Indian Institute of Information Technology Vadodara(IIITV)

CS362 Artificial Intelligence Laboratory Report

[1] Bhatt Krutarth(201951041)
[2] Bhimani Yash(201951042)
[3] Rakshilkumar Modi(201951124)
[4] Zala Indravijaysinh(201951182)

I. INTRODUCTION

In this lab report we have given an overview of the experiments, observation and results while performing 4 experiments. We have chosen experiment:

- Week1 :To design a graph search agent and understand the use of a hash table, queue in state space search.
- Week3 : Non-deterministic Search - Simulated Annealing
- Week5: Game Playing Agent
- Week6: Understanding Bayesian Network

We have performed the experiments on jupyter notebook for Python Code and RStudio for R. At the end of this report the GitHub repository link is also added.

II. WEEK 1

To design a graph search agent and understand the use of a hash table, queue in state space search.

A. Write a pseudocode for a graph search agent. Represent the agent in the form of a flow chart. Clearly mention all the implementation details with reasons

Answer:

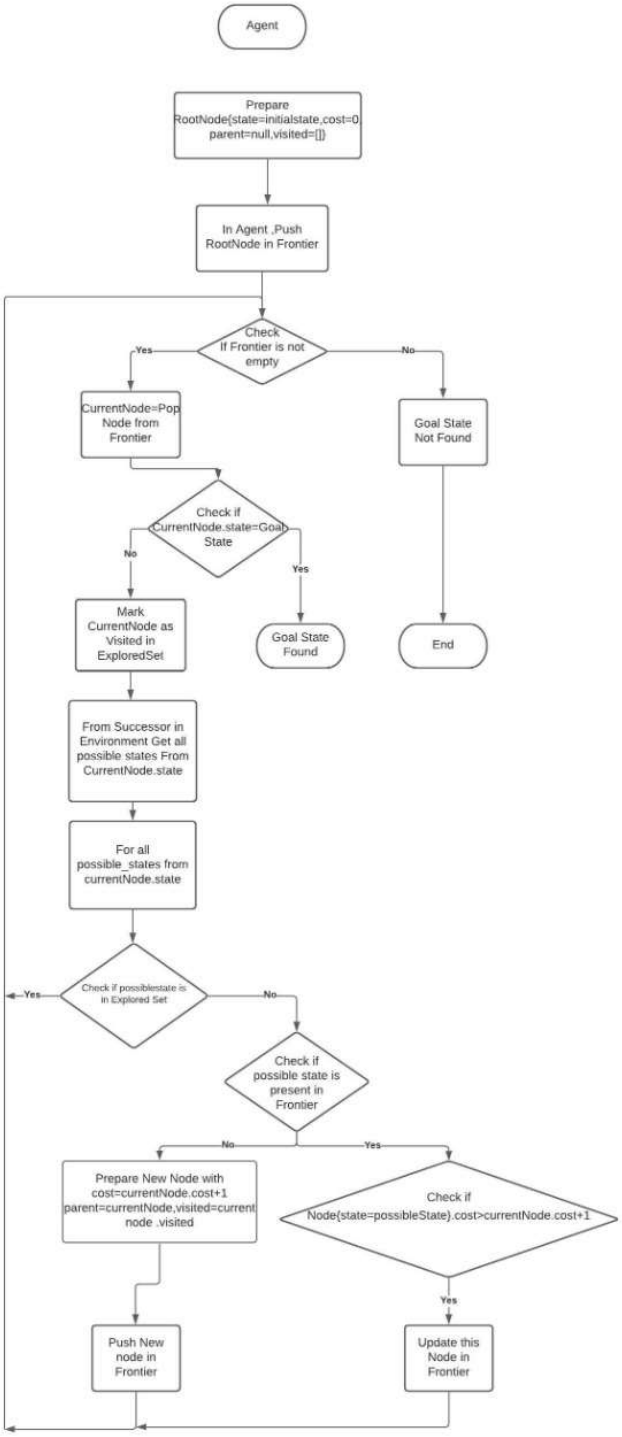

Fig. 1. Flow Chart of graph search agent

Algorithm 1 Graph Search Algorithm

```
Set environment for search by initializing start state, Goal state,hash map for explored nodes and Frontier queue
Initialize with the initial values of rootnode = start state,cost =0,parents,visited [] = in frontier
while Frontier is not empty do
    pop a element from frontier
    if CurrentNode.state = Goalstate then
        Goal state reached
    else
        mark this state as Explored node in ExploredSet.
        possiblestate ← currentstate
        for every state do
            if state not explored then
                if state present in frontier then
                    if new cost ¡ previous cost then
                        Update the cost, parent, visited
                    end if
                else
                    newNode with parent = currentNode
                    cost = currentNode.cost+1
                    visited = currentNode.visited+currentNode
                end if
                If Goal not found, path not found
            end if
        end for
    end if
end while
```

The $8Puzzle$ has a matrix of 3X3 in which numbers from 1 to 8 are arranged randomly with an empty space . In the given figure the start state is [[7,2,4][5,0,6][8,3,1]] where 0 is represented as the blank space , we need to reach the final state [[0,1,2][3,4,5][6,7,8]] by moving the tiles

In case of $8Puzzle$ the Start and Goal state are as follows:.

