# Poker Game Project 2023

By Mohammad Hammoud, Omar Alsaudi, Yaman Tallozy, Hamid Moradi

## Abstract

An AI agent for a 5-card poker game has been implemented in this project. The agent is designed as a reflex agent, meaning it acts based on the strength of its hand without considering the opponent's bidding. The underlying strategy is to use machine learning, game theory, and decision-making algorithms to adapt and improve the agent's performance over time. The agent's performance was evaluated by comparing it against earlier versions of itself and a random agent. After the evaluation, our agent was a part of a tournament where it won against 4 different AI poker agents. PEAS framework is used to describe the agent's characteristics.

# Table of content

**Abstract** 1

**Introduction** 3

**Method** 4

**Result** 8

**Conclusion** 9

**References** 10

# Introduction

5 Card Draw Poker is a popular card game with a standard 52-card deck that requires at least two players. The game's objective is to bet on the hand's value (i.e., card combination), and the winner is the non-folded player who holds the strongest hand. Each hand/round in 5 Card Draw Poker contains 7 phases: the starting phase, the dealing phase, the forced betting phase, the first betting round, the card drawing phase, the second betting round, and the showdown phase.

In the starting phase, the server provides information on the game. In the dealing phase, each player is dealt five cards. The forced betting phase requires all players to place a bet known as an ante, which is necessary to proceed to the first betting round. The first betting round allows players to open, check, raise, call, go all-in, or fold. In the card drawing phase, players can draw and change card(s) as per their choice. The second betting round follows the same pattern as the first betting round.

The first bet of the game is called "open". Until the first bet is made, each player may "check". After the first bet, players may "fold", "call" to match the highest bet so far made, "raise" to increase the previous bet, or "all-in", which means betting all the coins a player has. The showdown phase reveals the winner of the game who holds the strongest hand.

The project aims to develop an AI agent to competitively play the 5-card poker game. As part of this project, all groups were tasked with creating a simple AI agent that can evaluate its hand and make decisions based on its strength of the hand. To effectively implement such an agent, it is crucial to have a thorough understanding of the game and its rules. This includes knowledge of the different hand rankings, betting strategies, and other key elements of the game. From an AI engineering perspective, the project requires the use of machine learning, game theory, and decision-making algorithms to create an agent that can adapt and improve its performance over time. The agent should be able to learn from past experiences and make informed decisions in real-time during the game.

In this project, we decided to implement our agent to be reflexed, which means it takes the hand strength and acts based on it without considering the opponent's bidding.

# Method

PEAS (Performance measure, Environment, Actuators, Sensors) is a framework used to describe the characteristics of an intelligent agent [1]. Each component will be described in the following:

- Performance measure: the performance measure is used to evaluate the agent's performance, such as winning percentage or expected utility in a game. In this project, we made our agent compete against earlier versions of the same agent and against the random agent to see and compare the performance in order to decide if the agent became better or worse. Afterward, make the necessary changes.
- Environment: The environment is the context in which the agent operates. The environment includes the game's rules, the other players, and the cards dealt to the agent. The environment is partially observable, where we know our cards, the number of chips other players have, and the amount of the pot. However, the opponent's cards are unknown to our agent as they are in a real poker game.
- Actuators: Actuators are the mechanisms that allow the agent to take action. Each agent has multiple choices to take. In the opening phase, the agent can open with a bid, check, or goes all-in. In the callRaise phase, the agent can fold, call, raise, or all-in. The last actuator that the agent has is to choose cards to throw in order to get a better combination.
- Sensors: Sensors are the mechanisms that allow the agent to perceive the environment. In the case of a poker agent, sensors would include
    - the ability to see the cards dealt to the agent,
    - the bets made by the other players, and
    - the current state of the game.

  Since our king Agent is just a reflex agent, we did not need to create a complex network of sensors like a memory agent and other agent need. Our sensors were only about considering the most critical factor for the game flow, such as our card, the minimum amount of money to bet, etc.

The agent identifies its hand strength and evaluates the following:

1. In case of getting a “high card” the evaluation function returns “low”.
2. In case of getting “One pair” the evaluation function returns “very low”.
3. In case of getting “Two pairs” or “Three of a kind” the evaluation function return “High.”
4. Otherwise, the evaluation function returns “very high.

The evaluation is related to the probability table of getting different rank when getting random five cards. As the Table 1 shows below

| | |
|---|---|
| *High card* | 50.1177% |
| *One pair* | 42.2569% |
| *Two pair* | 4.7539% |
| *Three of a kind* | 2.1128% |
| *Straight* | 0.3925% |
| *Flush* | 0.1965% |
| *Full house* | 0.1441% |
| *Four of a kind* | 0.02401% |
| *Straight flush* | 0.00139% |
| *Royal flush* | 0.000154% |

The decision-making process of our AI agent is based on an evaluation of its hand and the game's current state. Specifically, in the opening action stage, the agent can make one of three choices: open with a specific amount of chips, check, or go all-in. The agent's choice is determined by the strength of its hand as well as the probability of winning the current round. In the case of very high cards, the agent's decision to go all-in is informed by the low probability of receiving such a hand (between 0.00139% and 0.3925%).

In the case of a "High" hand, the agent will open with the minimumPotAfterOpen + 25, provided that the remaining chips are above this sum. The agent will go all-in if the remaining chips are below this amount. In the case of a "Low" hand, the agent employs a probabilistic approach, utilizing a random value between 0 and 1 to determine its action. Specifically, if the random value is less than 0.15, the agent will check. Otherwise, it will open with the RemainingChips divided by 16 plus the minimumPotAfterOpen. In the last case, if the random value is less than 0.3, the agent will check, otherwise, it will open with the minimumPotAfterOpen. This approach allows the agent to adapt its behavior based on the strength of its hand and the current state of the game, balancing the risk and reward of its decisions.

