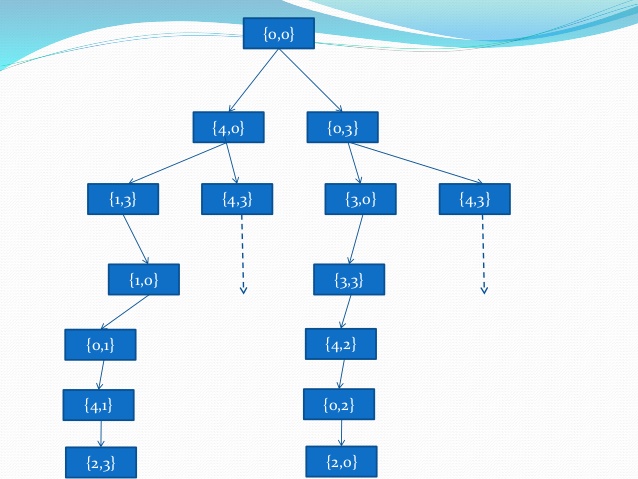
The start state is (0, 0) and the goal state is (2, n) .

where n may be any but it is limited to **three** holding from 0 to 3 gallons of water or empty. Three and four shows the name and numerical number shows the amount of water in jugs for solving the water jug problem. The major production rules for solving this problem are shown below:









The problem solved by using the production rules in combination with an appropriate control

strategy, moving through the problem space until a path from an initial state to a goal state is

found. In this problem solving process, search is the fundamental concept. For simple problems.

it is easier to achieve this goal by hand but there will be cases where this is far too difficult.

**PRODUCTION SYSTEMS**

Production systems can be defined as a kind of cognitive architecture, in which knowledge is represented in the form of rules. So, a  system that uses this form of knowledge representation is called a production system. To simply put, production systems consists of rules and factors. Knowledge is usually encoded in a declarative from which comprises of a set of rules of the form.

The global database is the central data structure which used by an AI production system. The production system. The production rules operate on the global database. Each rule usually has a precondition that is either satisfied or not by the global database. If the precondition is satisfied, the rule is usually be applied. Application of the rule changes the database. The control system then chooses which applicable rule should be applied and ceases computation when a termination condition on the database is satisfied. If multiple rules are to fire at the same time, the control system resolves the conflicts.

#### Major Features Of Production System

* Simplicity
* Modularity
* Modifiability
* Knowledge Intensive

#### <https://www.analyticsindiamag.com/wp-content/uploads/2019/01/Screenshot-54.jpg>

#### Advantages Of Production Systems In AI

* Provides excellent tools for structuring AI programs
* The system is highly modular because individual rules can be added, removed or modified independently
* Expressed in natural form.
* Separation of knowledge and Control – Recognises Act Cycle
* A natural mapping onto state space research – data or goal-driven
* Modularity of production rules
* The system uses pattern directed control which is more flexible than algorithmic control
* Provides opportunities for heuristic control of search
* Tracing and Explanation – Simple Control, Informative rules
* Language Independence
* A plausible model of human problem solving -SOAR, ACT
* A good way to model the state-driven nature of intelligent machines
* Quite helpful in real time in environment and applications.

#### Disadvantages Of Production Systems In AI

* It’s very difficult to analyse the flow of control within a production system
* It describes the operations that can be performed in a search for a solution to the problem. They can be classified as follows.
* There is an absence of learning due to a rule-based production system which does not store the result of the problem for future use.
* The rules in the production system should not have any type of conflict resolution as when a new rule is added to the database it should ensure that it does not have any conflict with any existing rules.

Production systems provide appropriate structures for performing and describing search

processes. A production system has four basic components as enumerated below.

A set of rules each consisting of a left side that determines the applicability of the rule and a

right side that describes the operation to be performed if the rule is applied.

A database of current facts established during the process of inference.

A control strategy that specifies the order in which the rules will be compared with facts in the

database and also specifies how to resolve conflicts in selection of several rules or selection of

more facts.

A rule firing module.