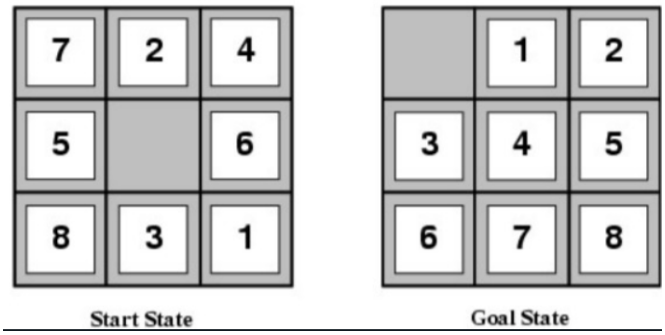

Fig. 2. Start state and Goal state of 8-Puzzle problem

B. Write a collection of functions imitating the environment for 8-Puzzle.

Answer: Environment for 8 puzzle includes current state visited state , path traversed to reach the current node and cost for reaching to the goal state *Function*
check(): If the current state is goal state
validate(): check if it is possible to move futhur or not
getnext():return next possible state

C. Describe what is Iterative Deepening Search ?

Answer: Iterative deepening search is search strategy used with depth first search(DFS).It does a series of depth first searches with increasing depth bounds(in every cycle it perfoms DFS with bound incremented by one) same way as that breadth first search(BFS), but the only difference is that BFS stores all nodes in memory, while iterative-deepening does it by repeating the previous layers, thereby saving memory at the cost of more time.Therefore memory requirements grows linearly with depth.The time complexity of iterative deepening is same as dfs i.e. **O(b^d)** where b is the branching factor and d is the depth . The space complexity is **O(bd)**.

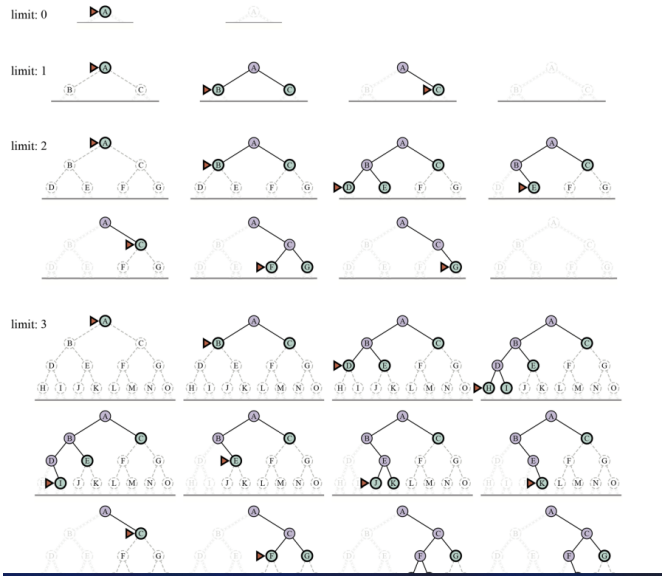

Fig. 3. Iterative Deepening Search

D. Considering the cost associated with every move to be the same (uniform cost), write a function which can backtrack and produce the path taken to reach the goal state from the source/ initial state

Answer:
Please refer to the code for given in github repository

E. Generate Puzzle-8 instances with the goal state at depth "d".

Answer:
For generating puzzle 8 instances with goal state at depth "d". We have to explore each possible states from initial function in order to reach the goal state. For implementation refer to the code

F. Prepare a table indicating the memory and time requirements to solve Puzzle-8 instances (depth "d") using your graph search agent.

Answer:

Depth	Space	time
1	12288	0.002992868423
2	16384	0.0059845
3	24576	0.01897954
4	98304	0.0446937084
5	98304	0.097214698
6	163840	0.1397825428
7	233472	0.2627568244
8	360448	0.345679283

TABLE I
LATITUDE AND LONGITUDE OF THE PLACES OF RAJASTHAN.

III. WEEK 3

Non-deterministic Search — Simulated Annealing

A. Travelling Salesman Problem (TSP) is a hard problem, and is simple to state. Given a graph in which the nodes are locations of cities, and edges are labelled with the cost of travelling between cities, find a cycle containing each city exactly once, such that the total cost of the tour is as low as possible.

Answer: Travelling Salesman Problem is popularly known as TSP problem. In this problems like "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" arises. It is an NP-hard problem in combinatorial optimization Solution to Travelling salesman problem:

1) Dynamic Programming(DP): The dynamic programming or DP method guarantees to find the best answer to TSP. However, its time complexity would exponentially increase with the number of cities. The time complexity with the DP method asymptotically equals N^2 x 2^N where N is the number of cities

2) Based on Euclidean distance: Let,

No. of cites = N

Total no of states = N!

here a city(i) is assumed to be present in x-y plane have coordinates = (x_i, y_i).

$D_{i,j}$ = Distance matrix of N x N which has the Euclidean distance between each city. Here a city is chosen through random permutation. Here we have to minimize the distance between cities and also find minimum distance between the origin and destination using the given equation :

$$min.f(n) = \sum_{i=1}^{N-1} d(x_i, x_{i+1}) + d(x_n, x_1) \tag{1}$$

3) Using Simulated Annealing: Here we apply stimulated annealing approach to solve tsp,as it provides an optimal global solution to the given problem.The fundamental idea is to accept moves resulting in solutions of worse quality than the current solution in order to escape from local Minima. The probability of accepting such a move is decreased during the search through parameter temperature. SA algorithm starts with an initial solution x , and candidate solution y is then generated (either randomly or using some pre-specified rule) from the neighbourhood of x. The Metropolis acceptance criterion, which models how a thermodynamic system moves from one state to another state in which the energy is being minimized, is used to decide whether to accept y or not. The candidate solution y is accepted as the current solution x based on the acceptance probability:

$$p = \begin{cases} 1 & if(f(y) \leq f(x)) \\ e^{-(f(y)-f(x))/t} & otherwise \end{cases} \tag{2}$$

Here t is the controlling parameter which plays an important role in computing the probability.Initially the t value is high and the process moves further in order to get the minimum cost we keep on decreasing the t by 0.1 after every iteration.

B. For the state of Rajasthan, find out at least twenty important tourist locations. Suppose your relatives are about to visit you next week. Use Simulated Annealing to plan a cost effective tour of Rajasthan. It is reasonable to assume that the cost of travelling between two locations is proportional to the distance between them.

