# Indain Institute of Information Technology, Vadodara

# Artificial Intelligence CS362
# Pre-EndSem Lab Report

Members:
**Sagar Deware (202051166)**
**Yash Agrawal (202051205)**
**Bibhav Shah (202051212)**
**Sushil Patel (202051188)**

Lecturer:
**Dr. Pratik Shah**

Academic Batch 2020-24

# Lab 06 Report CS362 AI

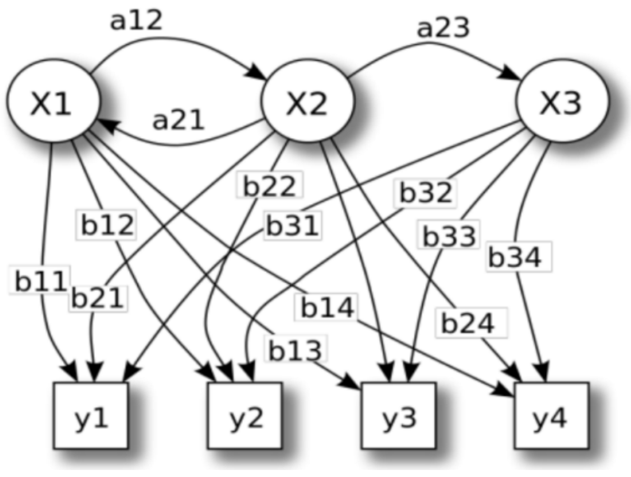
Sagar Deware *202051166* | Yash Agrawal *202051205* | Bibhav Kumar shah *202051212* | Sushil Petal *202051188*

***Abstract*——This lab report is the documentation of our CS362 coursework offered at the Indian Institute of Information Technology Vadodara.**

## I. LAB 6

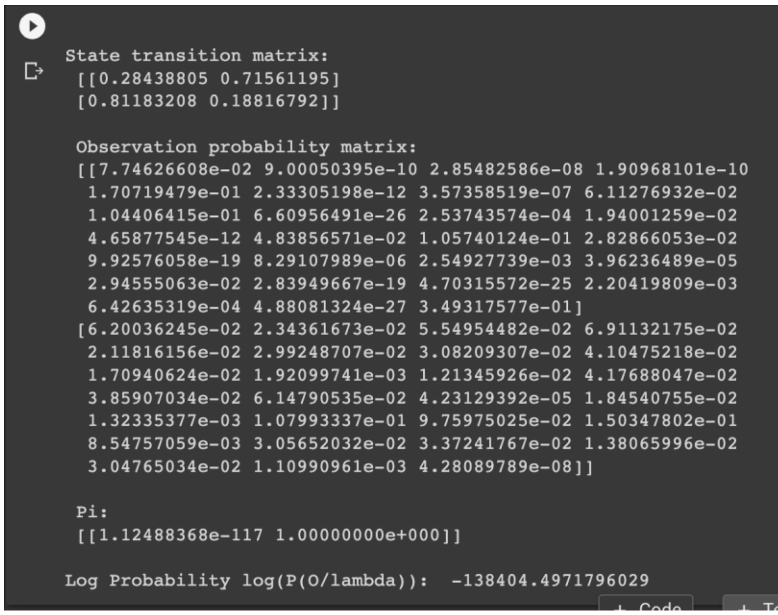
A statistical model called the Hidden Markov Model (HMM) asserts that the system under investigation is a Markov process with unobservable or "hidden" states, called X. also HMM

expects that Y, a process dependent on X, is a real process. The goal of HMM is to learn more about X through analyzing Y. Thermodynamics, statistical mechanics, physics, chemistry, economics, finance, signal processing, information theory, pattern recognition, and biological research are just a few of the domains where HMM has a wide range of applications. In other words, the prior distribution of hidden states (transition probabilities) and the conditional distribution of observations given states (emission probabilities) are modeled by the HMM, a generative model that simulates the combined distribution of hidden states and observations.



X: states, y: possible observations

## II. Output



```
State transition matrix:
 [[0.28438805 0.71561195]
 [0.81183208 0.18816792]]

 Observation probability matrix:
 [[7.74626608e-02 9.00050395e-10 2.85482586e-08 1.90968101e-10
  1.70719479e-01 2.33305198e-12 3.57358519e-07 6.11276932e-02
  1.04406415e-01 6.60956491e-26 2.53743574e-04 1.94001259e-02
  4.65877545e-12 4.83856571e-02 1.05740124e-01 2.82866053e-02
  9.92576058e-19 8.29107989e-06 2.54927739e-03 3.96236489e-05
  2.94555063e-02 2.83949667e-19 4.70315572e-25 2.20419809e-03
  6.42635319e-04 4.88081324e-27 3.49317577e-01]
 [6.20036245e-02 2.34361673e-02 5.54954482e-02 6.91132175e-02
  2.11816156e-02 2.99248707e-02 3.08209307e-02 4.10475218e-02
  1.70940624e-02 1.92099741e-03 1.21345926e-02 4.17688047e-02
  3.85907034e-02 6.14790535e-02 4.23129392e-05 1.84540755e-02
  1.32335377e-03 1.07993337e-01 9.75975025e-02 1.50347802e-01
  8.54757059e-03 3.05652032e-02 3.37241767e-02 1.38065996e-02
  3.04765034e-02 1.10990961e-03 4.28089789e-08]]

 Pi:
 [[1.12488368e-117 1.00000000e+000]]

Log Probability log(P(O/lambda)):  -138404.4971796029
```

Output of the code

## III. Reference

https://colab.research.google.com/drive/10Ao5KfRGgZtGcWtqrETC9ef

# AI Colossus

Sushil kumar patel
Yash Agarwal
Bibhav shah
Sagar deware

## Lab Assignment 8

### I. OBJECTIVES

A fundamental understanding of data structures is necessary for performing state-space search tasks, while the utilization of random numbers is essential in the context of Markov decision processes (MDP) and reinforcement learning (RL).

### II. PROBLEM STATEMENT

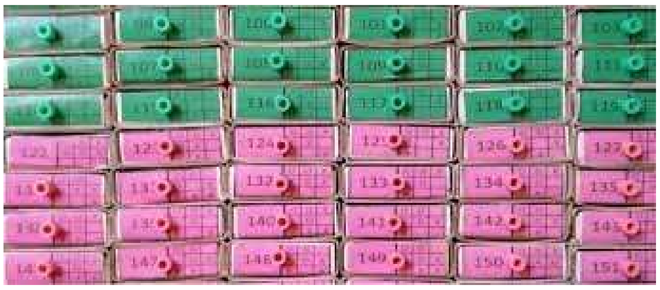
Consult Michie's reference on MENACE and explore its various implementations. Choose your preferred implementation and thoroughly review its code, emphasizing the essential components. If feasible, attempt to write your own version of MENACE using a programming language of your choice.

### III. EXPLANATION

MENACE (the Machine Educable Noughts And Crosses Engine) is a machine learning computer built from 304 matchboxes.

MENACE (the Machine Educable Noughts And Crosses Engine) is a machine learning computer built from 304 matchboxes.\\\\

MENACE is an algorithm designed to play Tic Tac Toe by utilizing a reinforcement learning technique to modify its approach according to the results of past games. MENACE can be regarded as a rudimentary reinforcement learning algorithm because it adapts its strategy based on the reward signal received from winning or losing a game.



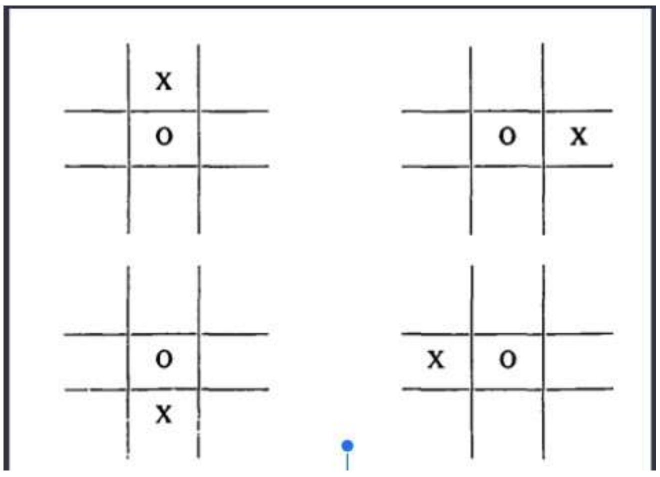
*A. Algorithm*

1) MENACE is comprised of multiple matchboxes, with each box representing a distinct configuration of the game board. Inside each matchbox, there are several colored beads that correspond to different moves that MENACE could potentially make.When MENACE plays Tic-Tac-Toe, it selects a matchbox that represents the current game board configuration.