The production rules operate on the knowledge database. Each rule has a precondition—that is,

either satisfied or not by the knowledge database. If the precondition is satisfied, the rule can be

applied. Application of the rule changes the knowledge database. The control system chooses

which applicable rule should be applied and ceases computation when a termination condition on

the knowledge database is satisfied.

**Example: Eight puzzle (8-Puzzle)**

The 8-puzzle is a 3 × 3 array containing eight square pieces, numbered 1 through 8, and

one empty space. A piece can be moved horizontally or vertically into the empty space, in effect

exchanging the positions of the piece and the empty space. There are four possible moves, UP

(move the blank space up), DOWN, LEFT and RIGHT. The aim of the game is to make a

sequence of moves that will convert the board from the start state into the goal state:



This example can be solved by the operator sequence UP, RIGHT, UP, LEFT, DOWN.

**CONTROL STRATAGY**

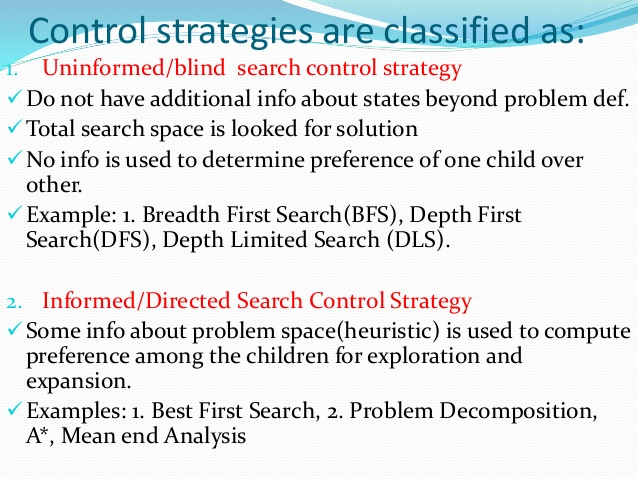
We will now consider the problem of deciding which rule to apply next during the process of searching for a solution for a solution. This question arises when more than one rule will have its left side match the current state.

**The first requirement of a control strategy is that it must cause motion**.

**The second requirement of a control strategy is that issue must be systematic**.

We will explain these two with respect to water jug problem. If we have implemented choosing the first operator and then the one which matches the first one, then we would not have solved the problem. If we follow any strategy which can cause some motion then will lead to a solution. But if it is not followed systematically , and then got the solution. One day to follow a systematic control strategy is to construct a tree with the initial state as its root. By applying all possible combinations from the first level leaf nodes. Continue the process until some rule produces a goal state. For the water jug problem a tree can be constructed as given in following diagram.

The control strategy for the search process is called breadth first search. Other systematical control strategies are also available . for example, we can select one single branch of a tree until it yields a solution or until some pre specified depth has been reached. If not we go back and explore to other branches . this is called depth – first – search. The water jug problems will lead to an answer by adoption any control strategy because the problem is simple. This is not always the case.

****

## Breadth-first Search:

* Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
* BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
* The breadth-first search algorithm is an example of a general-graph search algorithm.
* Breadth-first search implemented using FIFO queue data structure.

**Advantages:**

* BFS will provide a solution if any solution exists.
* If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

**Disadvantages:**

* It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
* BFS needs lots of time if the solution is far away from the root node.

## Depth-first Search

* Depth-first search isa recursive algorithm for traversing a tree or graph data structure.
* It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
* DFS uses a stack data structure for its implementation.
* The process of the DFS algorithm is similar to the BFS algorithm.

**Advantage:**

* DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
* It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

**Disadvantage:**

* There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
* DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

**PROBLEM CHARACTERISTICS**

Heuristic search is a very general method applicable to a large class of problem . It includes a variety of techniques. In order to choose an appropriate method, it is necessary to analyze the problem with respect to the following considerations.

1. **Is the problem decomposable ?**

A very large and composite problem can be easily solved if it can be broken into smaller problems and recursion could be used. Suppose we want to solve.

Ex:- ∫ x2 + 3x+sin2x cos 2x dx

This can be done by breaking it into three smaller problems and solving each by applying specific rules. Adding the results the complete solution is obtained.

