Artificial Intelligence

What is AI.?

Explaination & Base to Intelligent Agents.

Data
↓
Information
↓
Knowledge
↓
Intelligence

AI → Branch of Computer Science Concerned with the study & Creation of Computer Systems that behave some form of intelligence.

* LEARN
* Draw Conclusions (Reason)
* Natural Language Understanding. (NLP) To Enable it to Communicate Successfully in English
* That Preceive Visual science
* HUMAN INTELLIGENCE.

// Prolog and Lisp Programming Languages are Used.

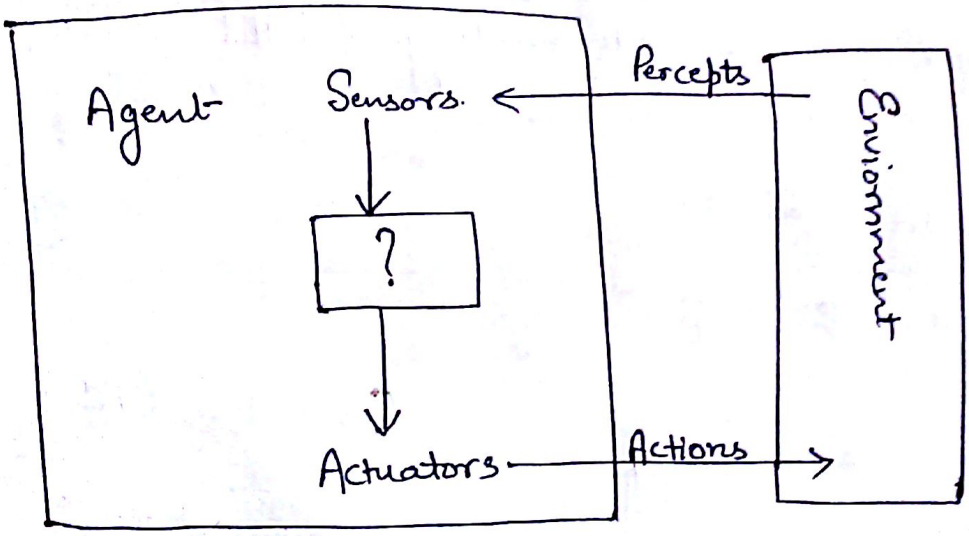
Human brain	Vs	Computer (AI)
1. Hearing device		1. Non-hearing device.
2. Self willing and Creating		2. Dependent and must be Programmed.
3. Has Continuous nature.		3. Discrete in nature.
4. Limited memory size.		4. Unlimited memory.
5. Basic unit is neurons		5. Basic unit is RAM cell.
6. Electrochemical storage device		6. electromechanical, electronic and magnetic.
7. Neuron speed 50 to 100 m/sec		7. Speed of light = Speed of neurons

Applications of AI

⇒ Replicate human intelligence.
⇒ Solve knowledge - intensive tasks.
⇒ Intelligent Connection of Perception and action.
⇒ Enhance human-human, human - Computer and Computer-Computer interaction.

Intelligent Agents

We adopt the View that intelligence is Concerned mainly with rational action, in a situation
Ideally an intelligent agent takes the best possible action in a situation.



* Agents interact with enviornments through sensors and actuators

An agent is anything that can be viewed as Perceiving its enviornment through Sensors and acting upon that enviornment through actuators

Percepts is the term refer to the agent's Perceptual inputs at any given instant

A rational Agent is one that does the right thing

PEAS

Example for agent

Agent Type	Performance Measure	Enviornment	Actuators	Sensors
Taxi driver	Safe, fast legal, Smooth drive / Comfort trip	Roads, other traffic, Pedestrians Customers	Steering, acc., break, signal horn, display.	Cameras, Sonar Speedometer, GPS, Odometer, accelerometer Engine Sensors, Keyboard

Performance Measure :— To inspire automated driver.

Enviornment :— That the taxi will face.

Actuators :— Actuators for those available to a human driver.

Sensors :— Includes one or more Controllable Cameras so that it Can see the road

Type of Agents

① → Simple reflex agents (Select actions on the basis of Current Perception) Ignoring the rest of the Percept history.

PERCEPTION ⟶ The ability to See hear or become aware of something through the senses

② Model Based Reflex agents — Keeps track of the Current state of the World, using an internal model. it then Chooses an action in the same way as reflex agent

③ Goal based Agents:— The agent needs some sort of goal info. that describes situations that are desireable.

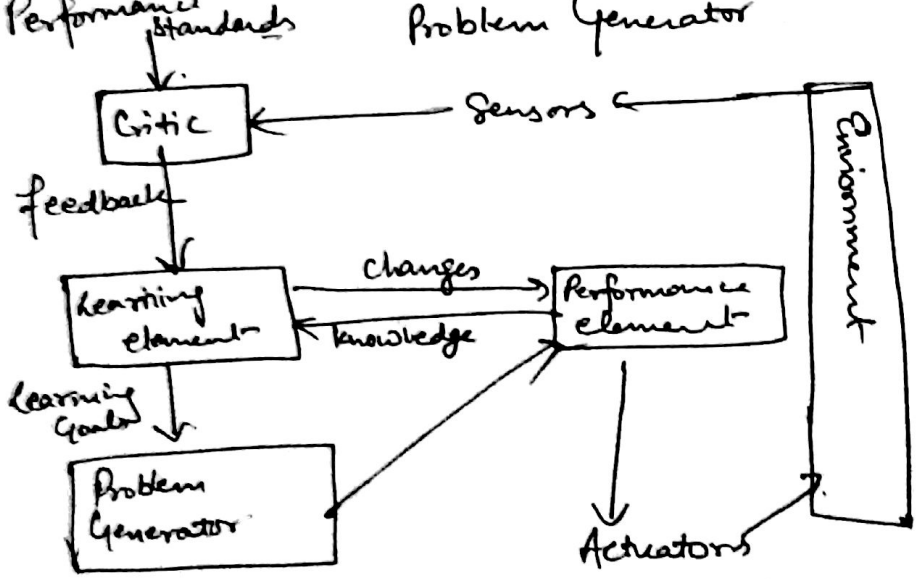
Search and planning fields are used to achieve the Goal.

(Behavior)

④ Utility based Agents:— Goals just Provide a Crude binary distinction b/w "happy" and "unhappy" states so Computer Scientists use the term Utility instead.

⑤ Learning Agents:— (Consists)

Learning element
Performance element
Critic element
Problem Generator



Problem Solving

Problem formulation :- It is the process of deciding what actions and states to consider, given a goal

Task domains of AI

Mundane Tasks

* Perception
  - Vision
  - Speech
* Natural language
  - understanding
  - Generation
  - Translation
* Commonsense Reasoning
* Robot Control

Formal Tasks

* Games
  - chess
  - Backgammon
  - checkers
  - Go
* Mathematics
  - Geometry
  - Logic
  - Integral Calculus
  - Proving Properties of Programs

Expert Tasks

* Engineering
  - design
  - fault finding
  - Manufacturing planning
* Scientific Analysis
* Medical diagnosis
* Financial Analysis

Problem formulation after Goal formation

Problem formulation means choosing a relevant set of states, operators for moving from one state to another, the goal test function and the path cost function.

The relevant set of states should include the current state, which is ~~finite~~ initial state and at least one goal state.

The operator correspond to imaginary actions and that the agent might take.

Goal test function is a function which determines if a single state is a goal state.

Path Cost is the sum of the cost of individual actions along a path from one state to another

- → Single State Problem
- → Multiple State Problem
- → Contingency problem
- → Exploration Problem.