2) MENACE then draws a bead at random from the matchbox to represent its move for that turn.

3) Following each game, MENACE is either rewarded or penalized depending on the result. If MENACE emerges victorious, it receives a reward, which takes the form of extra beads added to the matchbox. Conversely, if MENACE loses or draws, it is penalized, and beads are removed from the matchbox.

4) Using the feedback provided by the rewards and penalties, MENACE gradually learns which moves are more likely to lead to victory. It then modifies the number of beads in each matchbox to reflect this, thereby improving the likelihood of choosing optimal moves in future games.



5) MENACE gradually improves its gameplaying ability and becomes increasingly difficult to defeat through this trial-and-error process

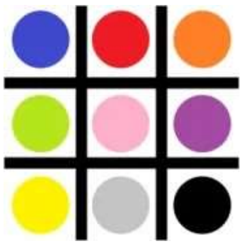
*B. Symmetric Layouts of Noughts and crosses game*

The Noughts and crosses game game board is symmetric, so many of the states are actually similar states if rotated or taken mirror image of the board. So we remove all the duplicate states that we generated. After this we are left with only around 900 distinct states out of 255,168 states.

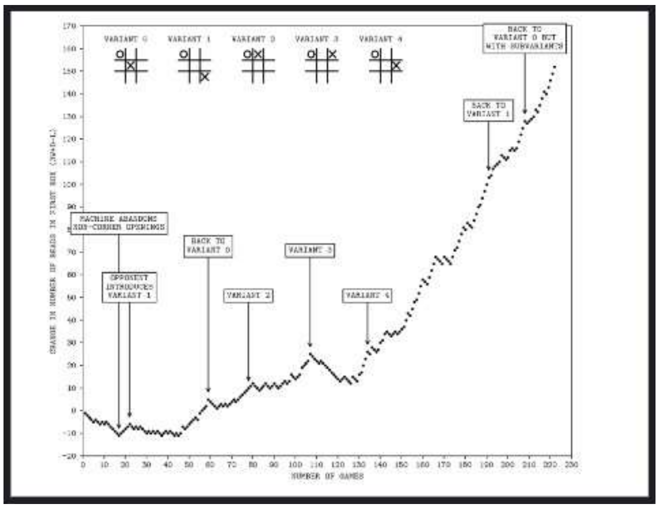
IV. RESULTS

*C. Colour Code*

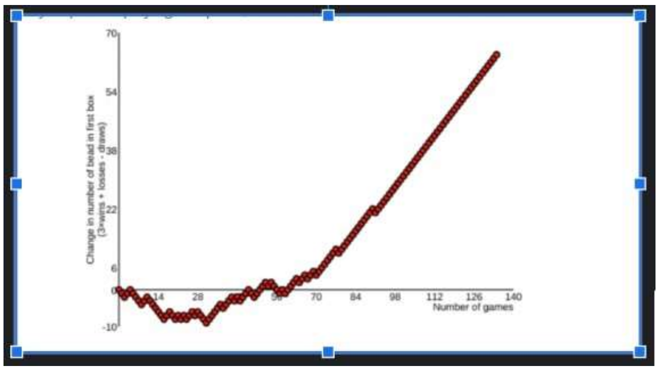
The 9 position in naughts and crosses has different color beads associated to it. Refer the following Figure.



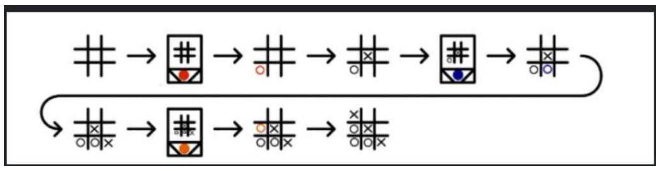
*D. Rules to play the MENACE:*



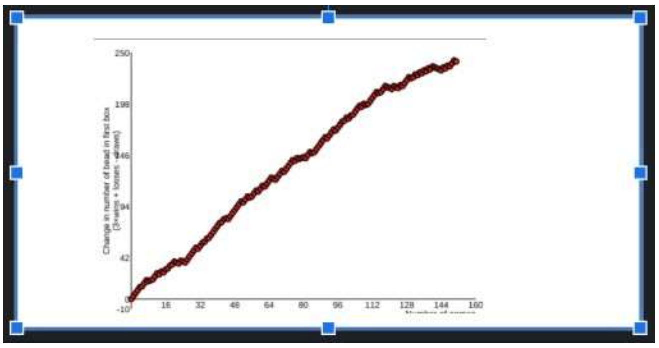
The progress of MENACE'S maiden tournament against a human opponent. The line of dots drops one level for a defeat, rises one level for a draw and rises three levels for a victory.