**2. Can solution steps be ignored or undone?**

Problem fall under three classes ignorable , recoverable and irrecoverable. This classification is with reference to the steps of the solution to a problem. Consider thermo proving. We may later find that it is of no help. We can still proceed further, since nothing is lost by this redundant step. This is an example of ignorable solutions steps.

Now consider the 8 puzzle problem tray and arranged in specified order. While moving from the start state towards goal state, we may make some stupid move and consider theorem proving. We may proceed by first proving lemma. But we may backtrack and undo the unwanted move. This only involves additional steps and the solution steps are recoverable.

Lastly consider the game of chess. If a wrong move is made, it can neither be ignored nor be recovered. The thing to do is to make the best use of current situation and proceed. This is an example of an irrecoverable solution steps.

1. Ignorable problems Ex:- theorem proving

                                         · In which solution steps can be ignored.

2. Recoverable problems Ex:- 8 puzzle

                                         · In which solution steps can be undone

3. Irrecoverable problems Ex:- Chess

                                         · In which solution steps can’t be undone

A knowledge of these will help in determining the control structure.

**3.. Is the Universal Predictable?**

Problems can be classified into those with certain outcome (eight puzzle and water jug problems) and those with uncertain outcome ( playing cards) . in certain – outcome problems, planning could be done to generate a sequence of operators that guarantees to a lead to a solution. Planning helps to avoid unwanted solution steps. For uncertain out come problems, planning can at best generate a sequence of operators that has a good probability of leading to a solution. The uncertain outcome problems do not guarantee a solution and it is often very expensive since the number of solution and it is often very expensive since the number of solution paths to be explored increases exponentially with the number of points at which the outcome can not be predicted. Thus one of the hardest types of problems to solve is the irrecoverable, uncertain – outcome problems ( Ex:- Playing cards).

**4. Is good solution absolute or relative ?**

                                            (Is the solution a state or a path ?)

There are two categories of problems. In one, like the water jug and 8 puzzle problems, we are satisfied with the solution, unmindful of the solution path taken, whereas in the other category not just any solution is acceptable. We want the best, like that of traveling sales man problem, where it is the shortest path. In any – path problems, by heuristic methods we obtain a solution and we do not explore alternatives. For the best-path problems all possible paths are explored using an exhaustive search until the best path is obtained.

**5. The knowledge base consistent ?**

In some problems the knowledge base is consistent and in some it is not. For example consider the case when a Boolean expression is evaluated. The knowledge base now contains theorems and laws of Boolean Algebra which are always true. On the contrary consider a knowledge base that contains facts about production and cost. These keep varying with time. Hence many reasoning schemes that work well in consistent domains are not appropriate in inconsistent domains.

Ex.Boolean expression evaluation.

**6. What is the role of Knowledge?**

Though one could have unlimited computing power, the size of the knowledge base available for solving the problem does matter in arriving at a good solution. Take for example the game of playing chess, just the rues for determining legal moves and some simple control mechanism is sufficient to arrive at a solution. But additional knowledge about good strategy and tactics could help to constrain the search and speed up the execution of the program. The solution would then be realistic.

Consider the case of predicting the political trend. This would require an enormous amount of knowledge even to be able to recognize a solution , leave alone the best.

Ex:- 1. Playing chess 2. News paper understanding

**7. Does the task requires interaction with the person.**

The problems can again be categorized under two heads.

1. Solitary in which the computer will be given a problem description and will produce an answer, with no intermediate communication and with he demand for an explanation of the reasoning process. Simple theorem proving falls under this category . given the basic rules and laws, the theorem could be proved, if one exists.

Ex:- theorem proving (give basic rules & laws to computer)

2. Conversational, in which there will be intermediate communication between a person and the computer, wither to provide additional assistance to the computer or to provide additional informed information to the user, or both problems such as medical diagnosis fall under this category, where people will be unwilling to accept the verdict of the program, if they can not follow its reasoning.

Ex:- Problems such as medical diagnosis.