Single State Problem :-

Example Automated Vacuum cleaner

Example 2 rooms
condition → dirt can be in zero, one or two rooms

Goal formulation → we want all the dirt cleaned up
formally the goal is {State 7, State 8}
{} operation indicates a set

Problem formulation Set of all possible states is ("move left", "move right", and "Vacuum")

Multiple State Problems

Suppose that robot has no sensors that can tell it doesn't know where it is initially.

Note that regardless of what the initial state is, the sequence of actions [right, vacuum, left, vacuum] endsup in a goal state

Contingency Problems:-

Suppose vacuum action sometimes actually deposit dirt on the carpet but only if the carpet is already clean Now right, vacuum, left, vacuum... doesn't work either There doesn't exist any FIXED plan that always works An agent for this environment must have a sensor and it must combine decision-making, sensing and execution this is called interleaving.

Exploration Problems :-

Suppose that Robot is completely ignorant. Then it must take actions for the purpose of acquiring knowledge about their effects, Not just their contribution towards achieving a goal.

This is called Exploration that the agent must do learning about the environment.

Uninformed Search Strategies

A Problem determines the graph and the goal but not which path to select from the frontier. This is the job of a search Strategy. A search Strategy specifies which paths are selected from the frontier.

DFS -> Depth First Search
BFS -> Breadth First Search
Lowest-Cost first Search.

-> in DFS frontier acts as a LIFO Stack
-> The elements are added to the stack one at a time

In BFS the frontier is implemented as FIFO. Thus the Path that is selected from the frontier is the one that was added earliest

(A.I)

Uninformed Search Strategies, heuristics, Informed Search Strategies

Constraint Satisfaction

Iterative deepening

Problem by Searching

State space formulation

dfs / BFS

Uninformed Search

A Problem determines the graph and the goal but not which path to select from the frontier.

* This is the job of a search strategy.

A search Strategy specifies which paths are selected from the frontier. ! Different Strategies are obtained by modifying how the selection of paths in the frontier is implemented.

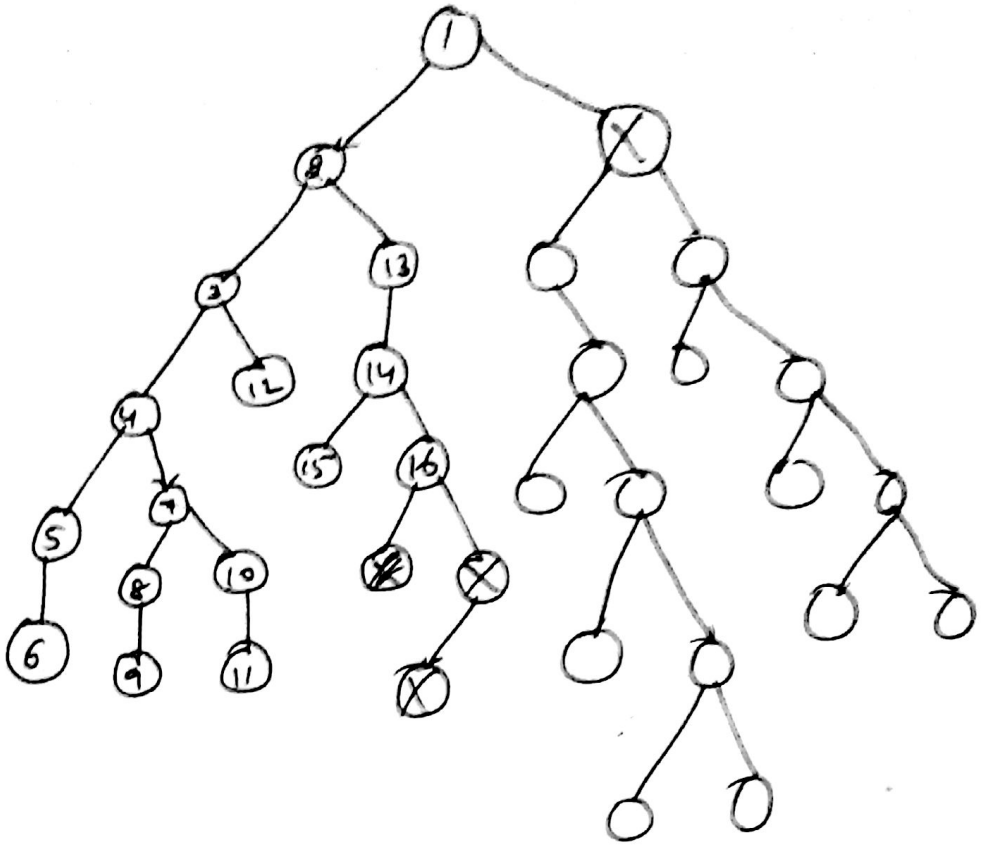
Three uninformed search Strategies that donot take into account the location of the goal, these algos ignore where they are going until they find a goal and report Success