When MENACE plays a perfect-playing computer, the results look like this: The Red colour symbolises that most of the games were draw!

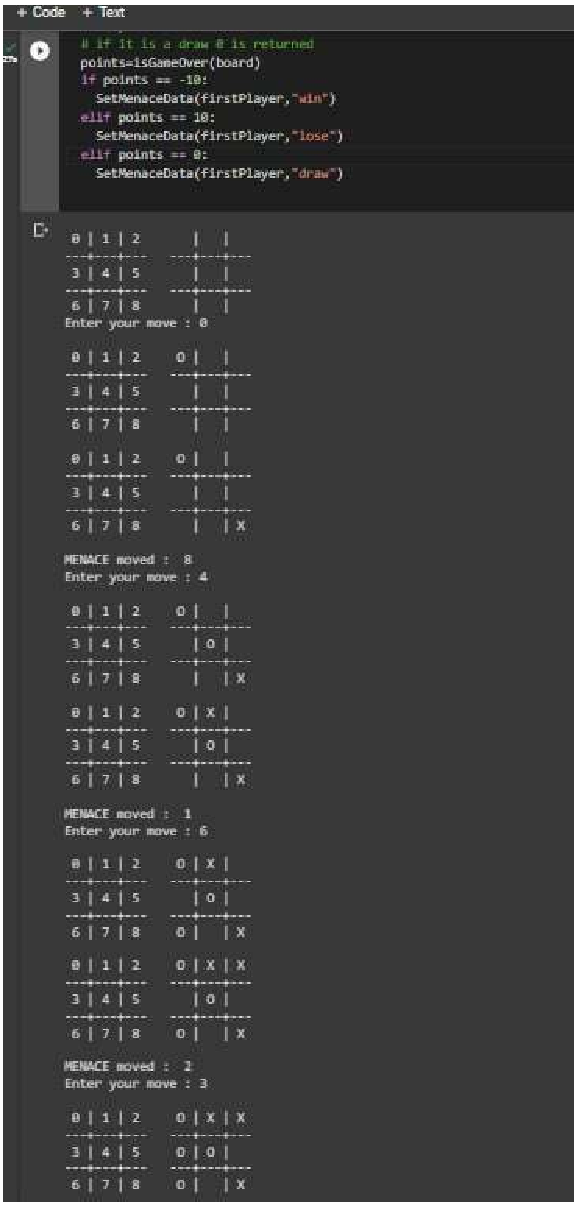


MENACE lost the game above, so the beads that were chosen are removed from the boxes. This means that MENACE will be less likely to pick the same colours again and has learned. If MENACE had won, three beads of the chosen colour would have been added to each box, encouraging MENACE to do the same again. If a game is a draw, one bead is added to each box



When MENACE played a with a random picking opponent, the result is a near-perfect positive correlation