Answer: Here we have taken twenty different places of Rajasthan and for calculating the distance, we first gathered there latitude and longitude for calculating the distance between the place. The gathered latitude and longitude of each place is shown in the table given below:

Sr.no	City	Latitude($N°$)	Longitude($E°$)
1	Udaipur	24.6	73.1
2	Jaisalmer	26.9	70.9
3	Ganganagar	29.9	73.8
4	Ajmer	26.4	74.6
5	Jaipur	26.9	75.7
6	Chittaurgarh	24.8	74.6
7	Alwar	27.5	76.6
8	Jodhpur	26.2	73.0
9	Dungarpur	23.8	73.7
10	Ranthambore	26.0	76.5
11	Pushkar	26.4	74.5
12	Kota	25.2	75.8
13	Bundi	25.4	75.6
14	Pali	25.7	73.3
15	Kumbalgarh	25.1	73.5
16	Bikaner	28.0	73.3
17	Abu	24.5	72.7
18	Ranakpur	25.1	73.4
19	Nathdwara	24.9	73.8
20	Fatehpur	25.9	80.7

TABLE II
LATITUDE AND LONGITUDE OF THE PLACES OF RAJASTHAN.

The distance between two places from given latitude and longitude can be found using Haversine-formula:
Resources: Harversine Formula

$$\text{d} = 2\text{rsin}^{-1}\sqrt{sin^2(\tfrac{N_2-N_1}{2}) + cos(N_2)cos(N_1)sin^2(\tfrac{E_2-E_1}{2})} \tag{3}$$

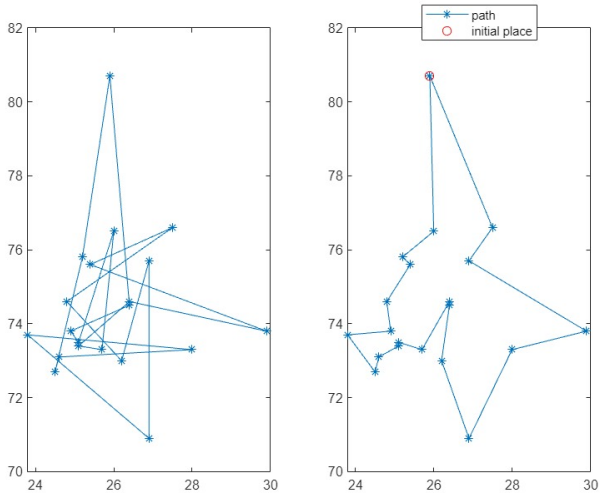

Fig. 4. Graphs showing difference between Initial travelling without simulated annealing and travelling with simulated annealing

After finding the distances from the Haversine-Formula we obtain distance between every cities

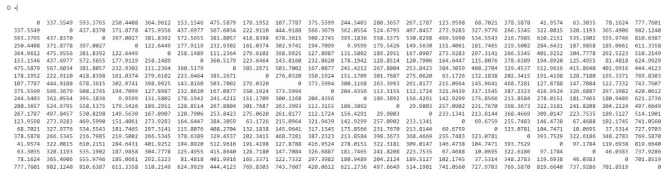

Fig. 5. Distance matrix cities

Total distance cost = 3033.9 **kms**.
Places traversed in the following manner :
Fatehpur,Alwar,Jaipur,Ganganagar,Bikaner,Jaisalmer,Jodhpur,Nathwara,
Chittaurgarh,Bundi,Kota,Ranthambore.

C. An interesting problem domain with TSP instances: VLSI: http://www.math.uwaterloo.ca/tsp/vlsi/index.html#XQF131 (Attempt at least five problems from the above list and compare your results.)

Answer:

1) **Board 1:** NAME: xqf131 COMMENT : Bonn VLSI data set with 131 points COMMENT : Uni Bonn, Research Institute for Discrete Math COMMENT : Contributed by Andre Rohe TYPE : TSP DIMENSION : 131 $EDGE\_WEIGHT\_TYPE$: EUC_2D

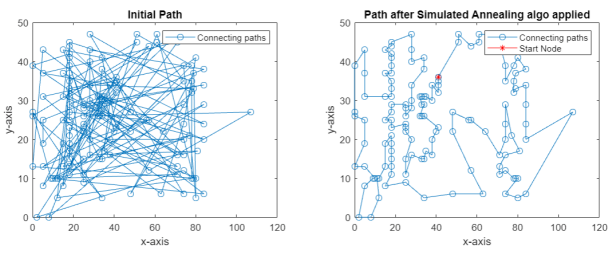

Fig. 6. Graphs showing difference between Initial travelling without simulated annealing and travelling with simulated annealing

2) **Board 2:** NAME : xqg237 COMMENT : Bonn VLSI data set with 237 points COMMENT : Uni Bonn, Research Institute for Discrete Math COMMENT : Contributed by Andre Rohe TYPE : TSP DIMENSION : 237 $EDGE\_WEIGHT\_TYPE$: EUC_2D

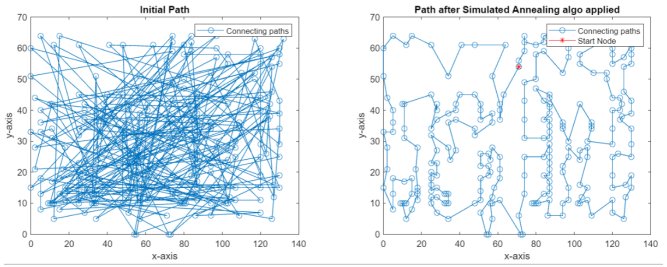

Fig. 7. Graphs showing difference between Initial travelling without simulated annealing and travelling with simulated annealing

3) **Board 3:** NAME : pbl395 COMMENT : Bonn VLSI data set with 395 points COMMENT : Uni Bonn, Research Institute for Discrete Math COMMENT : Contributed by Andre Rohe TYPE : TSP DIMENSION : 395 $EDGE\_WEIGHT\_TYPE$: EUC_2D $NODE\_COORD\_SECTION$

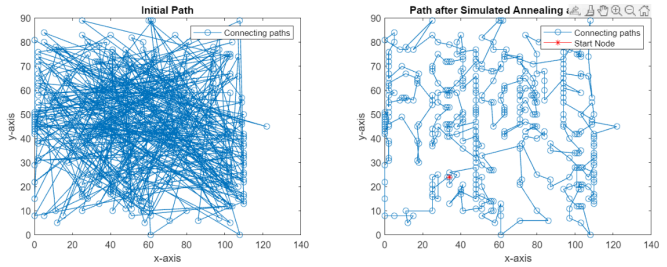

Fig. 8. Graphs showing difference between Initial travelling without simulated annealing and travelling with simulated annealing

