# CS367/659 Artificial Intelligence
Autumn 2025-26

General Instructions for Lab
Students enrolled in the course are expected to work on programming problems assigned for laboratories in groups. Groups once formed would remain fixed for the entire duration of the course. Evaluation of lab assignments should primarily be taken as a two-way feedback, both for the students and the instructor.

A group can have a maximum of 4 students from the same section.

Attending laboratory sessions is a must and absence during laboratory sessions without prior intimation to the course instructor (only genuine cases) will attract 0 credit for that particular session.

Feel free to contact me or the TAs in case you have any query or confusion regarding submissions.

Course Instructors:
Pratik Shah, Jignesh Patel
pratik@iiitvadodara.ac.in

Course GIT Repository: https://github.com/pratikiiitv/CS307

Evaluation Policy

Component 1: 60%

A. Attendance and participation
B. Each student group will be assigned a project definition under a teaching assistant (mentor).
C. The groups will meet the respective mentors for project discussions. The students will be assessed during these interactions on weekly basis based on
    i. Quality of explanation and clarity of exposition and discussion
    ii. Quality of the code base
    iii. Resolution of queries/ questions raised by mentor during the session.

Component 2: 25%
Students are required to work in groups on laboratory problems. Each group is expected to submit a total of two reports containing the discussion on the solution to four-four problems each. One such report will be due just before the mid-semester examination and the other just before the end-semester examination.

Report/ Term Paper Template : IEEE Transactions on Systems Man and Cybernetics: Systems in LaTeX format.
https://drive.google.com/file/d/1sdEyE-9DxDNFbuR_n3xJ0u0Og6aw9zKU/view?usp=sharing

It is expected that you will write the report as if you are writing a tutorial to explain the problems and the corresponding solutions.

Component 3: 15%
A project/ lab viva voce will be conducted immediately before/ after the mid/ end-semester examination.

Moral Code of Conduct
Zero tolerance to plagiarism. If the report submitted by any group is found to have copied extracts from any of the other submitted reports the group members will receive no credit for the entire report. In a report if some material is found to have been reproduced from some source without proper citation or reference, the report will attract zero credits.

Week 1 :
Lab Assignment - 1

Learning Objective: To be able to model a given problem in terms of state space search problem and solve the same using BFS/ DFS

Reference:
[1] Artificial Intelligence: a Modern Approach, Russell and Norvig (Fourth edition)
Chapter 1, 2, 3

Problem Statement for In-lab Discussions

A. The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem-formulation from an analytical viewpoint.

Problem Statement for Submission :

B. In the rabbit leap problem, three east-bound rabbits stand in a line blocked by three west-bound rabbits. They are crossing a stream with stones placed in the east west direction in a line. There is one empty stone between them. The rabbits can only move forward one or two steps. They can jump over one rabbit if the need arises, but not more than that. Are they smart enough to cross each other without having to step into the water?

For the above two problems,

1. Model the problem as a state space search problem. How large is the search space?
2. Solve the problem using BFS. The optimal solution is the one with the fewest number of steps. Is the solution that you have acquired an optimal one? The program should print out the solution by listing a sequence of steps needed to reach the goal state from the initial state.
3. Solve the problem using DFS. The program should print out the solution by listing a sequence of steps needed to reach the goal state from the initial state
4. Compare solutions found from BFS and DFS. Comment on solutions. Also compare the time and space complexities of both.

Week 2 :

Lab Assignment - 2

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

Reference:
[1] Artificial Intelligence: a Modern Approach, Russell and Norvig (Fourth edition)
Chapter 2 and 3
[2] A first course in Artificial Intelligence, Deepak Khemani
Chapter 2

Problem Statement for In-lab Discussion:

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.
B. Write a collection of functions imitating the environment for Puzzle-8.
C. Describe what is 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.
E. Generate Puzzle-8 instances with the goal state at depth “d”.
F. Prepare a table indicating the memory and time requirements to solve Puzzle-8 instances (depth “d”) using your graph search agent.

Problem Statement for Submission

Objective:

The goal of this lab is to implement a plagiarism detection system using the A* search algorithm applied to text alignment. The system will identify similar or identical sequences of text between two documents by aligning their sentences or paragraphs and detecting potential plagiarism.

References:
[1] Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. Prentice Hall.
[2] Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Draft chapters available online.

Background:

Plagiarism detection involves comparing two documents to identify shared sequences of words or phrases that might indicate copying. Text alignment is a common approach to this, where one text is aligned with another based on similarity measures. The A* search algorithm, a best-first search, can be applied to efficiently find the optimal alignment between two texts.

Prerequisites:

- Basic understanding of the A* search algorithm.
- Knowledge of string matching algorithms and similarity measures (e.g., Levenshtein distance).
- Python programming skills.

*A Search Overview:**

A* search is a graph traversal algorithm that finds the shortest path from a start node to a goal node. It combines the strengths of Dijkstra's algorithm and greedy best-first search by using a heuristic to estimate the cost of reaching the goal. The A* algorithm uses the following cost function: $f(n)=g(n)+h(n)$

Where:

- $g(n)$ is the cost of the path from the start node to node n.
- $h(n)$ is the estimated cost from node nn to the goal (heuristic function).

Problem Definition:

Given two text documents, the task is to align their sentences and detect plagiarism using A* search. The alignment should minimize the edit distance (or maximize similarity) between corresponding sentences.

Approach:

1. State Representation:
   a. Each state represents a partial alignment between the sentences of the two documents.
   b. A state includes the current position in both documents and the accumulated cost (edit distance).
2. Initial State:
   a. The initial state corresponds to the start of both documents (i.e., first sentence of each document).
3. Goal State:
   a. The goal state is reached when all sentences in both documents are aligned.
4. Transition Function:

    a. Possible transitions include aligning the current sentences, skipping a sentence in one document, or skipping sentences in both.
5. Cost Function (g(n)):
    a. The cost is the edit distance between the aligned sentences so far.
6. Heuristic Function (h(n)):
    a. The heuristic estimates the remaining alignment cost, such as the sum of minimum possible edit distances between remaining sentences.
7. Search Algorithm:
    a. Use A* search to explore the state space and find the optimal alignment with the minimum cost.
8. Lab Instructions:
9. Implement Text Preprocessing:
    a. Tokenize the input documents into sentences.
    b. Normalize the text (e.g., convert to lowercase, remove punctuation).
10. *Define the A* Search Function:*
    a. Implement the A* search algorithm with the state representation, transition function, cost function, and heuristic.
11. Compute Edit Distance:
    a. Implement the Levenshtein distance function to compute the cost of aligning two sentences.
12. Perform Alignment:
    a. Apply the A* search algorithm to align the sentences from the two documents.
13. Detect Plagiarism:
    a. After alignment, identify pairs of sentences with a low edit distance as potential plagiarism.
14. Evaluate the System:
    a. Use test cases to evaluate the accuracy and efficiency of your plagiarism detection system.

Test Cases:

Test Case 1: Identical Documents

    b. Input: Two identical documents.
    c. Expected Output: All sentences should align perfectly, with zero edit distance.

Test Case 2: Slightly Modified Document

    d. Input: One document with minor modifications (e.g., synonyms, word order changes) compared to the other.
    e. Expected Output: Most sentences should align with a low edit distance.

Test Case 3: Completely Different Documents

    f. Input: Two completely different documents.

g. Expected Output: High edit distances for most alignments, indicating no plagiarism.

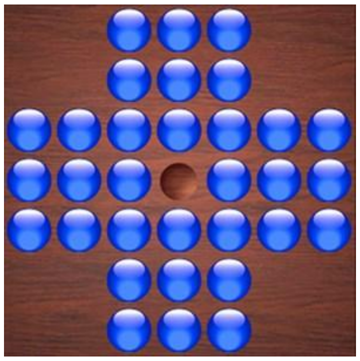
Test Case 4: Partial Overlap

h. Input: Two documents with some overlapping content.
i. Expected Output: The overlapping content should align with a low edit distance, indicating potential plagiarism.

Week 3 :

Lab Assignment - 3

Learning Objective: To understand the use of Heuristic function for reducing the size of the search space. Explore non-classical search algorithms for large problems.