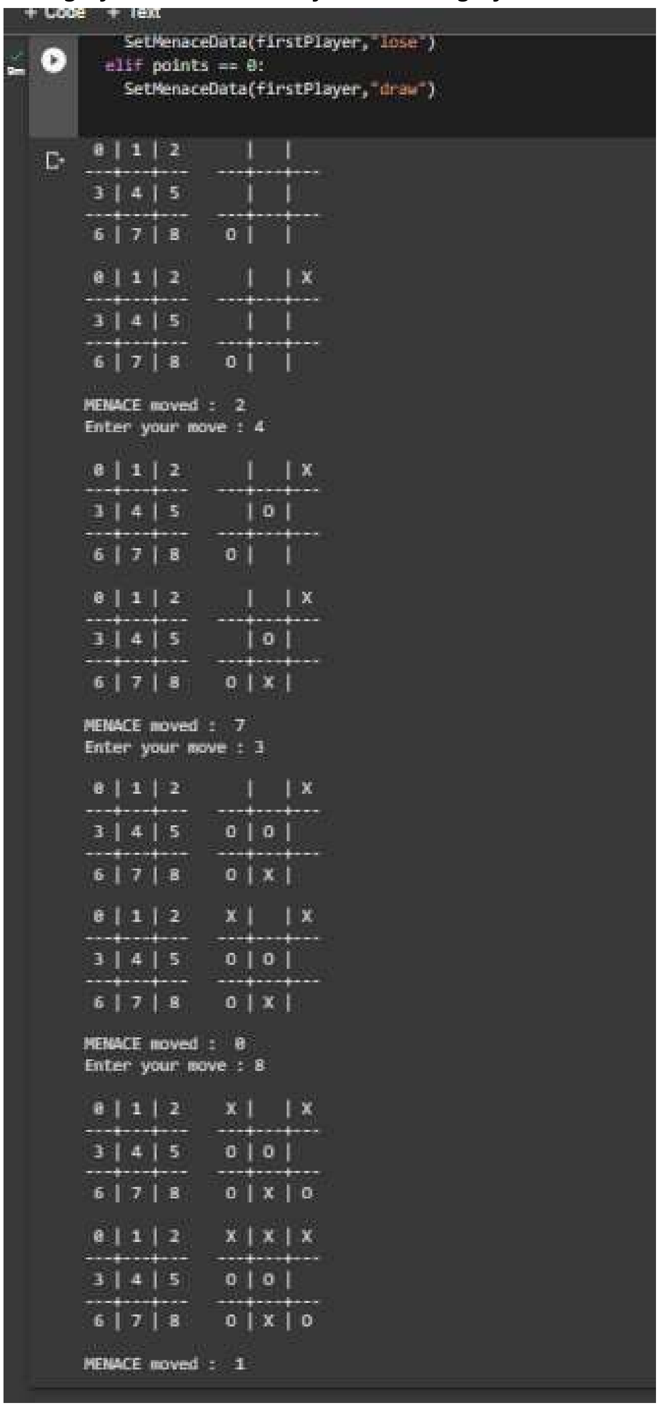
V. CODE RESULTS *B. After training of MENACE A. Before training of MENACE*



+ Code + Text

```
# if it is a draw 0 is returned
points=isGameOver(board)
if points == -10:
  SetMenaceData(firstPlayer,"win")
elif points == 10:
  SetMenaceData(firstPlayer,"lose")
elif points == 0:
  SetMenaceData(firstPlayer,"draw")
```

```
 0 | 1 | 2     |   |
---+---+---  ---+---+---
 3 | 4 | 5     |   |
---+---+---  ---+---+---
 6 | 7 | 8     |   |
Enter your move : 0

 0 | 1 | 2   O |   |
---+---+---  ---+---+---
 3 | 4 | 5     |   |
---+---+---  ---+---+---
 6 | 7 | 8     |   |

 0 | 1 | 2   O |   |
---+---+---  ---+---+---
 3 | 4 | 5     |   |
---+---+---  ---+---+---
 6 | 7 | 8     |   | X

MENACE moved :  8
Enter your move : 4

 0 | 1 | 2   O |   |
---+---+---  ---+---+---
 3 | 4 | 5     | O |
---+---+---  ---+---+---
 6 | 7 | 8     |   | X

 0 | 1 | 2   O | X |
---+---+---  ---+---+---
 3 | 4 | 5     | O |
---+---+---  ---+---+---
 6 | 7 | 8     |   | X

MENACE moved :  1
Enter your move : 6

 0 | 1 | 2   O | X |
---+---+---  ---+---+---
 3 | 4 | 5     | O |
---+---+---  ---+---+---
 6 | 7 | 8   O |   | X

 0 | 1 | 2   O | X | X
---+---+---  ---+---+---
 3 | 4 | 5     | O |
---+---+---  ---+---+---
 6 | 7 | 8   O |   | X

MENACE moved :  2
Enter your move : 3

 0 | 1 | 2   O | X | X
---+---+---  ---+---+---
 3 | 4 | 5   O | O |
---+---+---  ---+---+---
 6 | 7 | 8   O |   | X
```



