Course Code: CSE440 | Section: 1 | Group No: 7
Sheikh Mushrure Zucky (ID: 1821178642),
Md. Tajul Islam (ID: 2031210642),
Md. Taibur Rahaman (ID: 1931424642),
Md. Simanto Haider (ID: 2021211042)

Development of a Poker Game Simulator Using Expectiminimax for Strategic Decision-Making in Uncertain Events

Md. Simanto Haider
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
simanto.haider@northsouth.edu

Md Taibur Rahaman
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
taibur.rahman@northsouth.edu

Sheikh Mushrure Zucky
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
sheikh.zucky@northsouth.edu

Md. Tajul Islam
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
tajul.islam06@northsouth.edu

Mohammad Shifat-E-Rabbi
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
rabbi.mohammad@northsouth.edu

***Abstract*—This paper presents the development of a Poker game simulator where artificial intelligence (AI) is used to make decisions under uncertain conditions. Poker is a game where players do not have full information, making it difficult to choose the best move. We used the Expectiminimax algorithm, which is suitable for games involving both chance and strategy, to create a smart decision-making system. The simulator includes game rules, card handling, player turns, and the ability to make decisions based on probabilities. We also analyzed the performance of the algorithm and discussed how it behaves in different game scenarios.**

Keywords—Poker Game, Expectiminimax, AI Decision Making, Uncertainty, Game Simulation

I. Introduction

In games like Poker, players must make decisions with incomplete information. Unlike chess, where all pieces are visible, Poker hides the opponent's cards. This makes the game uncertain and probabilistic. AI systems like Expectiminimax help simulate smart decision-making by combining logic, chance, and opponent modeling.

This paper explains how we designed a Poker game simulator and applied the Expectiminimax algorithm to help a bot make good moves. We also evaluate how well this bot performs in different game scenarios.

II. Environment Design

A. Game Components

We used a standard 52-card deck. Each player has chips and can choose to fold, call, or raise. The game goes in turns. We wrote code that simulates how cards are shuffled, dealt, and how hands are evaluated.

B. Players

There are two types of players:

Human Player: Takes input from the keyboard.

Bot Player: Uses the Expectiminimax algorithm to decide.

C. Game Rules

We implemented basic Texas Hold'em rules:

Two hole cards per player

Five community cards

Betting rounds (pre-flop, flop, turn, river)

III. The Expectiminimax Algorithm

Expectiminimax is an extension of the minimax algorithm. It includes chance nodes in addition to max and min nodes. These are:

- Max nodes: The bot tries to maximize its chances

- Min nodes: Assumes the opponent tries to minimize bot's advantage
- Chance nodes: Random events (like drawing a card)

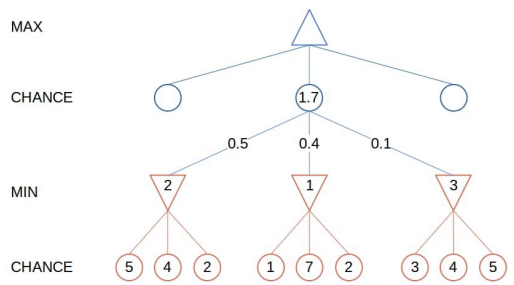

Fig: Expectiminimax tree

$$expectimax(u) = \begin{cases} utility(u, \text{MAX}), & \text{if } terminal(u) = true \\ \max_{a \in moves(u)} expectimax(result(u,a)), & \text{if } turn(u) = \text{MAX} \\ \min_{a \in moves(u)} expectimax(result(u,a)), & \text{if } turn(u) = \text{MIN} \\ \sum_{r \in moves(u)} P(r) \cdot expectimax(result(u,r)), & \text{if } turn(u) = \text{CHANCE} \end{cases}$$

r is an outcome of the random event at chance nodes, and P(r) is its probability.

The algorithm builds a tree of possible game states. It then evaluates which move gives the best expected value by combining strategy and probability.

IV. Evaluation Function

We created a scoring system that gives each game state a numeric value based on how likely it is to win. For example:

Strong hand = high score
Weak hand = low score
Folding = score of 0

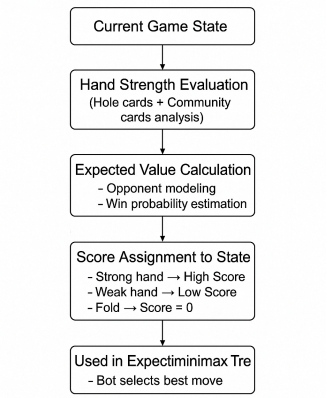

Fig: State Evaluation Logic (Poker Bot AI)

This helped the AI make more accurate choices instead of just guessing.

V. Conclusion

This project successfully used the Expectiminimax algorithm to create a basic Poker bot that can make smart decisions under uncertainty. It shows how game AI can combine probability and strategy. In future, we can improve it with learning-based evaluation and better pruning techniques to make it faster.

Acknowledgment

We thank our course instructor Mohammad Shifat-E-Rabbi, Assistant Professor, ECE, North South University, for guiding us throughout this project.

References

[1] D. Michie, "Advances in Programming and Non-Numerical Computation," 1966.

[2] M. Winands, M. Schadd, and J. Uiterwijk, "ChanceProbCut: Forward Pruning in Chance Nodes," Proc. IEEE CIG, pp. 1–8, 2009.

[3] D. Billings, D. Papp, J. Schaeffer, and D. Szafron, "Opponent Modeling in Poker," Proc. AAAI, 1998.

[4] J. Palomaäki et al., "Poker as a Domain of Expertise," Journal of Expertise, vol. 3, no. 2, Mar. 2020.

[5] Python Simulator with Expectiminimax, "csh130, Texas-Hold-em-poker-game," GitHub repository. [Online]. Available: https://github.com/csh130/Texas-Hold-em-poker-game

[6] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," 3rd ed., Pearson, 2010.