**Heuristic Search**

The term Heuristic is used for algorithms which find solutions among all possible one. The Heuristic is a rule of thumb or judgement technique that leads to a solution but it provides no guarantee of success.

Heuristic plays an important role in the searching process because they help to reduce the number of alternatives from an exponential number to a polynomial number.

So, we get a solution in a reasonable amount of time. The additional information about the properties of the specific domain which is built into the state and operator definitions is called Heuristic Information. A search using this Heuristic Information is called Heuristic Search or Informed Search.

* **Heuristic**search is an informed search technique. In this, the algorithm is aware of where the best chances of finding the element are and the algorithm heads that way!
* Solve complex problems efficiently ,it is necessary to compromise the requirements of the movability and systematically.
* A control structure has to be constructed that no longer guarantees the best solution, but that will almost always find a very good answer. Such a technique is said to be heuristic (rule of thumb).
* A heuristic search improves the efficiently of the search process, but sacrifices the claims of completeness.
* But they improve the quality of the paths that are explored. Using good heuristics we can get good solutions to hard problems, such as the traveling salesman problem.

Applying it to the traveling salesman problem produces the following procedure

Applying it to the traveling salesman problem produces the following procedure.

 1. Arbitrarily select a starting city.

 2. To select the next city, look at all cities not yet visited. Select the one closet to the current city.      Go to it next.

3. Repeat step 2 until all the cities have been visited.

* This procedure executes in time proportional to N \* N , instead of N! and it is possible to prove an upper bound on the error it incurs. In many AI problems , however, it is not possible to produce such bounds. This is true for two reasons.

i) For real world problems, it is often hard to measure precisely the goodness of a particular solution. For instance , answers to questions like “Why has inflation increased?” can not be precise.

 ii) For real world problems it is often useful to introduce heuristics based on relatively unstructured knowledge.

* This is because often a mathematical analysis is not possible. Without heuristics, it is not possible to tackle combinatorial explosion.
* Moreover, we go for optimum solution that satisfy some set of requirements. We stop with satisfactory solutions even though there might be better solutions.
* The heuristic function is used to generate this heuristic value.
* Different heuristic functions can be designed depending on the searching problem.
* So we can conclude that **Heuristic search is a technique that uses a heuristic value for optimizing the search.**

**Example** : A Game of 8-puzzle

* **State space:** Configuration of 8 tiles on the board.
* Initial state: Any configuration
* Goal state: Tiles in a specific order
* Action: Blank moves
  + Condition: the move is with in the board
  + Transformation: Blank moves, left,right,up,down
* In our 8-Puzzle problem, we can define the **h-score** as the number of misplaced tiles by comparing the current state and the goal state.
* **g-score** will remain as the number of nodes traversed from a start node to get to the current node.
* From Figure, we can calculate the **h-score** by comparing the initial(current) state and goal state and counting the number of misplaced tiles.  
  Thus, **h-score** = 5 and **g-score**= 0 as the number of nodes traversed from the start node to the current node is 0



**HEURISTIC FUNCTIONS**

* A Heuristic technique helps in solving problems, even though there is no guarantee that it will never lead in the wrong direction. There are heuristics of every general applicability as well as domain specific.
* The strategies are general purpose heuristics. In order to use them in a specific domain they are coupler with some domain specific heuristics.
* There are two major ways in which domain - specific, heuristic information can be incorporated into rule-based search procedure.

- In the rules themselves

 - As a heuristic function that evaluates individual problem states and determines how desired they are.

* A heuristic function is a function that maps from problem state description to measures desirability, usually represented as number weights.
* The value of a heuristic function at a given node in the search process gives a good estimate of that node being on the desired path to solution.
* Well designed heuristic functions can provides a fairly good estimate of whether a path is good or not. ( " The sum of the distances traveled so far" is a simple heuristic function in the traveling salesman problem) .
* The purpose of a heuristic function is to guide the search process in the most profitable directions, by suggesting which path to follow first when more than one path is available.
* That is the main objective of heuristic function is that we have to reach the final state with minimum cost i.e cost in terms of time and space.