+ Code + Text

```
  SetMenaceData(firstPlayer,"lose")
elif points == 0:
  SetMenaceData(firstPlayer,"draw")
```

```
 0 | 1 | 2     |   |
---+---+---  ---+---+---
 3 | 4 | 5     |   |
---+---+---  ---+---+---
 6 | 7 | 8   O |   |

 0 | 1 | 2     |   | X
---+---+---  ---+---+---
 3 | 4 | 5     |   |
---+---+---  ---+---+---
 6 | 7 | 8   O |   |

MENACE moved :  2
Enter your move : 4

 0 | 1 | 2     |   | X
---+---+---  ---+---+---
 3 | 4 | 5     | O |
---+---+---  ---+---+---
 6 | 7 | 8   O |   |

 0 | 1 | 2     |   | X
---+---+---  ---+---+---
 3 | 4 | 5     | O |
---+---+---  ---+---+---
 6 | 7 | 8   O | X |

MENACE moved :  7
Enter your move : 3

 0 | 1 | 2     |   | X
---+---+---  ---+---+---
 3 | 4 | 5   O | O |
---+---+---  ---+---+---
 6 | 7 | 8   O | X |

 0 | 1 | 2   X |   | X
---+---+---  ---+---+---
 3 | 4 | 5   O | O |
---+---+---  ---+---+---
 6 | 7 | 8   O | X |

MENACE moved :  0
Enter your move : 8

 0 | 1 | 2   X |   | X
---+---+---  ---+---+---
 3 | 4 | 5   O | O |
---+---+---  ---+---+---
 6 | 7 | 8   O | X | O

 0 | 1 | 2   X | X | X
---+---+---  ---+---+---
 3 | 4 | 5   O | O |
---+---+---  ---+---+---
 6 | 7 | 8   O | X | O

MENACE moved :  1
```

# Lab 10 Markov Decision Process(MDP) and Dynamic programming

Sagar Deware
*202051166*

Yash Agrawal
*202051205*

Bibhav Kumar shah
*202051212*

Sushil Petal
*202051188*

***Abstract*—This lab assignment focuses on using the Markov Decision Process and Dynamic Programming to comprehend the sequential decision-making process in a stochastic setting. In the first issue, an agent must pick actions to reach target states while coping with stochasticity and reaping rewards in a 4x3 environment. The best policy must be found through value iteration.**
**The remaining issues entail defining and resolving a finite MDP problem that is ongoing for Gbike's bike rental service using modified policy iteration. The answers to these issues are presented in the lab report, emphasising the use of ideas like value iteration, policy iteration, and the Bellman equation. The research also provides code and comparisons with existing Gbike problem implementations. Understanding the practical applications of reinforcement learning and making decisions in the face of uncertainty can be improved with this lab assignment.**

***Index Terms*—component, formatting, style, styling, insert**

## I. Problem Statement

The purpose of this lab project is to use the ideas of MDP and DP to address two different issues. Finding the best course of action for an agent to follow in a stochastic environment is the first issue.
The goal of the second issue, which involves a bike rental service, is to maximise profit by maximising the number of bikes available at two sites. Understanding the MDP framework, the Bellman equation, and value/policy iteration algorithms is essential for solving these issues. The answers to these issues will shed light on how RL algorithms might be used in real-world applications and the difficulties of making decisions in the face of ambiguity.

## II. Introduction

### A. Reinforcement learning:

A branch of artificial intelligence called reinforcement learning (RL) studies how agents might learn to make decisions by interacting with their surroundings. A common mathematical framework in RL called Markov Decision Process (MDP) represents a decision-making problem as a stochastic process. Finding an ideal policy—a mapping from states to actions that maximises a long-term objective function—is the aim. A set of algorithms known as dynamic programming (DP) solves MDPs by iteratively enhancing the value function, which symbolises the anticipated cumulative rewards from a particular state under a policy.

*1) Markov decision process:* A mathematical framework called a Markov Decision Process (MDP) is used to model decision-making issues in unpredictable situations. It is a framework for issues involving sequential decision-making in which an agent continuously engages with its surroundings. In an MDP, the agent's activities determine the state transition probabilities between a collection of states that represent the environment. Every action the agent does is rewarded, and the objective is to create a policy that maximises the expected cumulative benefit over time.
MDPs consist of a few essential parts. The agent has a range of possible states that it can be in at any given time, to start. Second, the agent has a range of options that, with a certain degree of probability, will cause a change in state. The reward the agent receives for changing states is determined by a reward function, which is the third component. Last but not least, there is a discount factor that establishes how significant long-term rewards are in comparison to short-term rewards.
In reinforcement learning, a branch of machine learning focused on teaching agents to make decisions based on feedback from their environment, MDPs are frequently utilised. To determine the best policy for an MDP, reinforcement learning algorithms frequently employ dynamic programming techniques like value iteration or policy iteration.