↳ DFS (Acts like LIFO stack)

↳ BFS

↳ Lowest Cost first Search.

DFS Example Next Page



Implementing the frontier as a stack results in paths being ~~Pursuded~~ Pursued in a depth first manner. Searching one path to its Completion before trying any alternative path.

This method is said to involve backtracking. The algorithm selects a first alternative at each node then it backtracks to the next alternative when it has Pursued all the paths from the first Selection. Some Paths may be infinite. When the graph has Cycles or infinitely many nodes, in which case a depth first search may never Stop.

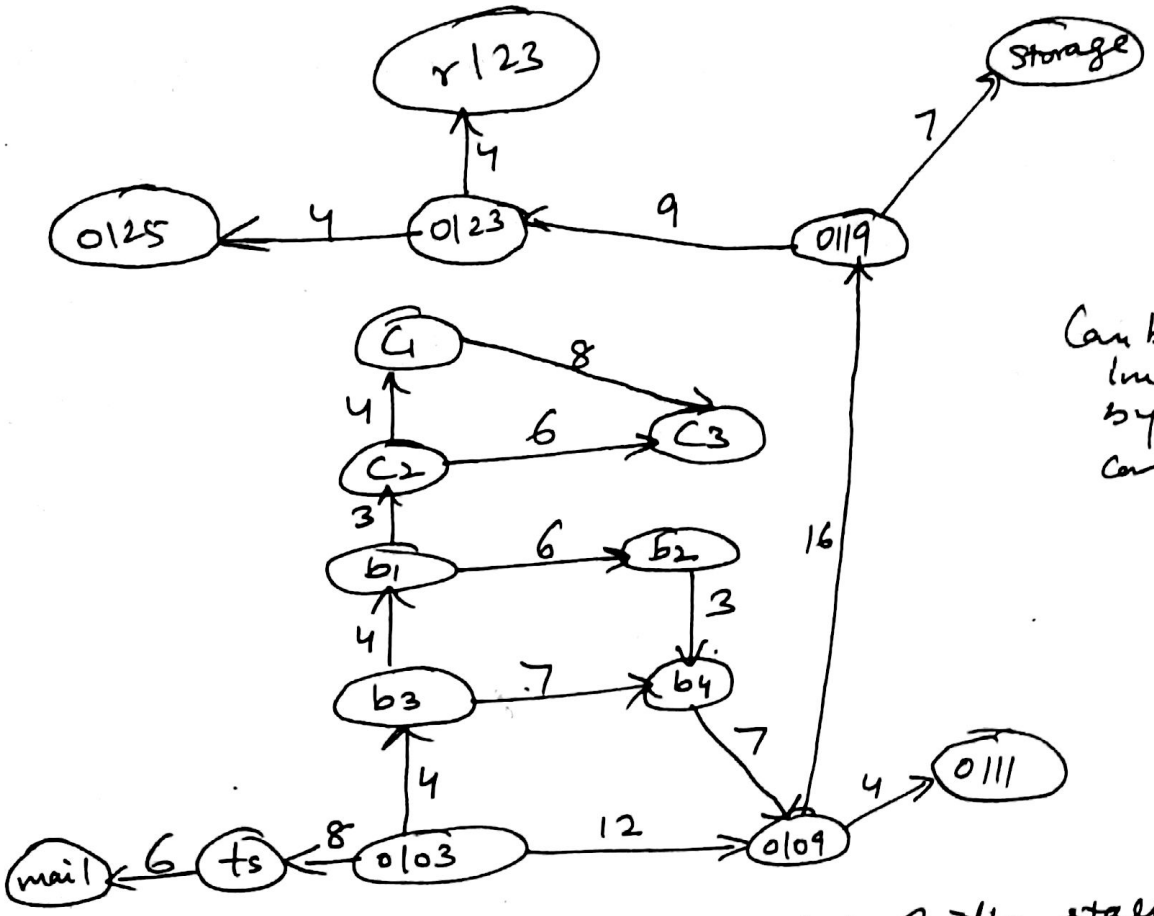
Example → Start node is 0103 and goal node is r123