4) **Board 4:** NAME : bcl380 COMMENT : Bonn VLSI data set with 380 points COMMENT : Uni Bonn, Research Institute for Discrete Math COMMENT : Contributed by Andre Rohe TYPE : TSP DIMENSION : 380 $EDGE\_WEIGHT\_TYPE$: EUC_2D $NODE\_COORD\_SECTION$

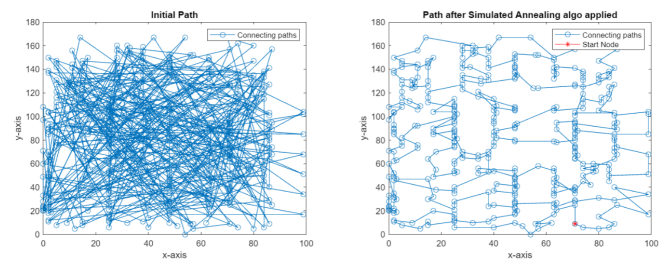

Fig. 9. Graphs showing difference between Initial travelling without simulated annealing and travelling with simulated annealing

5) **Board 5:** NAME : pbk411 COMMENT : Bonn VLSI data set with 411 points COMMENT : Uni

Bonn, Research Institute for Discrete Math COMMENT : Contributed by Andre Rohe TYPE : TSP DIMENSION : 411 $EDGE\_WEIGHT\_TYPE$: EUC_2D $NODE\_COORD\_SECTION$

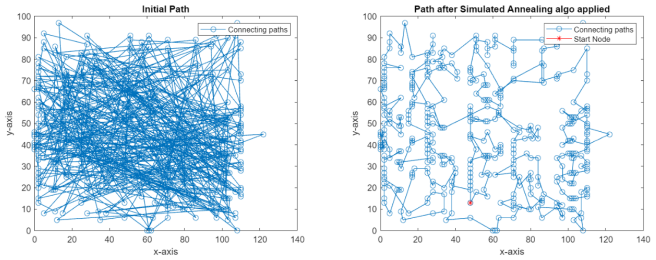

Fig. 10. Graphs showing difference between Initial travelling without simulated annealing and travelling with simulated annealing

IV. WEEK 5

Game Playing Agent — Minimax — Alpha-Beta Pruning Systematic adversarial search can lead to savings in terms of pruning of sub-trees resulting in lesser node evaluations

A. What is the size of the game tree for Noughts and Crosses? Sketch the game tree.

Answer: Noughts and crosses(tic-Tac-Toe) is a interesting game which require two players. Each player has to play alternatively. It has 3X3 Empty board which has to filled by 'X' and 'O'. Game continues till either of the player wins or the game draws. Let's suppose Player 1 starts the game so it has 9 choices of filling the box(assuming he will be fill 'X') which we can consider as depth 1, after player 1's chance Player 2 will have to fill 'O' in 8 possible choices. Then again Player1 gets chance to fill out of 7 choices, and this continues till a player wins or the game draws. For wining the game either following conditions must be satisfied:

- A row has same character i.e complete row has Use recurrence to show that under perfect ordering of leaf nodes, the alpha-beta pruning time complexity is O($b^{m/2}$), where b is the effective branching factor and m is the depth of the tree.to be either ('X' or 'O')
- A Column has same character i.e complete column has to be either ('X' or 'O')
- Either of the two diagonals have same character i.e complete diagonal has to be either ('X' or 'O')

So, Total possible states are 9! = 9x8x7x6x5x4x3x2x1 = 362880 states

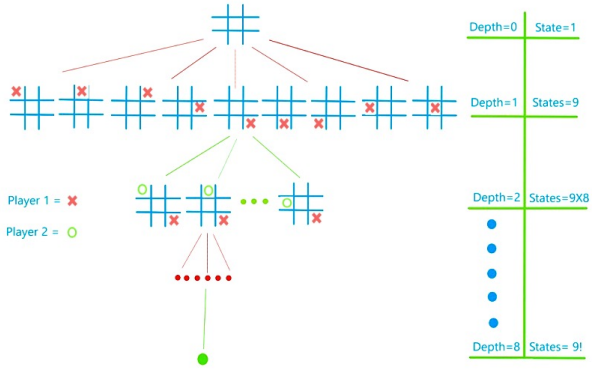

Fig. 11. Tic-Tac-Toe game tree

B. Read about the game of Nim (a player left with no move losing the game). For the initial configuration of the game with three piles of objects as shown in Figure, show that regardless of the strategy of player-1, player-2 will always win. Try to explain the reason with the MINIMAX value backup argument on the game tree.

Fig. 12. Hill Climbing with bdla Score

Answer: Game of Nim consists of 'n' piles where each pile contains i number of nims such that

$$1 \leq i \leq n \tag{4}$$

The player to pick the last nim is going to win the game The winning in the Nim game depends on the two factor:

- One who starts the game
- Initial configuration of game

Here with observation, if initial configuration of the Nims i.e if the XOR sum of the no. of the Nims is non-zero then who starts the game first will definitely lose. Here in the question starting configuration is [7,9,10] whose XOR is $4 \neq 0$, hence even if player 1 has the strategy of winning the game he will definitely lose.

C. Implement MINIMAX and alpha-beta pruning agents. Report on number of evaluated nodes for Noughts and Crosses game tree.

Answer: *number of different states explored during finding of best move for a state in Minimax algorithm:* 29

```
: print("Total number of different states explored during finding of best move for a state = ",len(states.keys()))
  Total number of different states explored during finding of best move for a state =  29
```

Fig. 13.

number of different states explored during finding of best move for a state in alpha-beta algorithm: 12

```
print("Total number of different states explored during finding of best move for a state = ",len(states.keys()))
Total number of different states explored during finding of best move for a state =  12
```

Fig. 14.

