Experiment 2
Heuristic Function for reducing search space

Jagruti Piprade
Computer Science Engineering
202151067@iiitvadodara.ac.in

Makwana Harsh Maheshkumar
Computer Science Engineering
202151082@iiitvadodara.ac.in

Khushi Saxena
Computer Science Engineering
202151078@iiitvadodara.ac.in

Nitin Gautam
Computer Science Engineering
202151101@iiitvadodara.ac.in

Google collab notebook for this assignment can be found here.

***Abstract*—This report looks at two problems: Marble Solitaire and Random k-SAT Problems. We experiment with various methods to solve these problems and make clever guesses to improve our search. By comparing different ways of solving them, we figure out which ones are most effective. This helps us understand how to solve difficult problems using easier techniques.**

I. 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.

II. Introduction

In the domain of problem-solving, heuristic functions and non-classical search algorithms play crucial roles in navigating complexity. Two fascinating aspects of computer reasoning are discussed in this experiment:

- Marble Solitaire
- Random k-SAT Problems

Through trying things out and comparing them, we will try to show that these techniques can make problem-solving easier and help improve how we solve problems using computers.

III. Problem I: Marble Solitaire

Problem Statement: 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.

Marble Solitaire

Marble Solitaire is a single player board game played on a board with holes, initially filled with marbles except for one hole in the center. The goal is to reach a board configuration where only one marble remains. To tackle this problem, we employ priority queue-based search algorithms, Best-First Search, and A* Search.

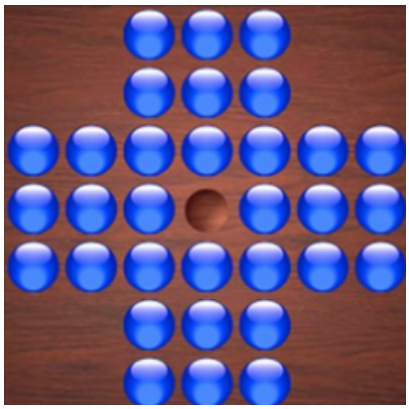

Fig. 1. Marble solitaire

Priority queue-based search: This method uses a priority queue to order the states to explore based on the total cost of the path from the starting point. States with lower path costs are given priority and explored before those with higher costs.

Best-First Search: This algorithm prioritizes states based on a heuristic function that estimates how much more effort is needed to reach the goal state from each state.

A Search:* In A* search, we take into account both the path cost (how far we've come) and the heuristic function (how far we estimate it is to the goal). We prioritize exploring states that have the lowest total of these two factors combined.

In our case, we're using two types of heuristic functions: Manhattan Distance and Exponential Distance. These are good choices for problems where you can only move in certain

directions, like up, down, left, and right, which is the case for our grid-based problem. The Manhattan Distance calculates the distance by moving only in horizontal and vertical directions, while the Exponential Distance might take into account diagonal moves or other non-linear paths.

Search Algorithm	Heuristic Function	Time Taken (seconds)
Priority Queue Search	None	0.19
Best First Search	Manhattan Distance	0.16
Best First Search	Exponential Distance	0.13
A* search	Manhattan Distance	0.12
A* search	Exponential Distance	0.15

Fig. 2. Time Comparison

IV. Problem II: Random k-SAT Problems

Problem Statement: 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.

Random K-SAT Problem

A Random k-SAT problem is a puzzle where you have statements with variables and their negations, and you need to figure out if there's a way to make all the statements true. In the implemented code,
k represents the number of variables that appear together in each statement.
m is the total number of these mini-combinations you need to solve to unlock everything.
n is the total number of individual dials (variables) used across all the mini-combinations.

V. Problem III: 3 Sat Problem

Problem Statement: 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.

Hill Climbing

Hill Climbing is like trying to climb a hill to reach the top. You start from a random spot and keep going up, step by step. If each step takes you higher, you stick with it, but if not, you try a different direction. This goes on until you can't climb any higher, reaching a point called a local maximum. In computer terms, Hill Climbing starts with a solution and improves it bit by bit based on how good it is according to a rule. This continues until no more improvements can be made.

Beam Search

You start with a set of possible paths (the beam) and explore them simultaneously. At each step, you keep only the most promising paths based on certain criteria. This process continues until you find a satisfactory solution or exhaust your options.

Both Hill Climbing and Beam Search have their own advantages and disadvantages, and which one you choose depends on what you need for the particular problem you're trying to solve.

VI. Conclusion

In the Marble Solitaire case, we saw how using various search algorithms like priority queue-based search, Best-First Search, and A* showed how different heuristic functions affect their performance. Similarly, when dealing with Uniform Random k-SAT problems, we compared different heuristic functions and search strategies. By comparing the performance of Hill-Climbing, Beam-Search, and Variable-Neighborhood-Descent algorithms, we learned that each has its own strengths and weaknesses depending on the problem at hand. Overall, this study not only enhanced our understanding of heuristic functions and non-traditional search algorithms but also showed how they can be applied practically in the field of AI.

References

[1] https://medium.com/@aksblog/peg-of-solitaire-game-busted-with-ai-c5f73466f8c3
[2] https://www.stat.berkeley.edu/ mossel/teach/206af06/scribes/sep14.pdf