Artificial Intelligence Lab Manual

Group Name: F A N U M
Gaganpreet Singh[202211023], Pragya Pranati[202211068], Suryansh Singh Raghuvansh[202211093]

***Abstract*—This lab report presents four experiments exploring various AI techniques in problem-solving scenarios. The first experiment, Rabbit Leap Problem, investigates whether a group of rabbits can cross each other on a set of stones without stepping into the water, using a forward-only movement and leap strategy. The second experiment, Plagiarism Checker, utilizes the A* search algorithm to detect plagiarism by aligning text sequences between two documents. The third experiment, K-SAT Problem, focuses on the generation and solving of uniform random k-SAT problems, comparing the performance of Hill-Climbing, Beam Search, and Variable Neighborhood Descent algorithms. Finally, the fourth experiment, Puzzle Solving, aims to create an AI agent capable of solving a complex puzzle by applying state space search methods. Through these experiments, this report demonstrates the effectiveness of AI techniques in solving both combinatorial and real-world problems.**

Keywords: AI algorithms, Beam Search, Variable Neighborhood Descent, state space search, puzzle solving, heuristic functions

The GitHub Repository to the the CS367 course lab(s) submissions can be accessed here

I. Introduction

IN Artificial Intelligence (AI) plays a crucial role in solving complex problems across various domains, from natural language processing to combinatorial optimization. This lab focuses on applying AI algorithms to four different types of problems, each requiring a unique approach for effective problem-solving. In Experiment 1, the challenge of orchestrating movement patterns for rabbits demonstrates AI's role in pathfinding and optimization. Experiment 2 addresses plagiarism detection, a critical issue in academic and professional environments, using heuristic search for efficient alignment of text sequences. Experiment 3 delves into solving k-SAT problems, a foundational problem in computational theory, by employing heuristic-based algorithms to explore solution spaces. Lastly, Experiment 4 explores puzzle-solving through AI agents, where the goal is to construct a search-based solution to navigate the puzzle state space efficiently. Collectively, these experiments highlight the diverse applications of AI and showcase how intelligent algorithms can be leveraged for both theoretical and practical problem-solving.

Experiment 1: Rabbit Leap Problem

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?

Experiment 2: Plagiarism Checker

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.

Experiment 3: n-SAT Problems

(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,

- Implement priority queue based search considering path cost
- suggest two different heuristic functions with justification
- Implement best first search algorithm
- Implement A*
- Compare the results of various search algorithms

(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.

Experiment 4: Financial Time Series Analysis Using Gaussian Hidden Markov Models

The problem involves analyzing financial time series data from Yahoo Finance API, such as stock returns, using **Gaussian Hidden Markov Model (HMM)** to identify hidden market regimes. The task is to model periods of varying market volatility (e.g., bull and bear markets) by inferring hidden states from observable data, such as daily stock returns, and interpreting the transitions between these regimes to gain insights into market behavior.

The process involves collecting and pre-processing historical stock data, calculating daily returns to capture market dynamics. A Gaussian Hidden Markov Model (HMM) is applied to infer hidden market states, and the results are visualized to highlight regimes and transition probabilities between market conditions like bull and bear markets.

II. Experiment 1: Rabbit Leap Problem

A. Problem Statement Description

- There are three east-bound rabbits and three west-bound rabbits standing in a line on stones.
- The stones are placed in the east-west direction in a straight line, with one empty stone between the two groups of rabbits.
- Rabbits can move forward either one or two steps.
- A rabbit can jump over one other rabbit if needed, but it cannot jump over more than one rabbit at a time.
- The objective is to determine if the rabbits can cross each other without stepping into the water.

