Simulating a Multi-Agent Mahjong game

Authored by: **Kelvin Mock**
Carleton University & University of Ottawa
Ottawa, Canada
kmock073@uOttawa.ca

Supervised by: **Dr. Alan Tsang**
alan.tsang@carleton.ca

ABSTRACT

Humans make decisions under high uncertainty every day. It is worth an investigation to see how one could establish a strategy to a complex problem in such situation, through simulations. Mahjong is an imperfect information game which involves high uncertainty, as the hand tiles are hidden from another player. As a result, given the vast hidden state space [10], simulating a mahjong game could be a great visualization about how people derive decisions under uncertainty. Inspired by existing agents and algorithms, I implemented the game's environment with some agents. Codes are available on GitHub: https://github.com/kmock930/Mahjong-Strategy-Simulation.git

KEYWORDS

Mahjong, Reinforcement Learning, Optimizing Hand Tiles, ShangTing distance, 5-block analysis, Monte-Carlo Tree Search, Supervised Learning, Unsupervised Learning, Global Reward Prediction, Oracle Guiding, Parametric Monte-Carlo Policy Adaption (pMCPA), Suphx, Markov Decision Process (MDP), uncertainty, trustability, RLCard, Multi-Agent Systems, extensive form game, imperfect information game, general-sum game, hidden states, action space, information set, dominant strategy, mixed strategy, regret value, Nash Equalibrium, finite game, game tree

ACM Reference Format:
Authored by: **Kelvin Mock** and Supervised by: **Dr. Alan Tsang**. 2024. Simulating a Multi-Agent Mahjong game. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 11 pages.

1 INTRODUCTION

Mahjong is a traditional **general-sum** 4-player board game originated from China. Each player holds 13 tiles at a time. Players can only see tiles on-hand. In each round, each player takes turns to obtain a new tile and then discard one from his deck. