ARTIFICIAL INTELLIGENCE - DECEMBER 2015
(SEMESTER 7)

TOTAL MARKS: 80
TOTAL TIME: 3 HOURS

(1) Question 1 is compulsory.
(2) Attempt any ***three*** *from the remaining questions.*
(3) Assume data if required.
(4) Figures to the right indicate full marks.

Attempt any four (4) questions from the following.

1 (a) Define heuristic function. Give an example heuristics function for Blocks World Problem. (5 marks)

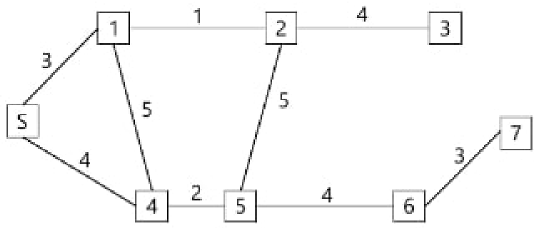
1 (b) Find the the heuristics values for a particular state of the Blocks World Problem. (5 marks)

1 (c) Define Rationality and Rational Agent. Give an example of rational action performed by any intelligent agent. (5 marks)

1 (d) Compare and Contrast problem solving agent and planning agent. (5 marks)

1 (e) Represent the following statement into FOPL. (5 marks)
(i) Anyone who kills an animal is loved by no on.
(ii) A square is breezy if and only if there is pit in a neighbouring square (Assume the wumpus world environment).
(iii) Give the PEAS description for an Internet shopping agent. Characterize its environment.

2 (a) Consider the graph given in Figure 1 below. Assume that the-initial state is S and the goal state is 7. Find a path from the initial state to the goal state using A* Search. Also report the solution cost. The straight line distance heuristic estimates for the nodes are as follows: b(1)=14, h(2)=10, h(3)=8, h(4)=12, h(5)=10, h(6)=10, h(S)=15. (10 marks)



2 (b) Draw and describe the architecture of expert system. (6 marks)

2 (c) Convert the following propositional logic statement into CNF. (4 marks)

---

3 (a) Consider the following axioms: (12 marks)
All people who are graduating are happy.
All happy people smile.
Someone is graduating.
i) Represent these axioms in first order predicate logic.
ii) Convert each formula to clause form
iii) Prove that 'Is someone smiling?' using resolution technique. Draw the resolution tree.

3 (b) What are the basic building blocks of Learning Agent? Explain each of them with a neat block diagram. (8 marks)

4 (a) Construct a decision tree for the following set of samples. Write any two decision rules obtained from the tree. Classify a new sample with (gender = 'Female', height='1.92m') (10 marks)

Person ID	Gender	Height	Class
1	Female	1.6m	Short
2	Male	2m	Tall
3	Female	1.9m	Medium
4	Female	2.1m	Tall
5	Female	1.7m	Short
6	Male	1.85m	Medium
7	Female	1.6m	Short
8	Male	1.7m	Short
9	Male	2.2m	Tall

4 (b) What are the problems/frustrations that occur in hill climbing technique? Illustrate with an example. (6 marks)

4 (c) Draw a game tree for a Tic-Tac Toe problem. (4 marks)

---

5 (a) Write a short note on genetic algorithm. (8 marks)

5 (b) It is known whether or not a person has cancer is directly influenced by whether she is exposed to second-hand smoke and whether she smokes. Both of these things are affected by whether her parents smoke. Cancer reduces a person's life expectancy. (6 marks)
i) Draw the Bayesian Belief Network for the above situation
ii) Associate a conditional probability table for each node.

5 (c) Explain a partial order planner with an example. (6 marks)

---

6 (a) Write a PROLOG program to find Fibonacci series. (10 marks)

6 (b) What are the levels of knowledge used in language understanding? Also write down the techniques used in NLP. (10 marks)

1 (a) Define heuristic function. Give an example heuristics function for Blocks World Problem. ***--- 5 Marks***

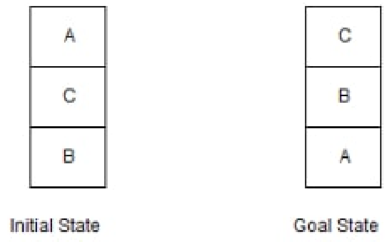
ADD NOTE BOOKMARK

Answer:

Heuristic Function:

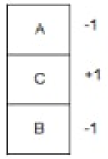
- Heuristic function is a concept that associates a value with every state in a given problem space and thus helps to compare any two states to achieve a goal oriented search.
- A value is associated with every state. The associated value is computed by applying the function on that state just to find the distance of that state from the goal state.
- Such a function is called as heuristic function and the value calculated by the function is called as the Heuristic Value.

Example:



To compute the heuristic value of a state we give point to each block in the state. The point to a block can be +1 or -1. It is +1 if the block has a correct block immediately below it compared to the goal state. It is -1 if the block has an incorrect block immediately below it compared to the goal. Finally the points of all blocks in the state are added to give the heuristic value of the particular state.

For instance let us compute the heuristic value for the initial state.



Initial State

For block B: In the initial state nothing is below block B but int he goal state A is block B hence point for block B is -1

For block C: In the initial state B is below block C and in the goal state also B is below block C and hence point for block C is +1

For block A: In the initial state C is below block A but in the goal state nothing is below block A and hence point for block A is -1

Now final heuristic value for the initial state can be computed as addition of all the points

Hence, H(initial state)=(-1)+(+1)+(-1)=-1

and so on heuristic value for each state can be computed.

For goal state since all the blocks are in place each block will have a point of +1 and hence its heuristic value is +3

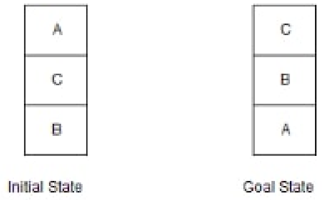
1 (b) Find the the heuristics values for a particular state of the Blocks World Problem. ***--- 5 Marks***

ADD NOTE BOOKMARK

This question was repeated in May 16.

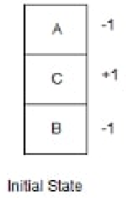
Answer:

Finding heuristic values for a Initial State of the Blocks World Problem:



To compute the heuristic value of a state we give point to each block in the state. The point to a block can be +1 or -1. It is +1 if the block has a correct block immediately below it compared to the goal state. It is -1 if the block has an incorrect block immediately below it compared to the goal. Finally the points of all blocks in the state are added to give the heuristic value of the particular state.

For instance let us compute the heuristic value for the initial state.



For block B: In the initial state nothing is below block B but int he goal state A is block B hence point for block B is -1

For block C: In the initial state B is below block C and in the goal state also B is below block C and hence point for block C is +1

For block A: In the initial state C is below block A but in the goal state nothing is below block A and hence point for block A is -1

Now final heuristic value for the initial state can be computed as addition of all the points

Hence, H(initial state)=(-1)+(+1)+(-1)=-1

and so on heuristic value for each state can be computed.

For goal state since all the blocks are in place each block will have a point of +1 and hence its heuristic value is +3

1 (c) Define Rationality and Rational Agent. Give an example of rational action performed by any intelligent agent. ***--- 5 Marks***

ADD NOTE BOOKMARK

Answer:

Rational agent is the one which has the ability to perceive from the environment and act on it so that the performance measure can be maximized.

Rationality is nothing but status of being reasonable, sensible, and having good sense of judgment.

Automated taxi driver is an example of rational action in which there are camera, GPS and speedometer which acts as sensor and senses all the activities while steering, brakes and accelator acts as effector which acts on certain action that the sensor perceives.

1 (d) Compare and Contrast problem solving agent and planning agent. ***--- 5 Marks***

ADD NOTE BOOKMARK

Answer::

No	Problem solving agent	Planning agent
1.	Problem solving agent just demonstrate one specific solution.	Planning is viewed as the producer or generator of the solution.
2.	Problem solving solves the complete problem and yields the solution.	It yields a plan, that when executed will achieve the goal.
3.	Problem solving is typically more concerned with plan execution.	planning is involved with plan generation.
4.	Every problem may not be docomposable into subproblems. Therefore cannot take the advantage of problem decomposition.	In planning one can treat the problem ad nearly docomposable and provide plan for subgoals that will collectively achieve the goal.
5.	Problem solving comprises standard search problems.	Planning is typically viewed as a generic term of problem solving because it deals with search on an abstract level.

1 (e) Represent the following statement into FOPL.
(i) Anyone who kills an animal is loved by no on.
(ii) A square is breezy if and only if there is pit in a neighbouring square (Assume the wumpus world environment). ***--- 5 Marks***
(iii) Give the PEAS description for an Internet shopping agent. Characterize its environment.

ADD NOTE BOOKMARK

Answer:

(i) $\forall x\ [\exists y\ (\text{Animal}(y) \land \text{Kills}(x,y))] \Rightarrow [(\forall z)\ \neg\text{Loves}(z,x)]$

(ii) $\forall s\ \text{Breezy}(s) \Leftrightarrow \exists\ r\ \ \text{Neighbour}(r,s) \wedge \text{Pit}(r)$

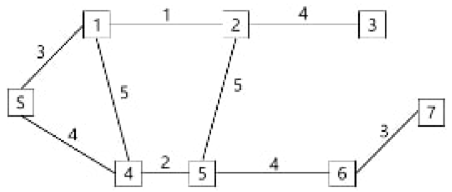
(iii) PEAS for an Internet Shopping Agent:

Performance	P: Quality, Price, Efficiency
Environment	E: Vendors, Shippers, websites, Internet
Actuators	A: Fill in forms, Follow URL, Display info
Sensors	S: Web pages (text, graphics, scripts), User requests

Environment Characteristics:

1. **Partially Observable**: Agent cannot excatly predict what the environment would be in advance.
2. **Deterministic (Semi)**: The next state is partially determined by the current state and the action executed.
3. **Sequential**: Current selections affect future actions.
4. **Static (Semi)**: Environment does not change much with the passage of time.
5. **Discrete**: There are fixed set of percepts and actions.
6. **Single agent**: As the agent operates by itself, it is single agent environment.

2 (a) Consider the graph given in Figure 1 below. Assume that the-initial state is S and the goal state is 7. Find a path from the initial state to the goal state using A* Search. Also report the solution cost. The straight line distance heuristic estimates for the nodes are as follows: b(1)=14, h(2)=10, h(3)=8, h(4)=12, h(5)=10, h(6)=10, h(S)=15.



--- 10 Marks

ADD NOTE

BOOKMARK

Answer:

A* search:

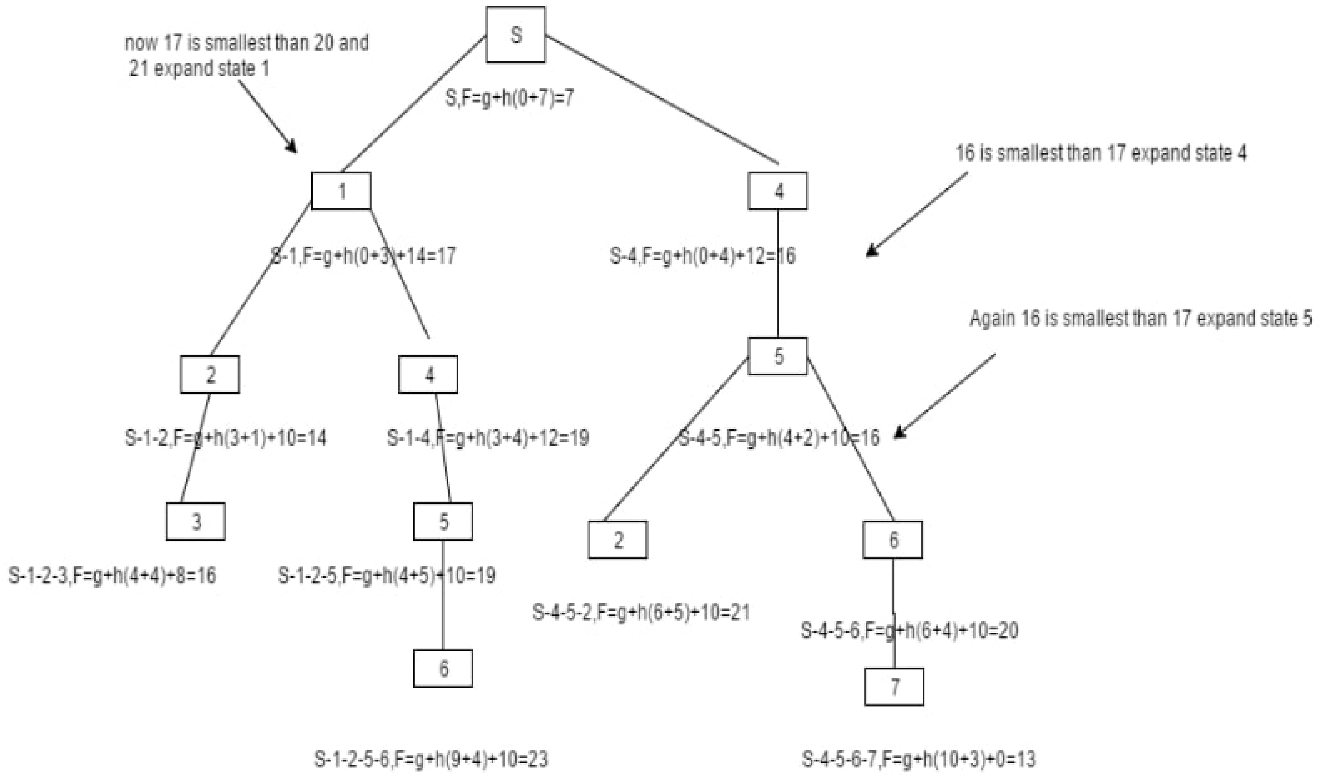
- A* algorithm is a best-first search algorithm in which the cost associated with a node is $f(n) = g(n) + h(n)$,
- where $g(n)$ is the cost of the path from the initial state to node n and $h(n)$ is the heuristic estimate or the cost or a path from node n to a goal.Thus, $f(n)$ estimates the lowest total cost of any solution path going through node n.
- At each point a node with lowest f value is chosen for expansion. Ties among nodes of equal f value should be broken in favor of nodes with lower h values. The algorithm terminates when a goal is chosen for expansion.
- A* algorithm guides an optimal path to a goal if the heuristic function $h(n)$ is admissible, meaning it never overestimates actual cost.

Here the following table shows the state and their heuristic value .

Here the following table shows the state and their heuristic value .

State	Heuristic value
S	15
1	14
2	10
3	8
4	12
5	10
6	10
7	0

- In the below solution we start with state 'S' whose F value is calculated by adding the cost associted with it which is 0 because it is starting state and heuristic value of S resultant value is 7.
- Now this node is expanded to state 1&4 .Value of F in state 1 is calculated as by adding the cost of the path from the initial state to node 1 and their heuristic value .Similarly value of F for state 4 is calculated .Now which state have lowest F value will be choosen for expansion .entire process is repeated as follows.



Optimal Path= S-4-5-6-7

2 (b) Draw and describe the architecture of expert system. ***--- 6 Marks***

ADD NOTE

BOOKMARK

Answer:

Expert System:

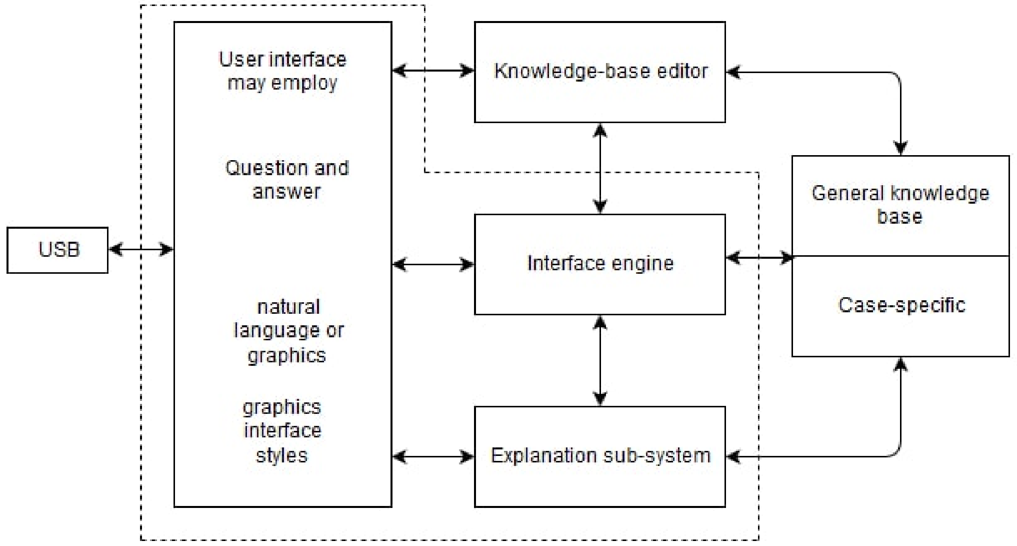


Fig: Architecture of typical expert system

- The above figure shows the modules that make up a typical expert system. The user interacts with the system through a user interface that simplified communication and hides much of the complexity such as the internal structure of the rule base.
- Expert system interfaces employ a variety of user styles, including question and answer menu driven or graphics interfaces. The final decision on the interface type is a compromise between user needs and the requirement of the knowledge base and inferencing system.
- The heart of the expert system is the knowledge base, which contains the knowledge of a particular application domain.
- In a rule-based expert system this knowledge is represented in the form of if..then. rules. The knowledge base containing both general knowledge as well as case specific information.
- The inference engine applies the knowledge to the solution of actual problems, It is essentially an interpreter for the knowledge base. In the production system, the inference engine perform the recognize-act control cycle.
- The procedures that implement the control cycle are separate from the production rules themselves. It is important to maintain this separation of the knowledge base and inference engine for several reasons:

1. This separation makes it possible to present to represent knowledge in a more natural fashion. If .. then rules, for example are closer to the way in which humans describe their problem-solving skills than is lower-level computer code.
2. Because, the knowledge base is separated from the program's lower-level control structures, expert system builders can focus on capturing and organizing problem solving knowledge rather than on the details of its computer implementation
3. Ideally the separation of knowledge and control allows changes to be made in one part of the knowledge base without creating side effects in others
4. The separation of the knowledge and control elements of the program allows the same control and interface software to be used in a variety of systems. The expert system shell has all the components of figure above except that the knowledge base and case specific data are empty and can be added for a new application. the broken lines of figure indicate the shell modules.

3 (a) Consider the following axioms:
All people who are graduating are happy.
All happy people smile.
Someone is graduating.
i) Represent these axioms in first order predicate logic.
ii) Convert each formula to clause form
iii) Prove that 'Is someone smiling?' using resolution technique.
Draw the resolution tree.

--- 12 Marks

ADD NOTE BOOKMARK

Answer:

1)First order logic for the statements:

a.All people who are graduating are happy.

$$\forall\ x\ graduating(x) \rightarrow happy(x)$$

b.All people who are graduating are happy.

$$\forall\ x\ happy(x) \rightarrow smiling(x)$$

c.Someone is graduating.

$$\forall\ x\ graduating(x)$$

d.Is someone is smiling

$\forall\ x\ smiling(x)$

Negate the above staement to prove for resolution:Â¬âˆƒ $x\ smiling(x)$

2)Conversion of each formula in to clause form are as sollows:

a.$\neg graduating(x) \rightarrow happy(x)$

b.$\neg happy(y) \rightarrow smiling(y)$

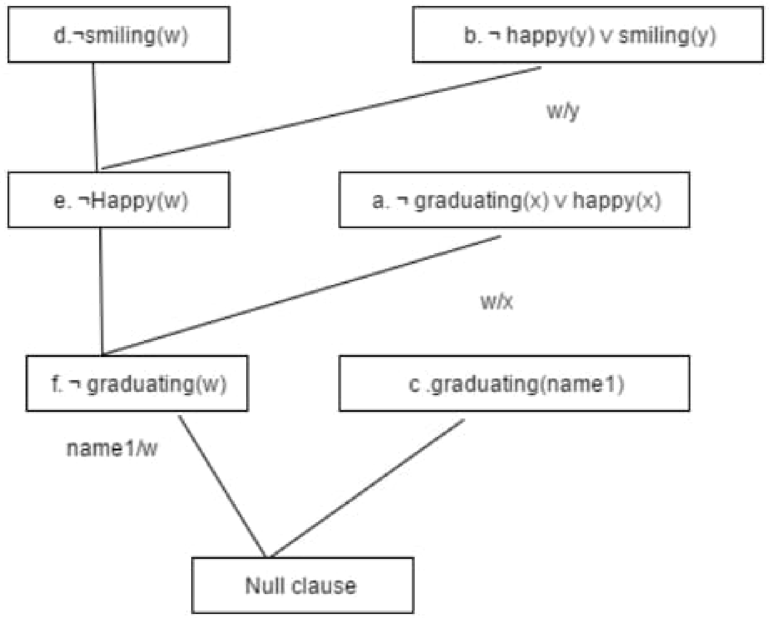
c. $graduating(Name1)$ <-name1 is the Skolemization constant

d.$\neg smiling(w)$

e.$\neg Happy$

f.$\neg graduating$

3) Proving for resolution:



3 (b) What are the basic building blocks of Learning Agent? Explain each of them with a neat block diagram. ***--- 8 Marks***

ADD NOTE BOOKMARK

Answer:

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

A learning agent takes advantage of "learning" allowing the agent to initially operate in unknown environments and to become more competent than its initial knowledge alone might allow.

A general learning agent has four basic components:

a. *The Performance Element* – which takes in percepts and decides on appropriate actions in the same way as a non-learning agent.
b. *The Critic* – which uses a fixed standard of performance to tell the learning element how well the agent is doing.
c. *The Learning Element* – that receives information from the critic and makes appropriate improvements to the performance element.
d. *The Problem Generator* – that suggests actions that will lead to new and informative experiences (e.g. as in carrying out experiments).

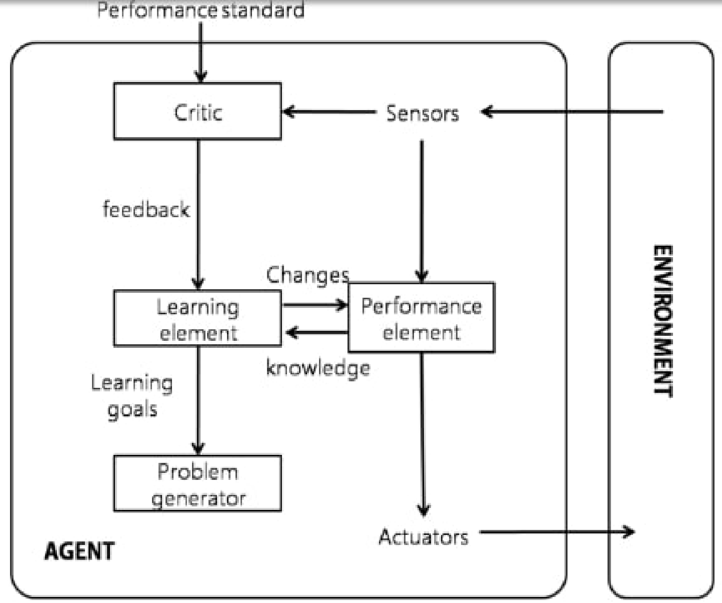


Figure: A general learning agent

The design of the learning element depends very much on the design of the performance element. The critic tells the learning element how well the agent is doing with respect to a fixed performance standard. The critic is necessary because the percepts themselves provide no indication of the agent's success.

Learning agent is also responsible for improving the efficiency of performance element for example when asked to make a trip to new destination, the taxi might take a while to consult its map and plan the best route. But next time, on a similar trip, the planning process should be much faster. This is called speed up learning request.

4 (b) What are the problems/frustrations that occur in hill climbing technique? Illustrate with an example. ***--- 6 Marks***

ADD NOTE BOOKMARK

Answer:

Hill Climbing:

- The Hill Climbing search always moves towards the goal.
- A Hill Climbing search always goes to the node closer to the goal.
- It is simply loop that continuously moves in the direction of increasing value.
- It is also called as Heuristic Search Technique.
- For example,Hill Climbing can be applied to traveling salesman problem.It is easy to find an initial solution that visits all the cities but will be very poor compared to optimal solution.The algorithm starts with such a solution and makes small improvements to it,such as switching the order in which two cities are visited. Eventually,a much shorter route is likely to be obtained.

Two types of Hill Climbing algorithms are:

1. Simple Hill Climbing Algorithm
2. Steepest Ascent Hill Climbing Algorithm

Simple Hill Climbing Algorithm: (Looks for first better successor)

Steps:

1. Evaluate initial state,if it is goal state then return the value and quit, else continue with initial state as current state.
2. Loop until a solution is found or there are no operators left to be applied in current state:
    - Select an operator that has not yet been applied to the cureent state and apply it to produce a new state.
    - Evaluate new state if:
        1. If is a goal state , return it and quit.
        2. If not and better state, make it current state.
        3. If it is not better than current state then continue in the loop.

Steepest Ascent Hill Climbing Algorithm:

Steps:

1. Evaluate initial state. If it is goal state then return and quit, otherwise continune initial state as current state.
2. Loop until a solution is found or until a complete iteration produces no change to current state :

- Let succ be a state such that any possible successor of current state will be better than succ.
- For each operator that applies to the current state do:

1. Apply the operator and generate new state.
2. Evaluate new state of it, if it is goal state then return and quit, else compare with succ. If it is better, set succ to current state.

- If successor is better than current state, then set current state to succ.

Problems in Hill Climbing:

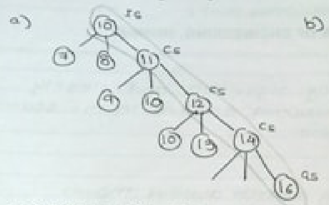
1. **Local Maxima**: Local Maxima is a state where all the successors of current state are worst than current state.
2. **Plateau:** Plateau is a state where all its successors are heuristically equivalent to current state.
3. **Ridge:** Ridge is a state where all the successors of current state are worst or equivalent to current state.

Solutions:

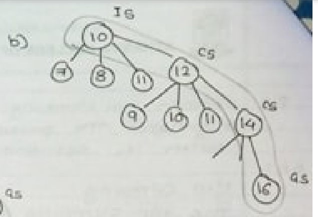
1. **Back Tracking**: Back tracks few steps and takes another alternative.
2. **Keep Moving:** For plateau or ridge, instead of back tracking, keep moving on straight path and there is a possibility that we may come acoss a shoulder which takes uo to the goal state.

Hill climbing Algorithm:

a)



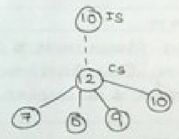
b)



- Simple Hill climbing (Looks for first better successor)
- Steepest ascent Hill climbing (Looks for best successor)

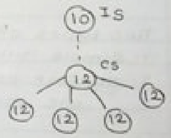
Problems in Hill Climbing:

1)



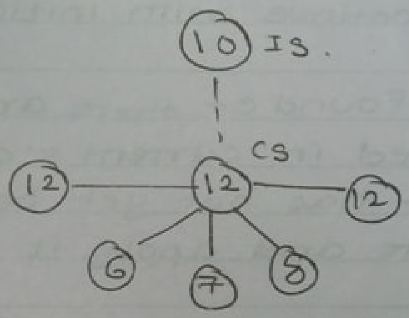
Local maxima

2)



Plateau

3)



Ridge.

5 (a) Write a short note on genetic algorithm. ***--- 8 Marks***

ADD NOTE BOOKMARK

Answer:

Genetic Algorithm:

- In the field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection.
- This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems.
- Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

1. Initialization

- The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

2. Selection

- During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.
- The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise.
- In some problems, it is hard or even impossible to define the fitness expression; in these cases, a simulation may be used to determine the fitness function value of a phenotype (e.g. computational fluid dynamics is used to determine the air resistance of a vehicle whose shape is encoded as the phenotype), or even interactive genetic algorithms are used.

3. Genetic operators

- The next step is to generate a second generation population of solutions from those selected through a combination of genetic operators: crossover (also called recombination), and mutation.

- For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests that more than two "parents" generate higher quality chromosomes.
- These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions. These less fit solutions ensure genetic diversity within the genetic pool of the parents and therefore ensure the genetic diversity of the subsequent generation of children.
- Opinion is divided over the importance of crossover versus mutation. There are many references in Fogel (2006) that support the importance of mutation-based search.
- Although crossover and mutation are known as the main genetic operators, it is possible to use other operators such as regrouping, colonization-extinction, or migration in genetic algorithms.
- It is worth tuning parameters such as the mutation probability, crossover probability and population size to find reasonable settings for the problem class being worked on. A very small mutation rate may lead to genetic drift (which is non-ergodic in nature). A recombination rate that is too high may lead to premature convergence of the genetic algorithm. A mutation rate that is too high may lead to loss of good solutions, unless elitist selection is employed.

Example:

Problem: "Find" the binary number 11010010.

Initialization:

We start with 5 random binary numbers with 8 digits each.
1.01001010
2. 10011011
3.01100001
4.10100110
5.01010011

The fitness function:

We define the fitness of a number to be the sum of the distances of its digits from these in the target, with the sign minus.
The target: 11010010

fitness(01001010)= -((|1-0|+|1-1|+|0-0|+|1-0|+|0-1|+|0-0|+|1-1|+|0-0|))=-3

1.fitness(01001010)=-3
2.fitness(10011011)=-3
3.fitness(01100001)=-5
4.fitness(10100110)=-4
5.fitness(01010011)=-2•

Parent Selection:

In each generation, some constant number of parents, say, 4 are chosen. Higher is the fitness, greater is the probability of choosing the individual.

{"01001010,10011011,01100001,10100110,01010011}"
{-3,-3, -5,-4,-2}
Assume we select "10011011", "10100110".
fitness("10011011") > fitness("10100110")è "10011011" selected.

We get {"01001010,10011011,10100110,01010011"}

Recombination:

Two selected parents will be coupled with probability 0.5.
("01001010,10011011") , ("01001010", "10100110"), ("10100110" ",01010011")
3 is selected as a random number from 1...8
Then we do a crossover:

From ("01001010,10011011") we get ("01011011,10001010")

We repeat that for each selected couple.

Mutation:

For each digit in each of the offsprings , we make the bit flip with probability 0.05.
For "10001010" we have "10011010".

5 (c) Explain a partial order planner with an example. --- 6 Marks

ADD NOTE BOOKMARK

This question was repeated in May 15.

Answer:

- Planning is "the task of coming up with a sequence of actions that will achieve a goal."
- There are several types of algorithms that allow us to construct a plan such as progression planning, regression planning and partial-order planning.
- Partial order planning is not totally ordered.
- Partial order planning enables us to take advantage of problem decomposition.
- The algorithm works on several subgoals independently, solves them with severals subplans, then combines the subplans.
- In addition, such an approach also has the advantage of flexibility.
- That is, the planner can work on 'obious' or 'important' decisions first, rather than being forced to work on steps in chronological order.
- Partially order plans are created by a search through the space of plans rather than through the state space.
- ***Example:*** the simple flat tire probelm

  Init(At(Spare,Truck) ^ At(Flat,Axle))

  Goal (At(Spare,Axle))

  Action (Remove(Spare,Trunk),

PRECOND: At(Spare,Truck)

EFFECT: At(Spare,Ground) ^ ¬ At(Spare,Trunk))

Action (Remove(Flat,Axle),

PRECOND: At(Flat,Axle)

EFFECT: At(Flat,Ground) ^ ¬ At(Flat,Axle))

Action (PutOn(Spare,Axle),

PRECOND: At(Spare,Ground) ^ ¬ At(Flat,Axle)

EFFECT: At(Spare,Axle) ^ ¬ At(Spare,Ground))

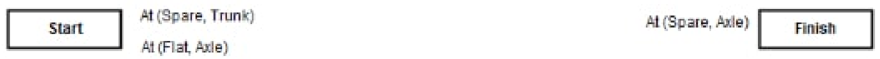
Action (LeaveOvernight,

PRECOND:

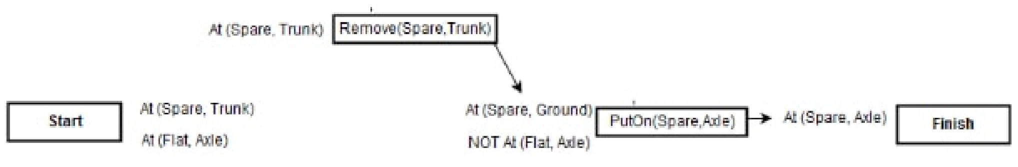
EFFECT: ¬ At(Spare,Ground) ^ ¬ At(Spare,Trunk) ^ ¬ At(Spare,Axle) ^ ¬ At(Flat,Axle) ^ ¬ At(Flat,Ground))

- We start with the empty plan consisting of just initial state and goal with no action in between, as in figure (a)
- The search procedure then looks for a flaw in the plan, and makes an addition to the plan to correct the flaw (or if no correction can be made the search backtracks and tries something else).

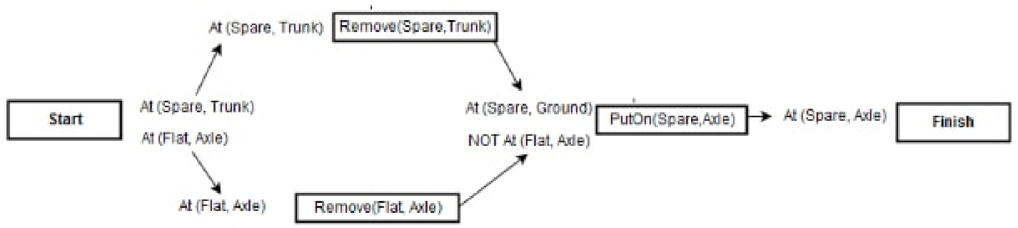
- For example one flaw in the empty plan is that no action achieves At(Spare,Axle). One way to correct the flaw is to insert into the plan the action PutOn(Spare,Axle).



(a)



(b)



(c)

(a) the tire problem expressed as an empty plan

(b) an incomplete partially ordered plan for the tire problem

(c) a complete partially-ordered solution

6 (a) Write a PROLOG program to find Fibonacci series. ***--- 10 Marks***

ADD NOTE BOOKMARK

Answer:

The Fibonacci numbers upto certain term can be represented as: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144..... or 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144....

First Term = 0
Second term = 1
Third Term = First + Second = 0+1 =1
Fourth term = Second + Third =1+1 = 2
Fifth Term = Third + Fourth = 2+1 = 3
Sixth Term= Fourth + Fifth = 3+2 = 5
Seventh Term = Fifth + Sixth = 3+5 = 8
Eighth Term = Sixth + Seventh = 5+8 = 13 ... and so on to infinity!

Algorithm:

1. Start
2. Declare variables i, a,b , show
3. Initialize the variables, a=0, b=1, and show =0
4. Enter the number of terms of Fibonacci series to be printed
5. Print First two terms of series
6. Use loop for the following steps

    -> show=a+b

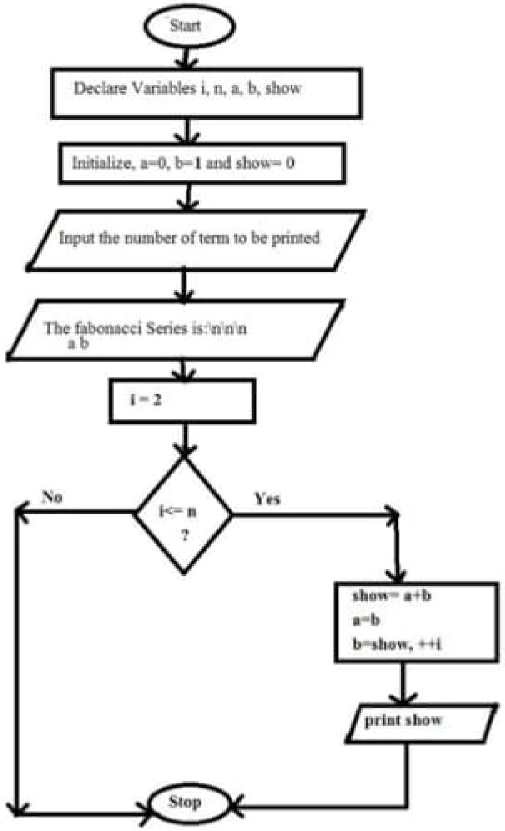
    -> a=b

    -> b=show

-> increase value of i each time by 1

-> print the value of show

7. End



Start
Declare Variables i, n, a, b, show
Initialize, a=0, b=1 and show= 0
Input the number of term to be printed
The fabonacci Series is:\n\n\n
a b
i = 2
No
i<= n
?
Yes
show= a+b
a=b
b=show, ++i
print show
Stop

PROLOG program for finding Fibonacci series.

```
domains
        x = integer
predicates
       fibonacci(x)
clauses
      fibonacci(1).
fibonacci(N) :-
     N1 = N - 1,
     N1 >= 0,!,
     fibonacci(N1),
     write(F1," ,"),
     F = F1 + N.
```

INPUT:

```
fibonacci(7)
```

OUTPUT:

```
1, 1, 2, 3, 5, 8, 13
```