Initially frontier Contains [0103]

next Stage

(1) [<0103, ts>, <0103, bs>, <0103, 0109>]

given



Can be improved by not considering cycles

→ ts is selected as it is at the top of the stack

So next step is

<0103, ts, mail>, <0103, b3>, <0103, 0109>].

hence next Path is mail

<0103, ts, mail>

now mail is empty set as it has no neighbours

So resulting frontier is

[<0103, b3>, <0103, 0109>]

now b3 at top

[<0103, b3, b1>, <0103, b3, b4>, <0103, 0109>].

now b1 is selected as frontier

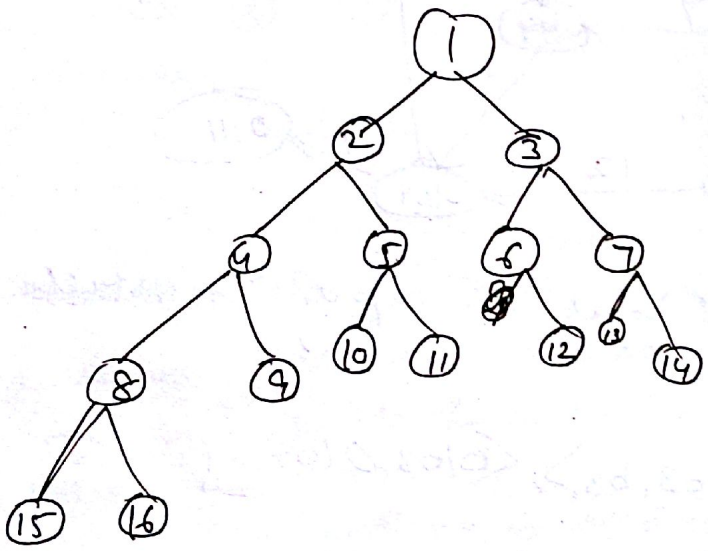
[<0103, b3, b1, C2> <0103, b3, b1, b2>, <0103, b3, b4>
<0103 0109>]

[<o|o3, b3, b1, c2, c3>, <o|o3, b3, b1, c2, c1>,
<o|o3, b3, b1, b2>, <o|o3, b3, b4>, <o|o3, o|o9>]

Iterative deepening

one way to Combine the space efficiency of DFS with the optimality of bfs methods to use iterative deepening

BFS



BFS is useful when

→ Space is not a Problem
→ you want to find the Sol. Containing fewest arcs.
→ few Solutions may exist and at least one ~~Path~~ has a short Path length and
→ Infinite paths may exist, because it explores all of the search space even with finite paths.

When more information than the initial state, the operators, and the goal test is available, the size of the search can usually be Constrained. When this is the Case, the better the info. available the more efficient the search process will be.

They often depend on the use of heuristic information.

Heuristic Info

Information about the problem ....

(the nature of the states, the cost of transforming from one state to another, the promise of taking a certain path, and the characteristics of the goal(s))

.... Can sometimes be used to Guide the search more efficiently. This information can be expressed in the form of a heuristic evaluation function $f(n,g)$ where n be the nodes and g be the goals.

P- Heuristic is a rule of thumb or judgmental technique that leads to a solution some of the time but provides no gurantee of success. it may infact end in failure.

Heuristics play an important role in search strategies because of the exponential nature of most Problems.

They help to reduce the number of alternatives from an exponential number to a polynomial number any thereby obtain a solution in a tolerable amount of time

Any policy which uses as little search effort as possible to find any qualified Goal has been called a satisfying policy.

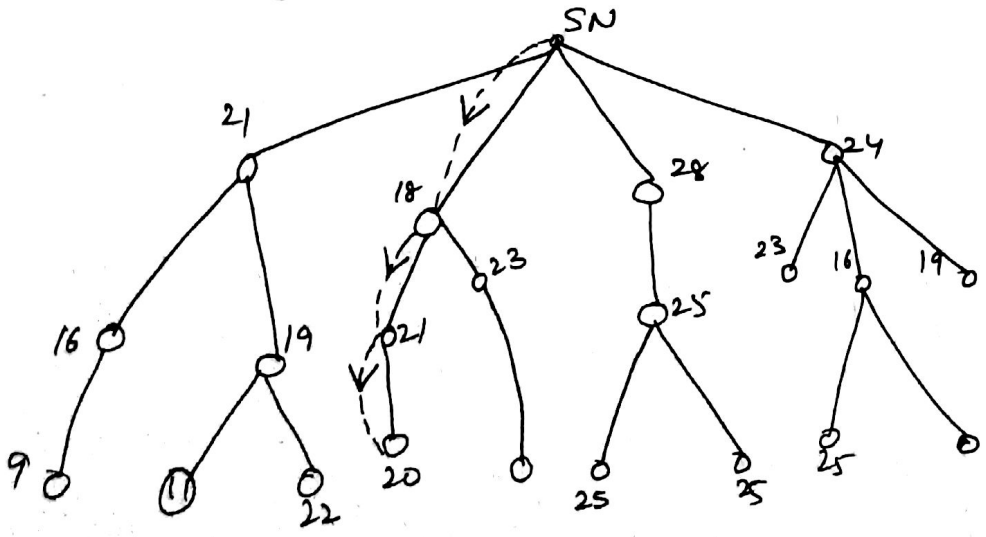
Example

A simple heuristic for choosing the next city at any point in a tour is one which picks the nearest unvisited neighbour.

In General problem solver. this is accomplished by first planning a solution by breaking the main problem down into several subproblems of lesser Complexity.

Hill Climbing Method

Hill climbing is like DFS where most Promising child is selected for expansion. When the children have been generated, alternative choices are evaluated using some type of heuristic function.



The path that appears most promising is then chosen and no further reference to the parent or children is retained. The Process continues from node to node with promising expanded nodes being discarded.

(anomalies are)

(foothill, ridge, plateau and traps)

these anomalies may be avoided using Best first search.

Heuristic Search Techniques

Informed

① Generate and test

Simplest of all approaches.

Algorithm

1. Generate a possible Solution. For some problems, this means generating a particular point in the problem space. or in other words. Generate a path from a start State.
2. Test to see if this is actually a solution by Comparing the chosen point or the Endpoint of the choosen path to the set of acceptable goal states.
3. If a solution has been found, Quit. otherwise, return to step 1.

If the generation of possible solution is done Systematically then this procedure will find a solution eventually. if one exists

Generate and test algo is a DFS Procedure (with Backtracking) since Complete Solution must be Generated before they Can be tested

* If some intermediate states are likely to appear often in the tree, however it may be better to modify that procedure, to traverse a graph rather than a tree.