## III. Approach

The following strategy can be used to determine the best policy using value iteration and to solve the Gbike bike rental problem using policy iteration:

### A. Problem I

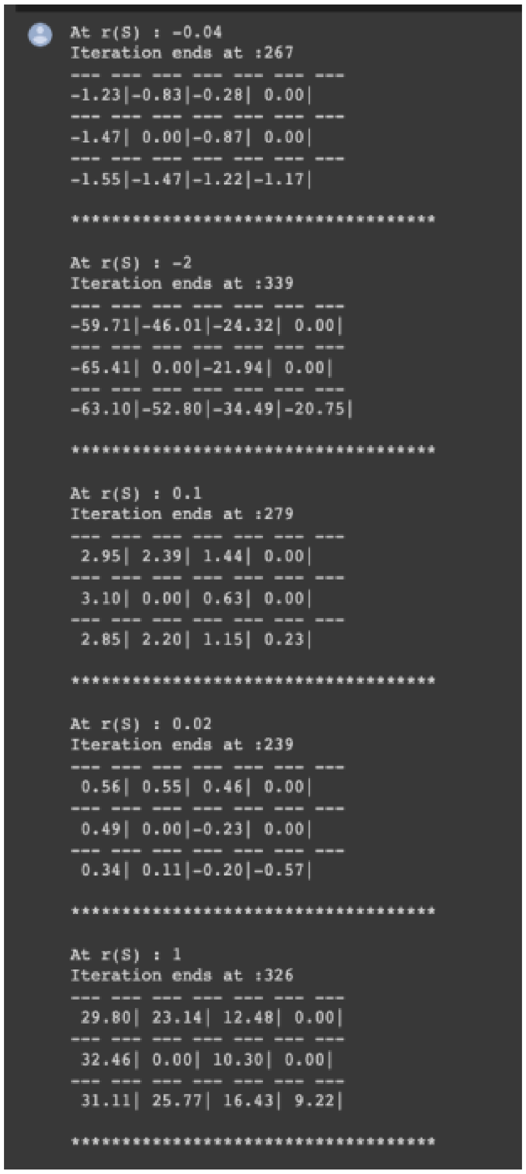
- Define the 4x3 environment's state space, action space, reward function, and transition probability matrix.
- To determine the best value function and best policy for each state, use the value iteration algorithm.
- Use the optimal policy to find the optimal sequence of actions to reach the terminal state with maximum reward.

### B. Problem II

- For the Gbike bicycle rental problem, define the state space, action space, reward function, and transition probability matrix.
- Use the policy iteration method to identify the best course of action for every state.

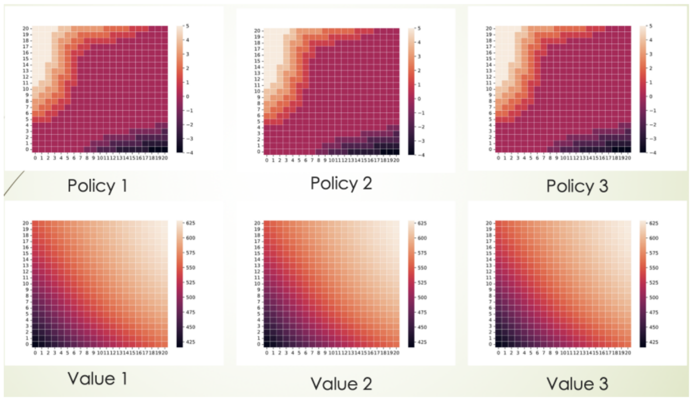
- Change the incentive mechanism to reflect the extra expense of moving bikes and using a second parking area.
- Use the optimal policy to find the optimal sequence of actions to maximize the expected profit for the Gbike bicycle rental problem.