D. Use recurrence to show that under perfect ordering of leaf nodes, the alpha-beta pruning time complexity is O($b^{m/2}$), where b is the effective branching factor and m is the depth of the tree.

Answer: The time complexity of Minimax algorithm is O(b^m) which can reduced by without checking each node. The technique of reduction in time complexity is called pruning. This involves two threshold parameters alpha and beta, thus this is also called alpha-beta pruning.

To determine exact value of a state, we need the exact value of one of its children and bounds on the rest of the children.For determining a bound of state's value, exact value of one of its children must be known.

Let S(k) be the minimum number of states to be considered k ply from a given state for knowing the exact value of the state. Similarly let R(k) be the minimum number of states to be considered k ply from a given state when we need to know a bound on the state's value.Let b be the branching factor. Thus,

$$S(k) = S(k-1) + (b-1)R(k-1) \tag{5}$$

i.e., the exact value of one child and bounds on the rest,and

$$R(k) = S(k-1) \tag{6}$$

i.e., the exact value of one child. The base is S(0) = R(0) = 1. Note, for fig, this gives S(3) = b^2 + b -1 = 11 for b = 3

when we expand the recursive equation, we get:

S(k) = S(k-1)+(b-1)R(k-1)
= (S(k-2)+(b-1)R(k-2)) + (b-1)S(k-2)
= bS(k-2)+ (b-1)R(k-2)
= bS(k-2) + (b-1)S(k-3)

It is obvious that S(k-3)¡S(k-2),so: S(k) ¡ (2b-1)S(k-2) ¡ 2bS(k-2)

That is, the branching factor every two levels is less than 2b, which means the effective branching factor is less than $\sqrt{2b}$.

So, for even k, we derive $S(k) \leq (\sqrt{2b})^k$, which is not too far off the asymptotic upper bound of $(\sqrt{b} + \frac{1}{2})^{k+1}$

In effect, alpha-beta pruning can nearly double the depth that a game tree can be searched in comparison to straightforward minimax.

Resources: a-b-analysis

V. WEEK 6

Understand the graphical models for inference under uncertainty, build Bayesian Network in R, Learn the structure and CPTs from Data, naive Bayes classification with dependency between features.

A. Consider grades earned in each of the courses as random variables and learn the dependencies between courses.

Answer: Using the hill Climbing greedy search function 'hc' of the bnlearn package several different bayesian network based on scores have been generated.

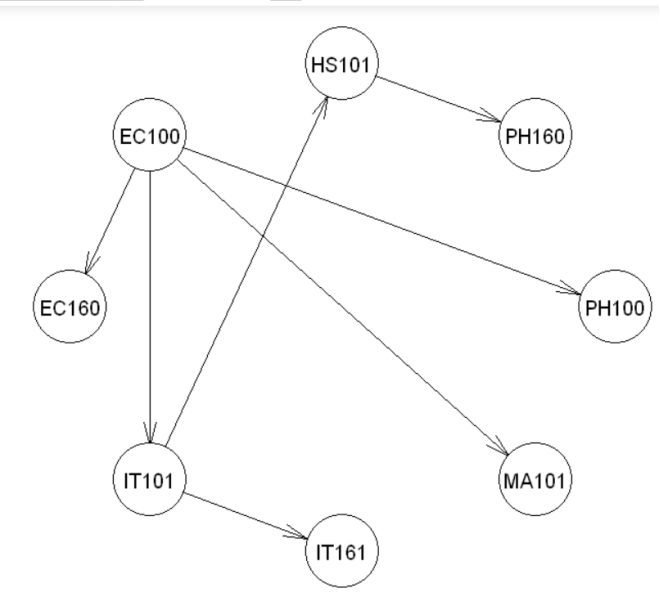

Fig. 15. Hill Climbing with aic Score

1) Using **'aic'** *score:* MODEL: There are 7 dependencies between each subject which can seen from the above fig. using aic score and the dependencies are [EC100][EC160—EC100][IT101—EC100] [MA101—EC100][PH100—EC100][IT161—IT101] [HS101—IT101][PH160—HS101]

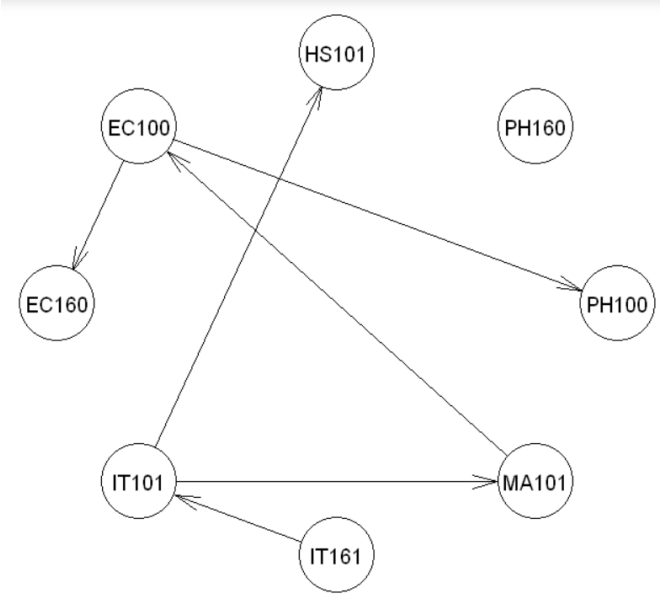

Fig. 16. Hill Climbing with bdla Score

2) Using **'bdla'** *score:* MODEL: There are 6 dependencies between each subject which can seen from the above fig. using bdla score and the dependencies are [IT161][PH160][IT101—IT161][MA101—IT101] [HS101—IT101][EC100—MA101][EC160—EC100] [PH100—EC100].