Hill Climbing

It is a Variant of Generate and test in which feedback from the test Procedure is used to help the generator decide which direction to move in the search space. In the pure generate and test Procedure the test function responds only with a yes or no. But if the test function is augmented with a heuristic function that Provides an estimate of how close a given state is to a goal state.

Algo (Simple hill Climbing)

1) Evaluate the initial state. If it is a goal state, then return it and quit. otherwise Continue with the initial state as Current State.

2) Loop until a solution is found or until there are no new 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;

The key difference between this algorithm and the one we have for generate and test is, the use of an evaluation function as a way to inject task specific knowledge into the control process.

Steepest - Ascent hill Climbing (gradient Search)

A useful variation on simple hill Climbing considers all the moves from the current state and selects the best one as the next state.

Algo

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 changes to the 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 ~~that applies to the current state do.~~ 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 it is not better, leave successor alone.

(c) If the successor is better than Current state. then Set Current state to successor.

Steepest-Ascent hill climbing may fail to find a solution. because:

A local Maximum → A state that is better than all it's neighbours but is not better than some states farther away..

At local maximum. all moves appear to make things worse..

local maximum are frustrating because they often occur almost within sight of a solution.

In this Case they are Called "foothills"

A plateau is a flat area of the search space in which a whole set of neighbouring states have the same value.

on a plateau it is not possible to determine the best direction.

A ridge is a special kind of local maximum. It is an area of the search space that is higher than surrounding areas and that itself has a slope.

Problems Can be resolved By

→ Backtrack

→ Big jump (plateau)

→ Apply two or more rules.

Best first Search

(DFS + BFS) => Best first Search

Best first Search

DFS is good because it allows a solution to be found without all Competing branches having to be expanded

BFS is good because it doesnot get trapped on dead Ends.

{one way of Combining the two is to follow a single path at a time, but switch paths whenever some Competing paths looks more promising than the Current one does.}

At each step of the best first search procers, we select the most promising of the nodes we have expanded/generated so far. This is done by applying an appropriate heuristic function to each of them.

Examples:

OR Graphs

Each node will Contain in addition to a description of the problem state it represents, an indication of how promising it is, A parent link that points back to the best node from which it Came and a list of the nodes that were generated from it. The parent link will make it possible to recover the path to the goal node once the goal is found.

The list of successors ~~or its successors~~ will make it possible, if a better path is found to an already Existing node' To Propagate the improvement down to it's successors,

We Call a graph of this sort an OR graph, Since each of it's Branches represents an alternative problem solving path.

To Implement such a graph search procedure, we need to use two list of nodes:—

OPEN :- Nodes that have been generated and have had the heuristic function applied to them but which have not yet been examined.

≡ highest priority (most promising)

Closed :-

nodes that have already been examined. We need to keep these nodes in the memory, if we want to search a graph rather than a tree.

Algo

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 them best node on OPEN
   (b) Generate its successor.
   (c) for each successor do :-
      i) If it has not been generated before, evaluate it and 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 A* Algorithm

Best first Search was the simplified form of A*.

A* Algorithm

1. Start with open Containing ~~only~~ only the initial node
   Set the node's g value to 0
   Cost I → Current

(1) its h' value to whatever it is and its f' value to h'+0 or node to sol
f' → heuristic func value
h' Set CLOSED to the empty list.