Reference:
[1] Artificial Intelligence: a Modern Approach, Russell and Norvig (Fourth edition)
Chapter 3 and 4
[2] A first course in Artificial Intelligence, Deepak Khemani
Chapter 3, 4

Problem Statement for In-lab Discussion:

A. Read about the game of marble solitaire. Figure shows the initial board configuration. The goal is to reach the board configuration where only one marble is left at the centre. To solve marble solitaire, (1) Implement priority queue based search considering path cost, (2) suggest two different heuristic functions with justification, (3) Implement best first search algorithm, (4) Implement A*, (5) Compare the results of various search algorithms.

Problem Statement for Submission

B. Write a program to randomly generate k-SAT problems. The program must accept values for k, m the number of clauses in the formula, and n the number of variables. Each clause of length k must contain distinct variables or their negation. Instances generated by this algorithm belong to fixed clause length models of SAT and are known as uniform random k-SAT problems.
C. Write programs to solve a set of uniform random 3-SAT problems for different combinations of m and n, and compare their performance. Try the Hill-Climbing, Beam-Search with beam widths 3 and 4, Variable-Neighborhood-Descent with 3 neighborhood functions. Use two different heuristic functions and compare them with respect to *penetrance*.

Week 4:
Lab Assignment 4

Learning Objective:
Non-deterministic Search | Simulated Annealing: For problems with large search spaces, randomized search becomes a meaningful option given partial/ full-information about the domain.

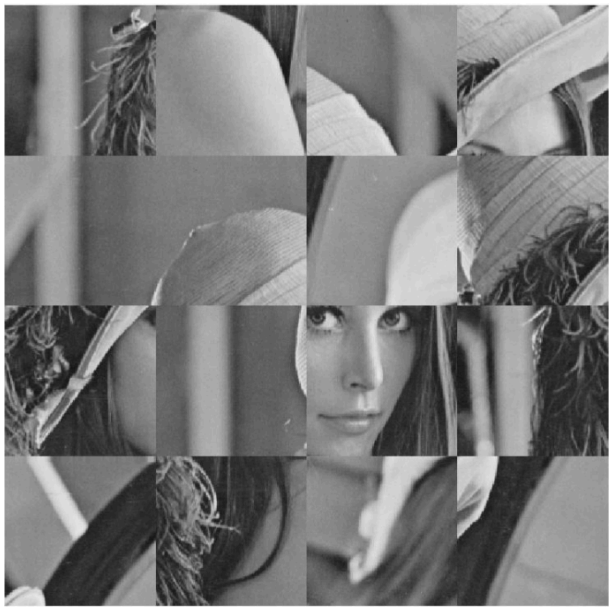
Reference:
Reference:
[1] Simulated annealing from basics to applications, Daniel Delahaye, Supatcha Chaimatanan, Marcel Mongeau
[2] A first course in Artificial Intelligence, Deepak Khemani (Chapter 4)
[3] Artificial Intelligence: a Modern Approach, Russell and Norvig (Fourth edition)
Chapter 3 and 4

Problem Statement for In-lab Discussions:

The Traveling 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 labeled with the cost of traveling between cities, find a cycle containing each city exactly once, such that the total cost of the tour is as low as possible.

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 traveling between two locations is proportional to the distance between them.

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.)



Problem Statement for Submission:

Ever played jigsaw puzzles in childhood? Time to make an agent learn to solve the puzzle.

Some rough, confusing code snippets are posted on the code repository of the course along with the scrambled_lena.mat

Formulate the problem as a state space search problem. Try to solve the same using simulated annealing.  Have fun!

Challenge Problem : (you may be able to earn some extra credits for meaningfully attempting this challenge)

I prompted ChatGPT with the following: “Generating melody using simulated annealing” and I was surprised to see a very interesting response. Challenge for you is to generate a melody in North Indian Classical Raag Bhairav using either simulated annealing or the genetic algorithm.

Assessment: Group which is able to generate melody which follows the grammar of Raag Bhairav and contains some fundamental phrases of the Raag will earn higher credits.

<u>Week 5:</u>
Lab Assignment 5

Learning Objective:
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.