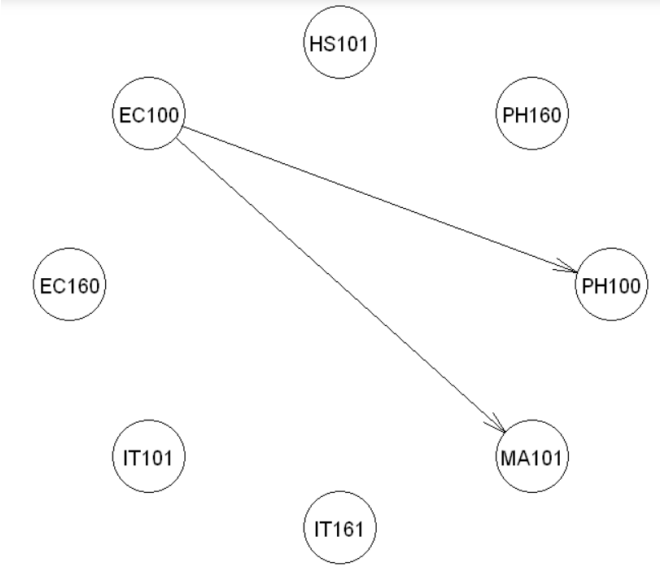

Fig. 17. Hill Climbing with bic Score

3) Using **'bic'** *score:* MODEL: There are 2 dependencies between each subject which can seen from the above fig. using bic score and the dependencies are [EC100][EC160][IT101][IT161][PH160] [HS101][MA101—EC100][PH100—EC100].

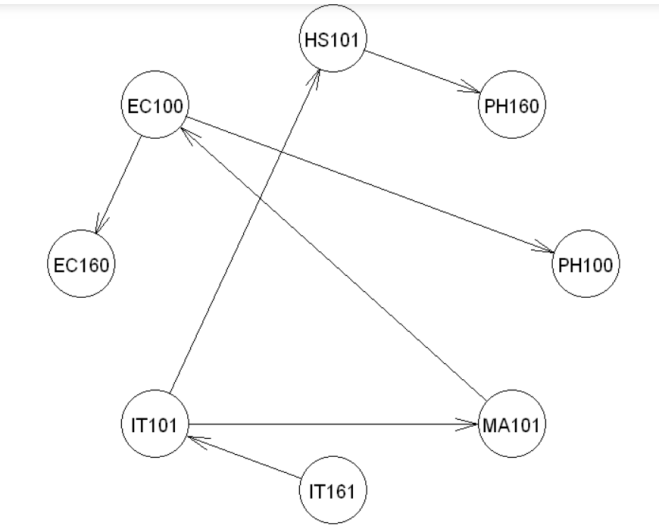

Fig. 18. Hill Climbing with k2 Score

4) Using **'k2'** *score:* MODEL: There are 7 dependencies between each subject which can seen from the above fig. using k2 score and the dependencies are [IT161][IT101—IT161][MA101—IT101][HS101—IT101] [EC100—MA101][PH160—HS101][EC160—EC100] [PH100—EC100]

B. Using the data, learn the CPTs for each course node.

Using hill climbing K2 score the CPTs of each course is plotted below:

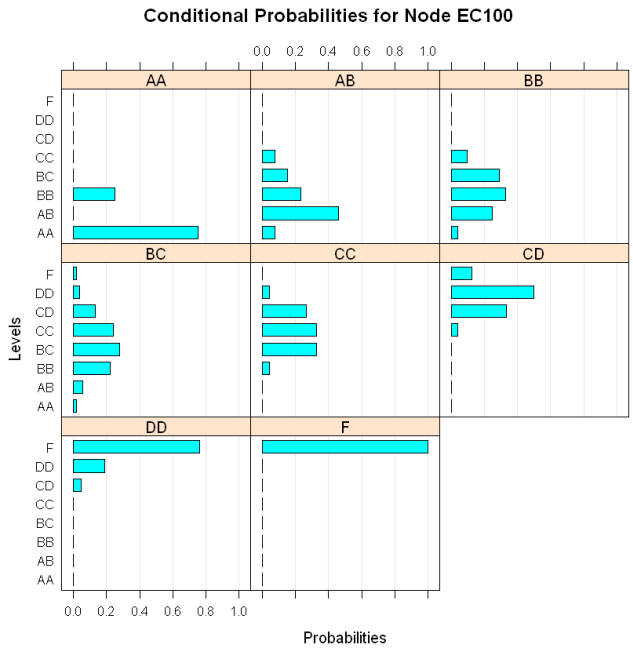

Fig. 19. CPT of EC100

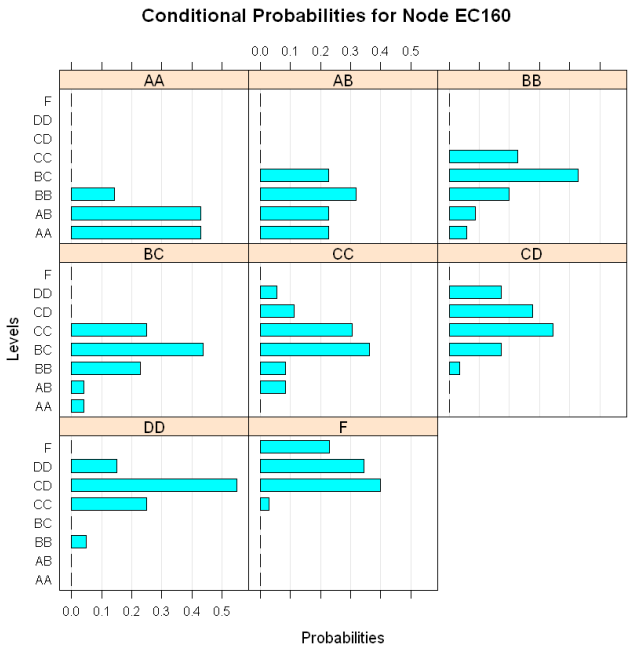

Fig. 20. CPT of EC100

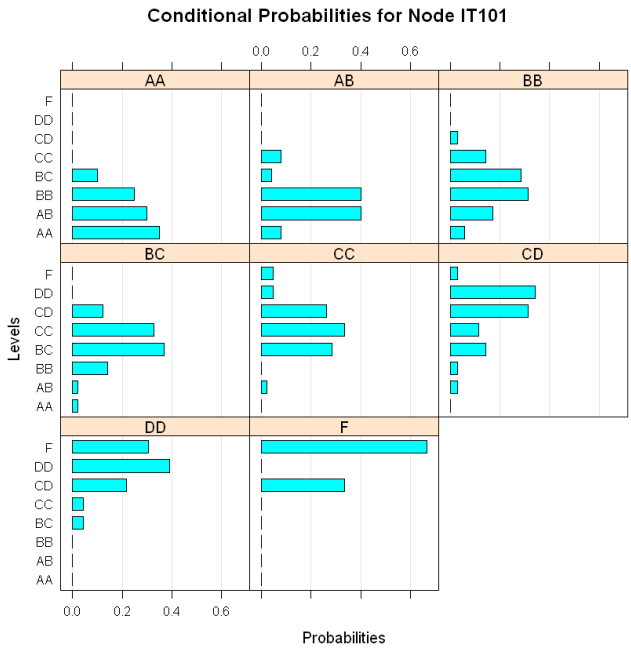

Fig. 22. CPT of EC100

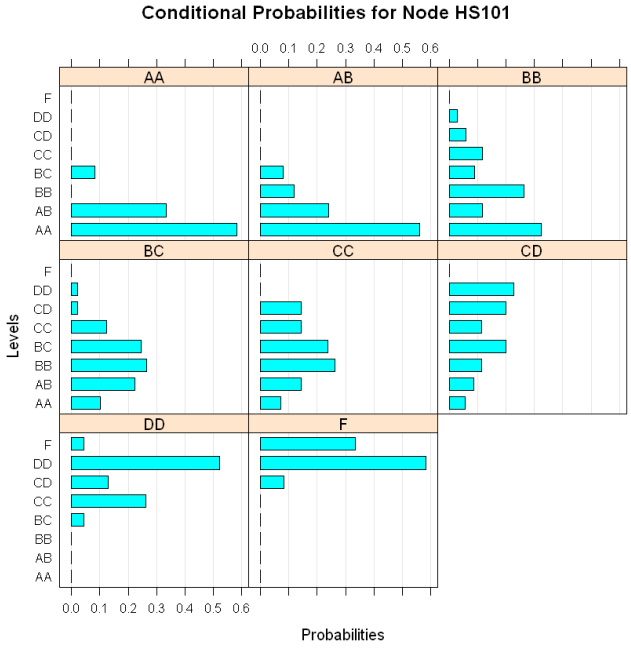

Fig. 21. CPT of EC100

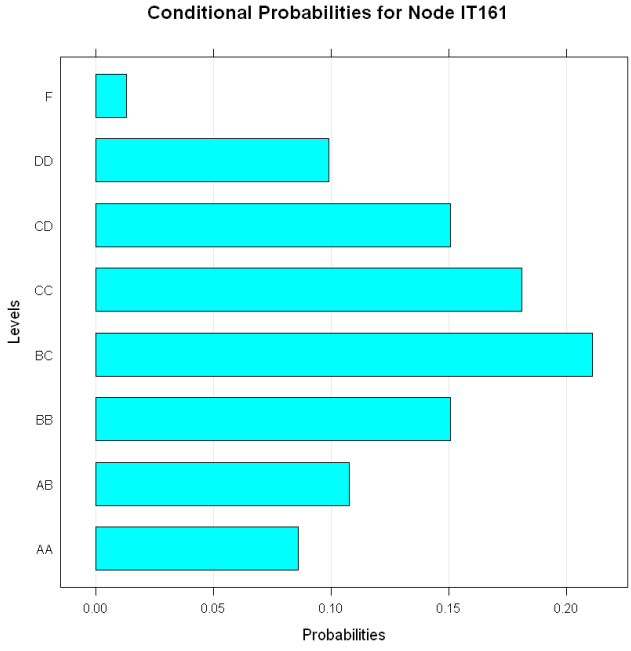

Fig. 23. CPT of EC100

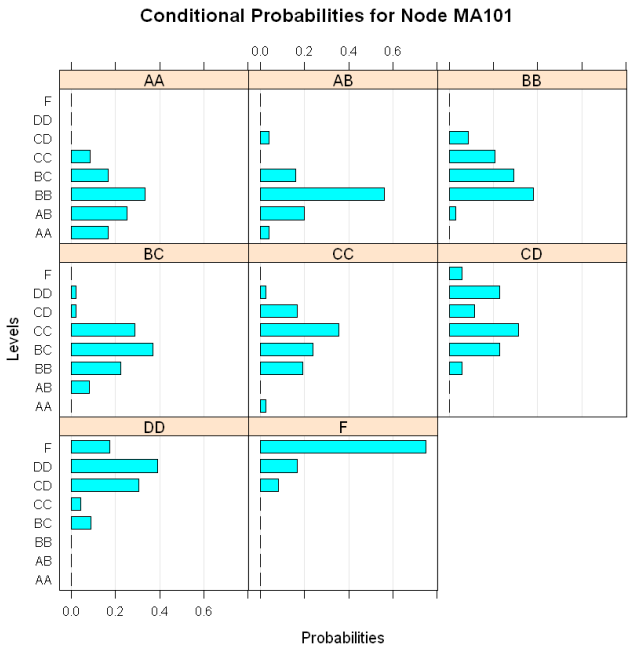

Fig. 24. CPT of EC100

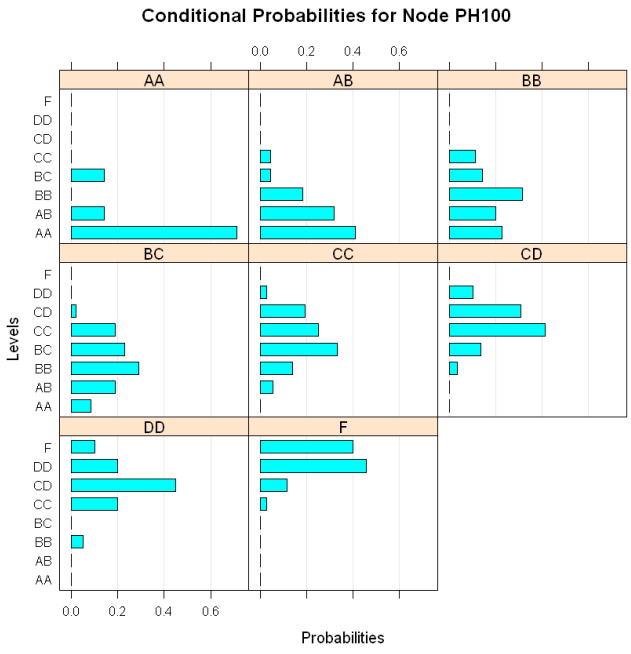

Fig. 25. CPT of EC100

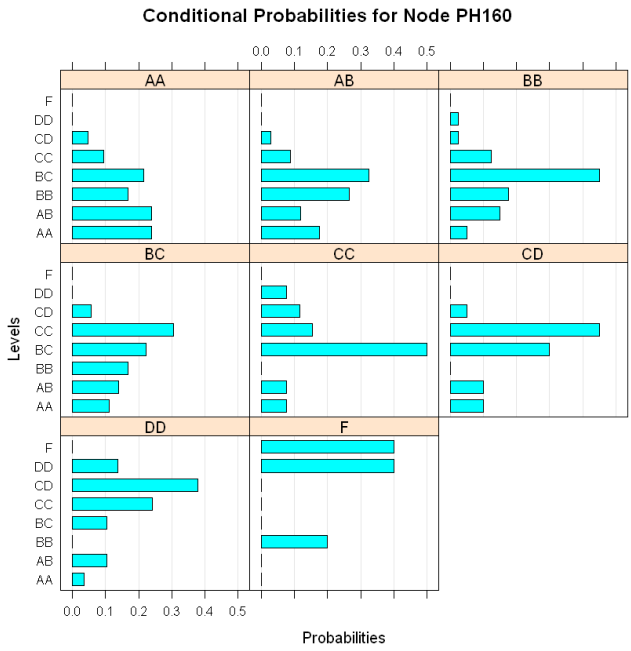

Fig. 26. CPT of EC100

C. What grade will a student get in PH100 if he arns DD in EC100, CC in IT101 and CD in MA101.Any of this trained model would give the same prediction about the grade

Answer: We use cpquery method for getting the value that match with given condition. After applying method it is most likely to 'DD' grade in PH100

D. The last column in the data file indicates whether a student qualifies for an internship program or not. From the given data, take 70 percent data for training and build a naive Bayes classifier (considering that the grades earned in different courses are independent of each other) which takes in the student's performance and returns the qualification status with a probability. Test your classifier on the remaining 30 percent data. Repeat this experiment for 20 random selection of training and testing data. Report results about the accuracy of your classifier.

