Poker Game simulator with Expectiminimax Algorithm for strategic Decision making

Yeamin Mahmud, Ashfaq Hassan Shifat, Dewan Fatin hasnat

***Abstract*—Poker, a game of incomplete information, is known for its complex strategic nature. The aim of this project was to create a poker game simulator capable of intelligent decision-making in uncertain environments. For this purpose, we employed the Expectiminimax algorithm—a variation of the Minimax algorithm that incorporates chance nodes—to simulate intelligent decision-making. Expectiminimax is particularly suited to games like poker, where chance and player decisions both influence outcomes.**

A. *Objectives*

The primary goals of this poker simulator were:

1. To develop a functional poker game environment where multiple agents can play.
2. To implement an Expectiminimax-based AI to make strategic decisions under uncertainty.
3. To evaluate the performance of the AI in terms of decision accuracy, effectiveness, and ability to compete with both deterministic and random opponents.

B. *Game enviroment*

The simulator was developed to support **Texas Hold'em Poker**, which is one of the most popular variations of poker. Also, the game has been developed by pygame environment. The environment includes:

- **Players**: Each player has a certain amount of chips and can make decisions such as fold, check, call, bet, or raise.
- **Dealer and Card Deck**: A standard 52-card deck is used, and cards are dealt to players according to Texas Hold'em rules.
- **Betting Rounds**: The game proceeds in four rounds: pre-flop, flop, turn, and river.

Action and rewards: Each action (fold, call, raise) influences the player's odds and expected payoff. Players aim to maximize their rewards by winning pots or minimizing losses.

C. ***Expectiminimax Algorithm****:* The Expectiminimax algorithm is an extension of the Minimax algorithm, commonly used in adversarial games, with the addition of chance nodes. The algorithm operates as follows:

1. **Minimax Nodes**: Represent choices made by opponents.
2. **Max Nodes**: Represent choices made by the AI agent.
3. **Chance Nodes**: Represent probabilistic events, such as drawing a card from the deck.

In each decision point:

- **Max nodes** simulate the AI's attempts to maximize expected rewards.
- **Min nodes** simulate the opponents' efforts to minimize the AI's reward.
- **Chance nodes** evaluate the possible outcomes and associated probabilities for chance events like card draws.

Heuristic Evaluation Function: An essential aspect of the Expectiminimax algorithm is its evaluation function. For poker, the evaluation function considers:

- **Hand strength**: Likelihood of winning based on the hand.
- **Pot odds**: The ratio of the current bet to the pot size.
- **Expected value**: Probabilistic outcomes of future game stages based on possible card draws and opponent behavior.

D. *Implementation details:*

Simulator Framework

The poker game simulator was implemented in Python using object-oriented principles to represent players, hands, and game mechanics. A Monte Carlo simulation was also integrated to estimate probabilities in complex scenarios, particularly during the evaluation of chance nodes.

Expectiminimax with Depth-Limiting

Given the complexity and branching factor in poker, we used a depth-limited Expectiminimax approach to manage computational feasibility. Depth limits were set according to the number of players, game stage, and remaining deck size.

Opponent Modeling

The simulator includes different opponent types:

- **Aggressive**: Bets frequently and raises high.
- **Conservative**: Bets and raises sparingly.
- **Random**: Chooses actions at random to simulate novice players.

E. Testing & Evaluation

The Expectiminimax-based AI was tested in several scenarios with different opponents. Performance was evaluated on win rate, accuracy of decisions, and expected value maximization.

- **Win Rate**: The AI consistently outperformed random players and showed competitive results against conservative and aggressive opponents.
- **Accuracy**: The AI's moves closely aligned with optimal poker strategies (e.g., folding weak hands and betting strongly with high-value hands).
- **Decision Efficiency**: Depth-limiting and Monte Carlo simulations enabled the AI to make decisions within reasonable time frames while maintaining strategic accuracy.

F. Conclusion and future work:

The poker game simulator successfully demonstrated the use of the Expectiminimax algorithm for decision-making in games of chance. The AI exhibited competitive decision-making abilities, simulating an intelligent poker player. Future work could focus on:

- **Enhanced Opponent Modeling**: Adding dynamic opponent modeling for adaptive strategies.
- **Improved Heuristics**: Refining the evaluation function to better approximate hand strength in uncertain conditions.
- **Parallel Processing**: Utilizing parallelism in Monte Carlo simulations for faster, deeper searches.

In conclusion, the Expectiminimax-based poker simulator represents a solid foundation for intelligent decision-making in games with incomplete information. With further optimization, it can potentially be expanded for more complex and realistic poker environments.