**Heuristic Search Techniques**

* Many of the problems are too complex to be solvable by direct techniques. They have to be solved only by suitable heuristic search techniques.
* Though the heuristic techniques can be described independently; it requires domain specific information.
* Heuristics are *rules of thumb; they do not guarantee a solution to a problem.*
* Heuristic Search is a weak technique but can be effective if applied correctly.
* Heuristic techniques are called weak methods, since they are vulnerable to combinatorial explosion. Even then these techniques continue to provide framework into which domain specific knowledge can be placed, either by hand or as a result of learning.
* The following are some general purpose control strategies ( often called weak methods).

Generate - and – test, Hill climbing, Breadth - First search, Depth - First search, Best First Search (A\* search),Problem reduction(AO\* search),Constraint satisfaction, Means - ends analysis

* A heuristic procedure, or heuristic, is defined as having the following properties.  
    
  1. It will usually find good, although not necessary optimum solutions.

2. It is faster and easier to implement than any known exact algorithm  ( one which guarantees an optimum solution ).

* In general, heuristic search improve the quality of the path that are exported.
* Using good heuristics we can hope to get good solutions to hard problems such as the traveling salesman problem in less than exponential time.
* There are some good general purpose heuristics that are useful in a wide variety of problems. It is also possible to construct special purpose heuristics to solve particular problems.

**1. GENERATE –AND- TEST**

This is the simplest search strategy. It consists of the following steps;

**Algorithm**

1. Generating a possible solution for some problems; this means generating a particular point in the problem space. For others it may be generating a path from a start state.

2. Test to see if this is actually a solution by comparing the chosen point at the end point of the chosen path to the set of acceptable goal states.

3. If a solution has been found, quit otherwise return to step 1.

* The generate - and - Test algorithm is a depth first search procedure because complete possible solutions are generated before test.
* This can be implemented states are likely to appear often in a tree; it can be implemented on a search graph rather than a tree.

**2. Hill Climbing Algorithm**

* 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:

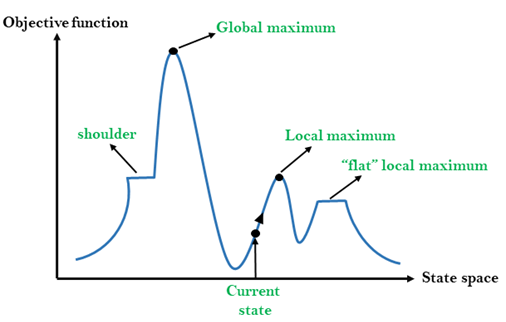
Following are some main features of Hill Climbing Algorithm:

* **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.

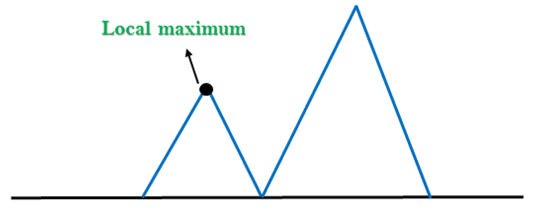
### 3. Stochastic hill climbing:

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:

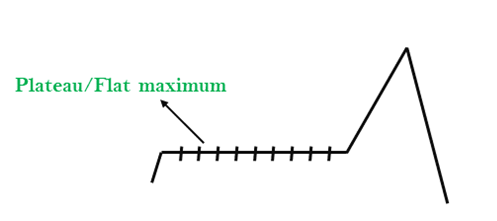
**1. Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

**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.



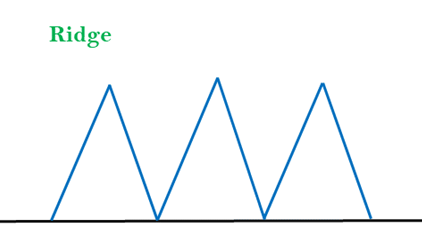
**2. Plateau:** A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

**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.



### 3.Best-first Search Algorithm (Greedy Search):

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e.

1. f(n)= g(n).

Were, h(n)= estimated cost from node n to the goal.

The greedy best first algorithm is implemented by the priority queue.