B. Solutions

- The problem is modeled as a sliding puzzle where the rabbits and the empty stone are represented as a list:
  - 'E' represents an east-bound rabbit.
  - 'W' represents a west-bound rabbit.
  - 'O' represents an empty stone.
- The goal is to simulate the valid movements of the rabbits until the east-bound rabbits cross to the west side and the west-bound rabbits cross to the east side.
- The initial state of the system is represented as:

$$['E','E','E','O','W','W','W']$$

- The final goal state is:

$$['W','W','W','O','E','E','E']$$

- A breadth-first search (BFS) algorithm is used to explore all possible configurations of the rabbits.
- The algorithm iteratively examines the current configuration and generates new configurations by moving the rabbits according to the following rules:
  - A rabbit can move one step to the adjacent empty stone.
  - A rabbit can jump over one adjacent rabbit to land on the empty stone if it is exactly two steps away.
- If the configuration matches the goal state, the algorithm returns the sequence of moves that achieves the crossing.
- If no valid solution exists, the algorithm terminates and returns 'None'.

The program code can be accessed through link

III. Experiment 3:n-SAT Problem

A. Problem Statements

1) 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,
   - Implement priority queue based search considering path cost
   - suggest two different heuristic functions with justification
   - Implement best first search algorithm
   - Implement A*
   - Compare the results of various search algorithms
2) 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.
3) 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 peneterance.

B. Solutions

1) The pseudo code for the following problem can be accessed through link and the flowchart of the following is:

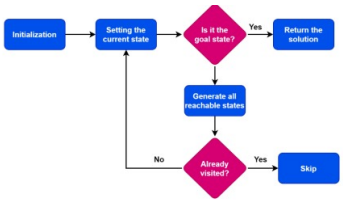

Figure 1: Flowchart

1.1 Priority Queue based Search Cost function => Number of Pegs remaining on the board Frontier = [N1, Cost1, N2, Cost2 .... Nk, CostK]

1.2 Heuristic functions

i) Number of Pegs Remaining on the board
- Is directly related to the number of playable moves on the board
- Having lower Peg remaining implies we are closer to the goal state
- **T**he number of pegs heuristic is admissible because it never overestimates the cost to reach the goal.

ii) Distance to center
- Returns the average of Manhattan distance of each peg from center
- A larger number indicates more pegs are remaining away from center
- A smaller number indicates fewer pegs remaining closer to the center
- This heuristic suggests how far the current state is from the goal state

1.3 Implementing Best First Search and A* Search

- Define Initial state, Goal state, Cost function, Heuristic function
- Cost function => Path cost + Heuristic
- Path cost += 1 for every move
- Heuristic function chosen => Pegs remaining

Both BeFS and A* Search uses a priority queue to select the state with minimum cost from the frontier.

1.5 Comparing results between different Search

- DFS > A* Search > BeFS
- DFS $\geq$ 2420 nodes
- A* $\geq$ 10831 nodes
- BeFS $\geq$ 3, 41, 491 nodes

Both BFS and A* Search uses a priority queue to select the state with minimum cost from the frontier.

2) k-sat is a NP-complete problem, meaning there is known polynomial-time algorithm that solves all instances of the problem efficiently. It is concerned with determining whether we can satisfy a given boolean condition by assigning truth values. For a problem we already have the Boolean formula made up of 'k' literals in each clause, having 'm' total clauses having 'n' variables
Example of a 2-SAT problem : (A v B) (-A v -B)

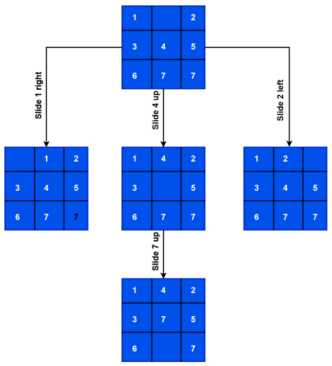

Figure 2: Algorithm

The pseudo code for the following problem can be accessed through link

3)

- **Beam search:** Beam search is an optimization of best-first search that reduces its memory requirements. Beam search uses breadth-first search to build its search tree.At each level of the tree, it generates all successors of the states at the current level, sorting them in increasing order of heuristic cost.
- **Hill climbing:** Evaluate the initial state. If it is a goal state then stop and return success. Otherwise, make the initial state as the current state. Loop until the solution state is found or there are no new operators present which can be applied to the current state
- **Variable-Neighborhood-Descent algorithm:** Start with an initial solution. Check if the current solution meets the desired criteria or solves the problem. If yes, return the solution. Otherwise, explore different neighborhoods around the current solution.