## IV. Result

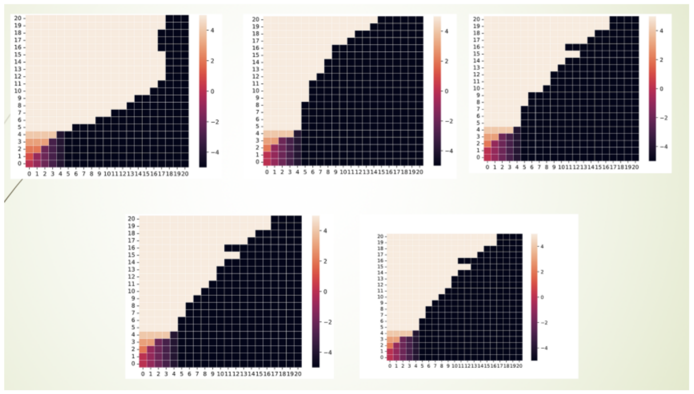


```
At r(S) : -0.04
Iteration ends at :267
--- --- --- --- --- --- ---
-1.23|-0.83|-0.28| 0.00|
--- --- --- --- --- --- ---
-1.47| 0.00|-0.87| 0.00|
--- --- --- --- --- --- ---
-1.55|-1.47|-1.22|-1.17|

************************************

At r(S) : -2
Iteration ends at :339
--- --- --- --- --- --- ---
-59.71|-46.01|-24.32| 0.00|
--- --- --- --- --- --- ---
-65.41| 0.00|-21.94| 0.00|
--- --- --- --- --- --- ---
-63.10|-52.80|-34.49|-20.75|

************************************

At r(S) : 0.1
Iteration ends at :279
--- --- --- --- --- --- ---
 2.95| 2.39| 1.44| 0.00|
--- --- --- --- --- --- ---
 3.10| 0.00| 0.63| 0.00|
--- --- --- --- --- --- ---
 2.85| 2.20| 1.15| 0.23|

************************************

At r(S) : 0.02
Iteration ends at :239
--- --- --- --- --- --- ---
 0.56| 0.55| 0.46| 0.00|
--- --- --- --- --- --- ---
 0.49| 0.00|-0.23| 0.00|
--- --- --- --- --- --- ---
 0.34| 0.11|-0.20|-0.57|

************************************

At r(S) : 1
Iteration ends at :326
--- --- --- --- --- --- ---
 29.80| 23.14| 12.48| 0.00|
--- --- --- --- --- --- ---
 32.46| 0.00| 10.30| 0.00|
--- --- --- --- --- --- ---
 31.11| 25.77| 16.43| 9.22|

************************************
```

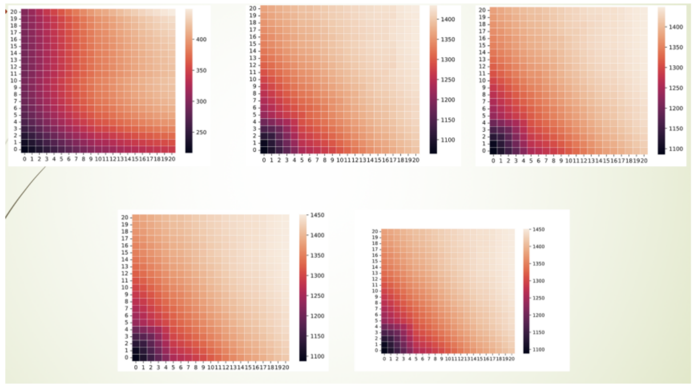
fig: 1 Result of the Experiment



Policy and value for problem 2



Policy for problem 3



value for problem 2

## V. Conclusion

In conclusion, In this lab assignment, we have explored the Markov decision process (MDP) and dynamic programming concepts to solve sequential decision-making problems in stochastic environments. We have used value iteration to find the value function corresponding to the optimal policy for a grid world problem and formulated a continuing finite MDP for the Gbike bicycle rental problem.
We have also implemented policy iteration to resolve the Gbike bicycle rental problem with additional constraints, such as a free shuttle service for one bike and limited parking space at

each location. The results obtained from the program showed the optimal policy and corresponding value function for the problem.
This lab assignment has helped us understand the importance of formulating the problem as an MDP and using dynamic programming algorithms to find the optimal policy in complex decision-making problems. We have also gained experience in implementing policy iteration and value iteration algorithms to solve MDP problems.

## References

[1] Artificial Intelligence: a Modern Approach, Russell and Norvig (Fourth edition) (Chapter 5)
[2] A first course in Artificial Intelligence, Deepak Khemani (Chapter 8)
[3] https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da