### Best first search algorithm:

* **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:

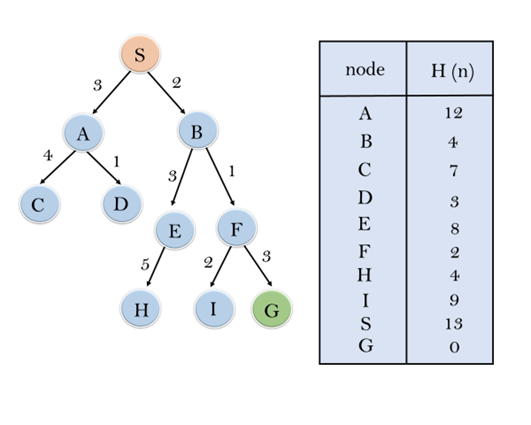
* Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
* This algorithm is more efficient than BFS and DFS algorithms.

### Disadvantages:

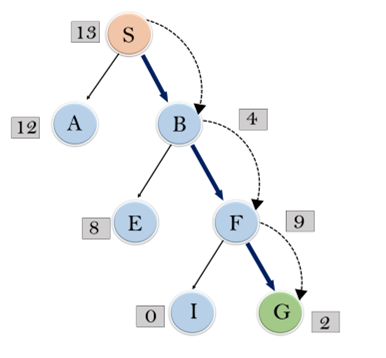
* It can behave as an unguided depth-first search in the worst case scenario.
* It can get stuck in a loop as DFS.
* This algorithm is not optimal.

### Example:

Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.



In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.



**Expand the nodes of S and put in the CLOSED list**

**Initialization:** Open [A, B], Closed [S]

**Iteration 1:** Open [A], Closed [S, B]

**Iteration 2:** Open [E, F, A], Closed [S, B]  
                  : Open [E, A], Closed [S, B, F]

**Iteration 3:** Open [I, G, E, A], Closed [S, B, F]  
                  : Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F----> G**

**Time Complexity:** The worst case time complexity of Greedy best first search is O(bm).

**Space Complexity:** The worst case space complexity of Greedy best first search is O(bm). Where, m is the maximum depth of the search space.

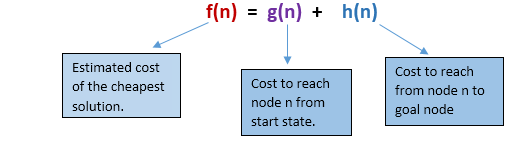
**Complete:** Greedy best-first search is also incomplete, even if the given state space is finite.

**Optimal:** Greedy best first search algorithm is not optimal.

### 4.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). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. 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).

In A\* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.



At each point in the search space, only those node is expanded which have the lowest value of f(n), and the algorithm terminates when the goal node is found.

### 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.

# 5.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.

## 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.

## Operator Subgoaling

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

## Algorithm for Means-Ends Analysis:

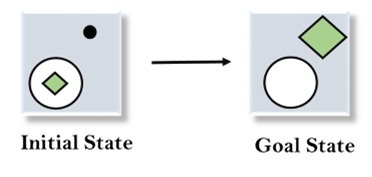
Let's we take Current state as CURRENT and Goal State as GOAL, then following are the steps for the MEA algorithm.

* **Step 1:** Compare CURRENT to GOAL, if there are no differences between both then return Success and Exit.
* **Step 2:** Else, select the most significant difference and reduce it by doing the following steps until the success or failure occurs.
  1. Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.
  2. 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.
  3. 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.

The above-discussed algorithm is more suitable for a simple problem and not adequate for solving complex problems.

### 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:

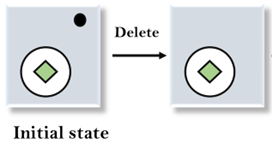
To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

* **Move**
* **Delete**
* **Expand**

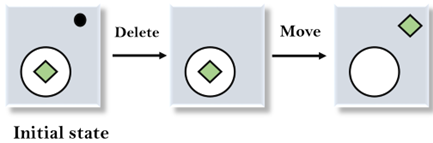
**1. Evaluating the initial state:** In the first step, we will evaluate the initial state and will compare the initial and Goal state to find the differences between both states.