Reference:
1. https://www.bnlearn.com/
2. http://gauss.inf.um.es/umur/xjurponencias/talleres/J3.pdf

Problem Statement for In-lab Discussions:
A table containing grades earned by students in respective courses is made available to you in (codes folder) 2020_bn_nb_data.txt.
1. Consider grades earned in each of the courses as random variables and learn the dependencies between courses.
2. Using the data, learn the CPTs for each course node.
3. What grade will a student get in PH100 if he earns DD in EC100, CC in IT101 and CD in MA101.
4. 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.
5. Repeat 4, considering that the grades earned in different courses may be dependent.

Problem Statement for Submission:

The goal of this lab assignment is to analyze financial time series data using Gaussian Hidden Markov Models (HMMs). The HMM is employed to identify hidden market regimes (e.g., periods of high and low volatility) and model the dynamics of financial returns. The task involves collecting real-world financial data, modeling it with HMM, and interpreting the results to draw insights about market conditions.

Problem Statement:

In this assignment, you are required to model the hidden regimes of a financial time series, such as stock returns or index performance, using a Gaussian Hidden Markov Model. Financial markets often exhibit distinct periods of different volatility or risk levels, and these periods may not be directly observable. Gaussian HMM allows you to infer these hidden states (e.g., bull and bear markets) from observable data, such as stock price returns.

[Use Yahoo Finance API to download historical stock price data.]

## Part 1: Data Collection and Preprocessing

1. Data Acquisition: Download historical financial data (e.g., stock prices or indices) from a reliable source like Yahoo Finance. The data should cover a sufficiently large time frame (e.g., 10 years or more) to capture long-term trends and market regimes. You may choose a specific company (e.g., Apple, Tesla) or an index (e.g., S&P 500).
2. Preprocessing: Extract the relevant features for modeling, such as daily adjusted closing prices. Calculate the daily returns from the adjusted closing prices. Handle any missing or erroneous data points by cleaning the dataset to ensure quality input.

## Part 2: Gaussian Hidden Markov Model

1. Fitting the Model: Use the Gaussian Hidden Markov Model to model the financial returns data. Select an appropriate number of hidden states (e.g., 2 or more) to capture different market conditions (e.g., high and low volatility regimes). Fit the HMM to the returns data and predict the hidden states that best describe the observed data.
2. Parameter Analysis: Analyze the mean and variance of each hidden state to characterize the market regimes. Interpret these hidden states in the context of financial markets (e.g., identifying bear and bull markets or high/low-risk periods).

## Part 3: Interpretation and Inference

1. Inferred Hidden States: Decode the hidden states predicted by the HMM and identify the time periods corresponding to each hidden state. Visualize the inferred hidden states over time and analyze how the market transitions between different regimes.
2. Transition Matrix: Investigate the transition matrix of the HMM, which defines the probabilities of switching between different hidden states. Use the transition probabilities to understand the likelihood of the market shifting from one regime to another (e.g., from low volatility to high volatility).

## Part 4: Evaluation and Visualization

1. Visualization: Plot the stock prices or returns and color-code the time periods based on the inferred hidden states. Create visualizations that clearly differentiate between the different market regimes identified by the HMM.
2. Model Evaluation: Discuss how well the HMM captures the underlying market dynamics. Evaluate whether the hidden states provide meaningful insights into the stock's volatility patterns. Analyze how changes in the number of hidden states impact the interpretation of the results.

## Part 5: Conclusions and Insights

1. Market Regime Interpretation: Summarize the hidden market regimes identified by the model and describe their characteristics (e.g., high volatility versus low volatility periods). Discuss how these hidden states can inform financial decision-making, such as risk management or portfolio adjustment.
2. Future State Prediction: Predict the likely future state of the market based on the most recent data. Discuss how the transition matrix can be used for short-term market regime forecasting and its potential impact on financial strategies.

## Bonus Task (Optional):

- Extend the Gaussian HMM to include more than two hidden states to capture more complex market behavior, such as extreme volatility periods or stable growth phases.
- Compare the results of the HMM model on different financial instruments (e.g., comparing stocks of different companies or indices) to understand how market dynamics vary across assets.

### Deliverables:

1. Analysis Report: Discuss the application of Gaussian Hidden Markov Models to financial time series data. Provide a detailed explanation of the steps taken to preprocess the data, fit the model, and interpret the results. Include visualizations and insights from the analysis, highlighting the inferred hidden states and market regimes.
2. Code Implementation: Submit GIT repo with the Python implementation used to collect data, fit the Gaussian HMM model, and perform the analysis.

Lab Assignment for FUN

Objective: (unsupervised learning)
Smoothing filter as a clustering mechanism, Hierarchical Agglomerative Clustering

References:

1. Diffusion and Confusions in Signal and Image Processing
   http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C857BB546B94BFA72996815EE6DAB6C4?doi=10.1.1.228.3667&rep=rep1&type=pdf
2. Hierarchical Agglomerative Clustering
   Chapter 16 and 17, Introduction to Information Retrieval, Manning, Raghvan and Schutze

Problems:

1. Given the noisy image test_noisy.jpg (available in the codes folder). Use adaptive smoothing (Ref. Diffusions and Confusions in Signal and Image Processing) process to remove the noise. Explain the selected values of the parameters in the algorithm.
2. Use HAC with Euclidean/ Manhattan distance as a measure (Single link, complete link, Ward's distance, Group average, Centroid, Clusteroid) cluster the states of India based on the feature vector comprising of the following parameters (for one of the financial year values available in the data-set)
   a. Percentage of schools with electricity
   b. Percentage of schools with girls toilet
   c. Percentage of schools with drinking water
   d. Percentage of schools with boys toilet

For second problem you should get the data from the following:
https://data.gov.in/

As an example, a simple visualization based on percentage of schools with electricity is available at https://data.gov.in/major-indicator/percentage-schools-electricity

This lab is more fun than serious derivations and implementations. I hope you will enjoy working on the problems.

Week 6:
Lab Assignment 6

Learning Objective: To understand the working of Hopfield network and use it for solving some interesting combinatorial problems

References: David McKay, Information Theory, Inference and Learning Algorithms

Problems for in-lab discussions:

1. Implement 10x10 associative memory (binary) using Hopfield network.
2. Find out the capacity of the hopfield network in terms of storage of distinct patterns.

Problems for submission:

3. What is the error correcting capability of your Hopfield network?
4. Setup the energy function for the Eight-rook problem and solve the same using Hopfield network. Give reasons for choosing specific weights for the network.
5. Solve a TSP (traveling salesman problem) of 10 cities with a Hopfield network. How many weights do you need for the network?

Week 7:
Lab Assignment 7

Learning Objective: Basics of data structure needed for state-space search tasks and use of random numbers required for MDP and RL, Understanding Exploitation - Exploration in simple n-arm bandit reinforcement learning task, epsilon-greedy algorithm

Title: Matchbox Educable Naughts and Crosses Engine
Reference: [http://people.csail.mit.edu/brooks/idocs/matchbox.pdf](http://people.csail.mit.edu/brooks/idocs/matchbox.pdf)

Problems for submission:

(1) Read the reference on MENACE by Michie and check for its implementations. Pick the one that you like the most and go through the code carefully. Highlight the parts that you feel are crucial. If possible, try to code the MENACE in any programming language of your liking.

Reference: Reinforcement Learning: an introduction by R Sutton and A Barto (Second Edition) (Chapter 1-2)

(2) Consider a binary bandit with two rewards {1-success, 0-failure}. The bandit returns 1 or 0 for the action that you select, i.e. 1 or 2. The rewards are stochastic (but stationary). Use an epsilon-greedy algorithm discussed in class and decide upon the action to take for maximizing the expected reward. There are two binary bandits named binaryBanditA.m and binaryBanditB.m are waiting for you.