Answer: This networks works on splited data which we have use-in for training the model.In the second case, dependencies to exits among courses as well with QP through an direct acyclic graph(DAG).We here use Naive Bayes network from bnclassify. The structure is shown below:

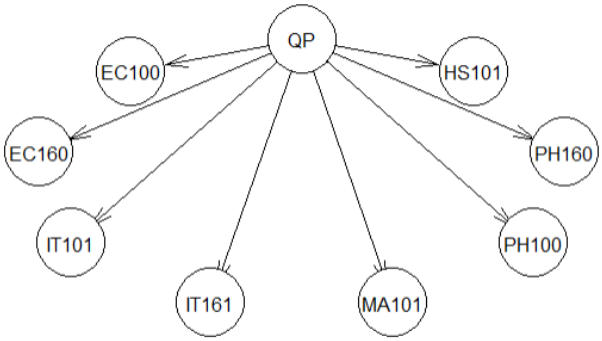

Fig. 27. Naive Bayes Independent data

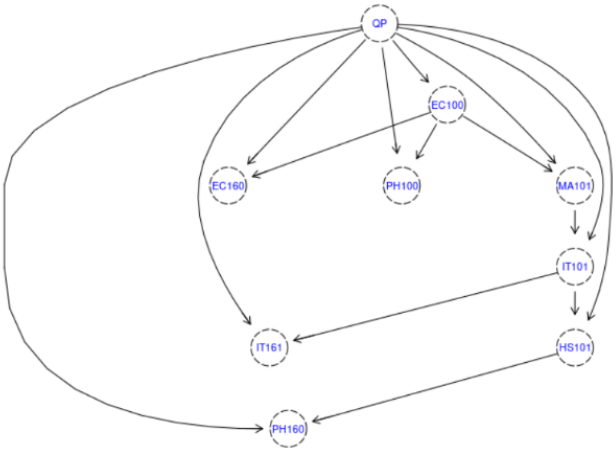

Fig. 28. Naive Bayes dependent data

This network learns on the training dataset that we split before and test it on test data.The Table below shows the accuracy:

Experiment No.	Accuracy
1	0.987013
2	0.974026
3	0.987013
4	0.974026
5	0.987013
6	0.974026
7	0.987013
8	0.9871795
9	0.961039
10	0.9615385
11	0.9871795
12	0.974026
13	0.987013
14	1
15	0.987013
16	0.974026
17	1
18	0.974026
19	0.987013
20	0.974026

TABLE III
ACCURACY ON TEST DATASET ON NAIVE BAYES CLASSIFIER IN INDEPENDENT DATA

Experiment No.	Accuracy
1	0.8974359
2	0.961039
3	0.9358974
4	0.974359
5	0.9473684
6	0.9487179
7	0.9480519
8	0.9480519
9	0.974026
10	0.9615385
11	0.9342105
12	0.9358974
13	0.9480519
14	0.9480519
15	0.9487179
16	0.9220779
17	0.8961039
18	0.9615385
19	0.9868421
20	0.9342105

TABLE IV
ACCURACY ON TEST DATASET ON NAIVE BAYES CLASSIFIER IN DEPENDENT DATA

E. Repeat 4, considering that the grades earned in different courses may be dependent.

Answer:

VI. LINK FOR LABORATORY CODE

The complete codes for different Laboratories is given below:
GitHub