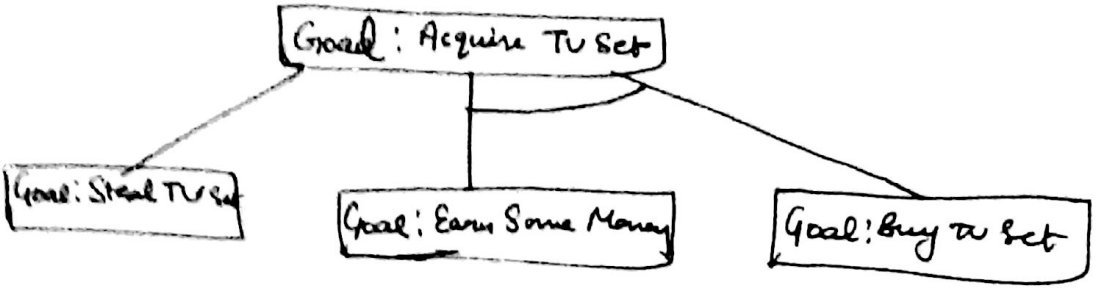
(2) Until a goal node is found, repeat the following procedure.
→ If there are no nodes on OPEN, report failure. Otherwise pick the node on OPEN with the lowest f' value. Call it BESTNODE Remove it from OPEN. place it on CLOSED.
See if BESTNODE is a goal node. If So exit and report a solution
otherwise generate the successors of BESTNODE but donot set BESTNODE to point to them yet.

For each Successor do!—

(a) Set Successor to point to BESTNODE
(b) Compute g (SUCCESSOR) = g(BESTNODE) + the cost of getting from BESTNODE to successor
(c) See if successor is the same as any node on OPEN.
Call that node OLD
Compare OLD ⇔ Current BESTNODE.
new cheaper path in g(OLD) and update f'(OLD)
(d) If the successor was not on Open. see if it is on closed.
If so Call the node on closed. old.
(Check and Compare).
(e) If successor was not already on Open or closed then put it on Open and add it to the list of BESTNODEs successors.

Applied to Graphs

Problem Reduction
AND / OR Graphs



Algo for problem Reduction

1. Initialize the graph to the starting node.
2. Loop until the starting node is labelled solved or until its cost goes above FUTILITY.
   (a) Start from initial node on the current best path for the path not yet been Extended or Solved.
   (b) Pick any unexpanded node & expand it. If there are no successors assign FUTILITY as the value of the nodes
   (c) If any nodes contain a successor are whose decedents are all Solved labelled the node itself as solved.

Constraint Satisfaction

Discover some problem that satisfies a given set of Constraints.

* It is a two step process. first Constraints are discovered and propagated as far as 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 and so forth.

Algo

1. Propagate available Constraints. To do this first set OPEN to the set of all objects that must have values assigned to them in a ~~so~~ Complete solution
Then do until Inconsistency is detected or until OPEN is empty:-

(a) Set the object OB from OPEN - Strengthen as much as possible the set of Constraints that apply to OB.

(b) If this set is different from the set that was designed last time OB was examined if this is the first time OB has been examined, then add to OPEN all objects that share any Constraint with OB

(c) Remove OB from ~~OB~~ OPEN.

(2) If the union of the Constraints discovered above defines a solution. defines a sol., then quit and report the sol.

(3) If it is a Contradiction then report failure

(4) If neither occurs then make a guess to proceed

(a) Select an object that has not yet determined & proceed

(b) Recursively invoke Constraint Satisfaction with the Current set of ~~states~~ Constraints augmented by the strengthening Constraint just selected.

* Constraints are propagated by using rules that Corresponds to the list of ~~ops~~ arithmetic.
* A value is guessed for some letter whose value is not yet determined.

MEANS End's Analysis

Means end analysis process centers around the detection of differences b/w the current state and the goal state

The kind of Backward chaining in which operators are selected and then subgoals are setup to establish the preconditions of the operators is called operator Subgoaling

A separate data structure called a difference table indexes the rules by the differences that they can be used to reduce

Algo

1. Compare Current to goal (no differences then return)
2. Select difference and reduce it by doing the following until success or failure.
3. (a) Select untried operator O
   (b) Attempt to apply O to Current.

   (First part ← MEA (CURRENT, O-START))
   and
   (LAST-PART ← (MEMO-RESULT, GOAL))

   FIRST-PART, O and LAST-PART.