(3) Develop a 10-armed bandit in which all ten mean-rewards start out equal and then take independent random walks (by adding a normally distributed increment with mean zero and standard deviation 0.01 to all mean-rewards on each time step).
{function [value] = bandit_nonstat(action)}

(4) The 10-armed bandit that you developed (bandit_nonstat) is difficult to crack with a standard epsilon-greedy algorithm since the rewards are non-stationary. We did discuss how to track non-stationary rewards in class. Write a modified epsilon-greedy agent and show whether it is able to latch onto correct actions or not. (Try at least 10000 time steps before commenting on results)

Week 8:
Lab Assignment 8

Learning Objective: Understand the process of sequential decision making (stochastic environment) and the connection with reinforcement learning

Title: Markov Decision Process and Dynamic Programming
Reference:
[1] Artificial Intelligence a Modern Approach, Russell and Norvig (third edition)
Chapter 16, 17, 21
[2] Reinforcement Learning, Sutton and Barto (second edition)
Chapter 3, 4

Problem for In-lab Discussion:

(1) Suppose that an agent is situated in the 4x3 environment as shown in Figure 1. Beginning in the start state, it must choose an action at each time step. The interaction with the environment terminates when the agent reaches one of the goal states, marked +1 or -1. We assume that the environment is fully observable, so that the agent always knows where it is. You may decide to take the following four actions in every state: Up, Down, Left and Right. However, the environment is stochastic, that means the action that you take may not lead you to the desired state. Each action achieves the intended effect with probability 0.8, but the rest of the time, the action moves the agent at right angles to the intended direction with equal probabilities.

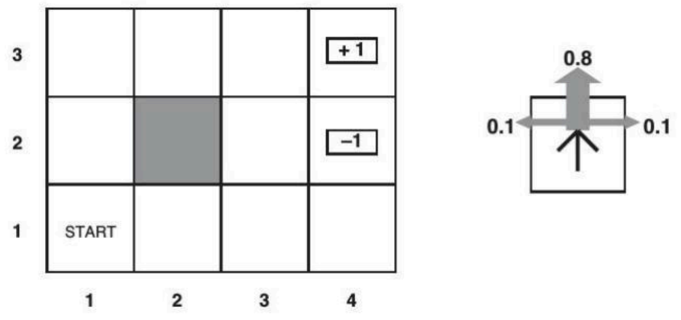


Figure 1: $4 \times 3$ grid world with uncertain state change.

Furthermore, if the agent bumps into a wall, it stays in the same square. The immediate reward for moving to any state (s) except for the terminal states $S^+$ is r(s)= -0.04. And the reward for moving to terminal states is +1 and -1 respectively. Find the value function corresponding to the optimal policy using value iteration.

Find the value functions corresponding optimal policy for the following:

(a) r(s)=-2
(b) r(s)=0.1

(c) r(s)=0.02
(d) r(s)=1

Problems for Submission:

(2) [Gbike bicycle rental] You are managing two locations for Gbike. Each day, a number of customers arrive at each location to rent bicycles. If you have a bike available, you rent it out and earn INR 10 from Gbike. If you are out of bikes at that location, then the business is lost. Bikes become available for renting the day after they are returned. To help ensure that bicycles are available where they are needed, you can move them between the two locations overnight, at a cost of INR 2 per bike moved.

    Assumptions: Assume that the number of bikes requested and returned at each location are Poisson random variables. Expected numbers of rental requests are 3 and 4 and returns are 3 and 2 at the first and second locations respectively. No more than 20 bikes can be parked at either of the locations. You may move a maximum of 5 bikes from one location to the other in one night. Consider the discount rate to be 0.9.

    Formulate the continuing finite MDP, where time steps are days, the state is the number of bikes at each location at the end of the day, and the actions are the net number of bikes moved between the two locations overnight.

    Download and extract files from gbike.zip. Try to compare your formulation with the code. Before proceeding further, ensure that you understand the policy iteration clearly.

(3) Write a program for policy iteration and resolve the Gbike bicycle rental problem with the following changes. One of your employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one bike to the second location for free. Each additional bike still costs INR 2, as do all bikes moved in the other direction. In addition, you have limited parking space at each location. If more than 10 bikes are kept overnight at a location (after any moving of cars), then an additional cost of INR 4 must be incurred to use a second parking lot (independent of how many cars are kept there).

Note: Problem (2) and (3) are directly taken from the book by Sutton and Barto. The problem if attempted independently (without referring to the solutions available) is a good exercise in understanding the implementation issues and finite MDP. Changing the Jack's car rental to Gbike bicycle rental is just to put things in context. The instructor has no intention of earning credit for posing the problem. Credit goes to Sutton and Barto.