The program code can be accessed through link

C. Comparison

Penetrance measures the ability of a heuristic function to find satisfying solutions (solutions where all clauses are true). It's calculated as the average number of satisfying solutions the heuristic finds across multiple problem instances with the same parameters (m and n).

Problems	Heuristic functions	Penetrance
Problem 1	Hill Climbing	6/13
	Beam Search (3)	6/7
Problem 2	Hill Climbing	2/13
	Beam Search (3)	2/7
Problem 3	Hill Climbing	1/7
	Beam Search (3)	8/13
Problem 4	Hill Climbing	1/7
	Beam Search (3)	13/13
Problem 5	Hill Climbing	2/13
	Beam Search (3)	2/7

IV. Experiment 4: Financial Time Series Analysis Using Gaussian Hidden Markov Models

A. Problem Statement Description

The task is to analyze financial time series data, particularly stock returns from the Yahoo Finance API, using a Gaussian Hidden Markov Model (HMM). The goal is to identify hidden market regimes, such as bull and bear markets, by inferring hidden states from the observable data. The model captures periods of varying market volatility and interprets the transitions between market regimes to provide insights into market behavior.

B. Solutions

1) **Data Collection and Pre-processing**
The financial time series data is collected from the Yahoo Finance API. The key steps for pre-processing are:

- Retrieve historical stock price data, focusing on daily closing prices.
- Compute the **daily returns**, which capture day-to-day changes in stock price using the formula:

$$\text{Return}(t) = \frac{P(t) - P(t-1)}{P(t-1)}$$

where $P(t)$ is the stock price at day t.

2) **Gaussian Hidden Markov Model**
The Gaussian Hidden Markov Model (HMM) is applied to model varying market volatility:

- **Hidden States:** These represent unobserved market regimes, such as bull and bear markets.

- **Observations:** The daily returns are the observable data used to infer the hidden states.
- **Emission Probabilities:** Each hidden state is modeled by a Gaussian distribution, representing the volatility of returns.
- **Transition Probabilities:** These define the likelihood of transitioning between hidden states (e.g., from a bull market to a bear market).

3) **Model Training**
The Gaussian HMM is trained using the Expectation-Maximization (EM) algorithm:
   a) **E-step:** Estimate the probabilities of the hidden states (bull or bear) given the observed returns.
   b) **M-step:** Update the model parameters, including transition probabilities and emission distributions, to maximize the likelihood of the observed data.

4) **Hidden State Inference**
Once the model is trained, the hidden states for each day are inferred using the **Viterbi algorithm**. This provides the most likely sequence of hidden states, revealing the underlying market regimes over time.

5) **Visualization of Results**
The results are visualized to highlight the hidden market regimes and the transitions between them:
   - **Market Regimes:** A time series plot of daily stock returns, with different colors representing the inferred hidden states (e.g., bull markets in green and bear markets in red).
   - **Transition Matrix:** A plot showing the transition probabilities between hidden states, offering insights into how likely it is for the market to switch between regimes.

6) **Evaluation**
The Gaussian Hidden Markov Model successfully identifies periods of high and low volatility in the stock returns, corresponding to bull and bear markets. Transition probabilities provide insights into the likelihood of shifting between different market regimes.

7) **Python Implementation**
The model is implemented using Python. Historical stock data is collected using the Yahoo Finance API, and the Gaussian Hidden Markov Model is applied to infer hidden market regimes. The model outputs inferred states along with visualizations of market regimes and transition probabilities.

V. References

1) Artificial Intelligence A Modern Approach By Stuart Russell , Peter Norvig[Link].
2) Neural Networks - A Systematic introduction by Raul Rojas [Link].
3) A First Course in Artificial Intelligence, Khemani D., Tata McGraw Hill, 2014.