In the CallRaise action stage, the agent can “Fold,” “Call,” “Raise,” or “All-in.” The agent will take different actions based on the evaluation of its hand. If the hand is considered "Very Low," the player has a 20% chance to fold and an 80% chance to call the current bet. If the hand is evaluated as "Low," the player has a 40% chance to call and a 60% chance to raise, with the raise amount being calculated using the minimum raise amount and the player's remaining chips divided by 16. If the hand is evaluated as "High," the player has a 50% chance to raise by a set amount of the player's remaining chips and a 50% chance to go all-in. If the hand is evaluated as "Very High," the player will automatically go all-in.

The function is used to help the agent decide which cards to discard during the discarding phase. The agent will first identify the grouping of the cards in the player's hand. Based on this grouping, it will then select cards to discard that are not part of any pairs or groups. For example, if the agent has one pair, it will discard the remaining three cards that are not part of the pair. This approach is intended to improve the chances of the agent getting a better hand by discarding the less useful cards

From above, we expect our agent to make reasonable decisions to pluff the opponents with no fixed actions (When the actions are taken based on the chance value) and discard the less valuable cards.

## The functionality of the throwing cards phase.

If our Agent was still a part of the game and reached the stage of throwing cards here we handled the situation depending on the rank of our cards as the Table 2 below shows

| | |
|---|---|
| *High Card* | The King Agent throws four cards to keep the highest card in hand. The logic behind this idea is to increase the probability of getting a better rank, for example, getting a pair or two pairs after getting new cards for the one we got rid of. |
| *One pair* | In this case, the King Agent throws three cards to keep the pair in hand. Keep in mind that if we get rid of any of the cards that represent the pair, we will end up with a worse card evaluation (*High- card*) almost 50 % of the time, and we have only an 8% chance of getting a better rank of cards, According to our probability table. |
| *Two pair* | In this case, the King Agent throws only one card with the main idea of keeping the two pairs in hand. |
| *Three of a kind* | The King Agent throws two cards in this case with the main idea of keeping the *Three of a kind* in hand and maybe looking up for a (*Four of a kind* or *Full house*) |
| *Four of a kind* | The King Agent throws only 1 card in this case with the main idea of keeping the *Four of a kind,* the logic behind this move is not getting a better rank since we will not end up in *Royal or Straight flush, but it is more to confuse our opponent if they have a smart algorithm should not be able of guessing that we have a Four of a kind* |

Probability of winning the game against three players as we will be in the tournament

The probability of winning when the King Agent gets five cards with the rank of high card is when All three opponents are having a high-card too (0.501177)^3 with a worse hand evaluation. The probability could be computed with this mathematical expression (0.501177)^3 = 0.0157356037 * ¼ =

The Table 3 below shows the probability for winning for other ranks

| | |
|---|---|
| *High Card* | (0.501177)^3 / 4 = 0.0157356037 |
| *One pair* | (0.501177 + 0.422569/2)^3 = 0.36164644699 |
| *Two pair* | (0.501177 + 0.422569 + 0.047539/2)^3 = 0.85066579322 |
| *Three of a kind* | (0.501177 + 0.422569 + 0.047539 + 0.021128/2 )^3 = 0.94652939639 |
| *All other hight ranks are having over 98% win rate* | (0.501177 + 0.422569 + 0.047539 + 0.021128 + 0.003925/2 )^3 = 0.98322122707 |

# Result

To evaluate the performance of our AI agent, we initially planned to conduct a pre-tournament with other groups. However, due to a lack of participation or readiness from other groups. We instead took a different approach where we ran the developed agent against earlier versions of itself, as well as against a random agent. The results of these comparisons were then recorded and analyzed, as presented in Table 4. This approach allowed us to measure the performance of our agent relative to its past versions and a baseline random agent, providing valuable insights into the agent's decision-making capabilities and areas for improvement.

*Table 4: shows the results of six games.*

| **Game** | **Rounds number** | **Result** |
|---|---|---|
| 1 | 19 | Our agent |
| 2 | 16 | Our agent |
| 3 | 15 | Our agent |
| 4 | 15 | Our agent |
| 5 | 18 | Our agent |
| 6 | 27 | Our agent |

From Table 1, our agent won six games in a row. Those results show that the agent is suitable enough to compete against others. When it gets to the random agent, we notice that the agent does not play rationally because sometimes, when it gets low cards, it bids with a non-reasonable amount and vice versa. The random strategy can sometimes be effective as it can catch opponents off guard if they base their decisions on the amount of bidding.

# Conclusion

Using our engineering background and knowledge of probabilities, we implemented one of the best versions of a five-cards poker reflex agent. Usually other agents that can read opponent behaviors and bedding, such as (goal-based, utility-based, and model-based agents ) would easily win against a reflex agent, and that is something we thought about and decided to give our agent a little bit of randomness when it comes to bidding with its reflex behaviors. This is what makes it unique and hard to beat, even against advanced and intelligent agents.

King Agent can read its card, evaluate it and decide to bid a specific amount of money depending on how good the cards are. Sometimes, the agent would overbid, pretending to have a better card. That was our plan to win against intelligent agents.

If we had more time, we could make other probability tables that consider the game when we become three or two players. Our current probability table considers a game of 4 players, including our agent (like the tournament case). The change of probabilities could lead to other behavior because the probability of winning with one pair against one player is more than 50%. However, against four,  it would be much lower.

# References

[1] Russell, S. and Norvig, P., 2010. Artificial intelligence: a modern approach. 3rd. *Upper Saddle River, EUA: Prentice-Hall*.
[2] Jeff Duda, Probabilities of Poker Hands with Variations, Microsoft Word - 492 (iastate.edu) .