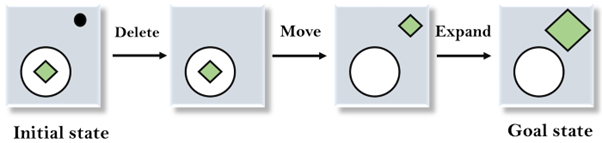
**2. Applying Delete operator:** As we can check the first difference is that in goal state there is no dot symbol which is present in the initial state, so, first we will apply the **Delete operator** to remove this dot.



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



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



**6. Problem Reduction**

**PROBLEM REDUCTION ( AND - OR graphs - AO \* Algorithm)**

So far we have considered search strategies for OR graphs through which we want to find a single path to a goal. Such structure represent the fact that we know how to get from anode to a goal state if we can discover how to get from that node to a goal state along any one of the branches leaving it.

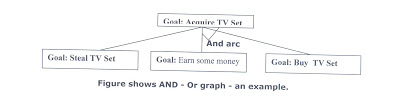
**Problem Reduction algorithm:**

1. Initialize the graph to the starting node.  
2. Loop until the starting node is labelled **SOLVED** or until its cost goes above **FUTILITY**:  
(i) Traverse the graph, starting at the initial node and following the current best path and accumulate the set of nodes that are on that path and have not yet been expanded.  
(ii) Pick one of these unexpanded nodes and expand it. If there are no successors, assign FUTILITY as the value of this node. Otherwise, add its successors to the graph and for each of them compute f'(n). If f'(n) of any node is O, mark that node as SOLVED.  
(iii) Change the f'(n) estimate of the newly expanded node to reflect the new information provided by its successors. Propagate this change backwards through the graph. If any node contains a successor arc whose descendants are all solved, label the node itself as SOLVED.

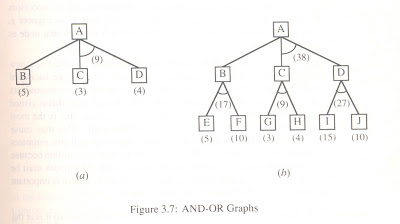
***AND-OR GRAPHS***

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

**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. One AND are may point to any number of successor nodes. All these must be solved so that the arc will rise to many arcs, indicating several possible solutions. Hence the graph is known as AND - OR instead of AND. Figure shows an AND - OR graph.**

[](http://1.bp.blogspot.com/_ZGzaqHb40vU/TEk4aFB9nkI/AAAAAAAAAGg/wk43hjjHOoc/s1600/BestFirstSearch1.jpg)

**An algorithm to find a solution in an AND - OR graph must handle AND area appropriately. A\* algorithm can not search AND - OR graphs efficiently. This can be understand from the give figure.**

[](http://4.bp.blogspot.com/_ZGzaqHb40vU/TEk1fyQdz8I/AAAAAAAAAGY/qiwz__Js55k/s1600/BestFirstSearch2.jpg)

**In figure (a) the top node A has been expanded producing two area one leading to B and leading to C-D . the numbers at each node represent the value of f ' at that node (cost of getting to the goal state from current state). For simplicity, it is assumed that every operation(i.e. applying a rule) has unit cost, i.e., each are with single successor will have a cost of 1 and each of its components. With the available information till now , it appears that C is the most promising node to expand since its f ' = 3 , the lowest but going through B would be better since to use C we must also use D' and the cost would be 9(3+4+1+1). Through B it would be 6(5+1).**

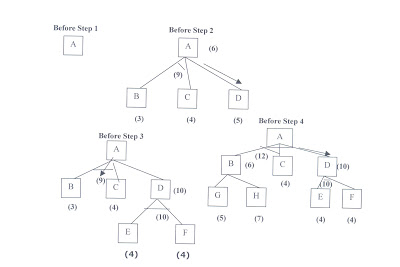
**Thus the choice of the next node to expand depends not only n a value but also on whether that node is part of the current best path form the initial mode. Figure (b) makes this clearer. In figure the node G appears to be the most promising node, with the least f ' value. But G is not on the current beat path, since to use G we must use GH with a cost of 9 and again this demands that arcs be used (with a cost of 27). The path from A through B, E-F is better with a total cost of (17+1=18). Thus we can see that to search an AND-OR graph, the following three things must be done.**

**1. traverse the graph starting at the initial node and following the current best path, and accumulate the set of nodes that are on the path and have not yet been expanded.**

**2. Pick one of these unexpanded nodes and expand it. Add its successors to the graph and computer f ' (cost of the remaining distance) for each of them.**

**3. Change the f ' estimate of the newly expanded node to reflect the new information produced by its successors. Propagate this change backward through the graph. Decide which of the current best path.**

**The propagation of revised cost estimation backward is in the tree is not necessary in A\* algorithm. This is because in AO\* algorithm expanded nodes are re-examined so that the current best path can be selected. The working of AO\* algorithm is illustrated in figure as follows:**

[](http://1.bp.blogspot.com/_ZGzaqHb40vU/TEkyIi1PzHI/AAAAAAAAAGQ/FwR_3J0S7qo/s1600/BestFirstSearch3.jpg)

**Referring the figure. The initial node is expanded and D is Marked initially as promising node. D is expanded producing an AND arc E-F. f ' value of D is updated to 10. Going backwards we can see that the AND arc B-C is better . it is now marked as current best path. B and C have to be expanded next. This process continues until a solution is found or all paths have led to dead ends, indicating that there is no solution. An A\* algorithm the path from one node to the other is always that of the lowest cost and it is independent of the paths through other nodes.**

**The algorithm for performing a heuristic search of an AND - OR graph is given below. Unlike A\* algorithm which used two lists OPEN and CLOSED, the AO\* algorithm uses a single structure G. G represents the part of the search graph generated so far. Each node in G points down to its immediate successors and up to its immediate predecessors, and also has with it the value of h' cost of a path from itself to a set of solution nodes. The cost of getting from the start nodes to the current node "g" is not stored as in the A\* algorithm. This is because it is not possible to compute a single such value since there may be many paths to the same state. In AO\* algorithm serves as the estimate of goodness of a node. Also a there should value called FUTILITY is used. The estimated cost of a solution is greater than FUTILITY then the search is abandoned as too expansive to be practical.**

**For representing above graphs AO\* algorithm is as follows**

**And-Or Graphs**

Useful for certain problems where

* The solution involves decomposing the problem into smaller problems.
* We then solve these smaller problems.

Here the alternatives often involve branches where some or all must be satisfied before we can progress.

For example if I want to learn to play a Frank Zappa guitar solo I could

***AO*\* Algorithm**

1. Initialise the graph to start node
2. Traverse the graph following the current path accumulating nodes that have not yet been expanded or solved
3. Pick any of these nodes and expand it and if it has no successors call this value *FUTILITY* otherwise calculate only *f*' for each of the successors.
4. If *f*' is 0 then mark the node as *SOLVED*
5. Change the value of *f*' for the newly created node to reflect its successors by back propagation.
6. Wherever possible use the most promising routes and if a node is marked as *SOLVED* then mark the parent node as *SOLVED*.
7. If starting node is *SOLVED* or value greater than *FUTILITY*, stop, else repeat from 2.

**AO\*  Search Procedure.**

**1. Place the start node on open.**

**2. Using the search tree, compute the most promising solution tree TP .**

**3. Select node n that is both on open and a part of tp, remove n from open and place it no closed.**

**4. If n is a goal node, label n as solved. If the start node is solved, exit with success where tp is the solution tree, remove all nodes from open with a solved ancestor.**

**5. If n is not solvable node, label n as unsolvable. If the start node is labeled as unsolvable, exit with failure. Remove all nodes from open ,with unsolvable ancestors.**

**6. Otherwise, expand node n generating all of its successor compute the cost of for each newly generated node and place all such nodes on open.**

**7. Go back to step(2)**

**Note: AO\* will always find minimum cost solution.**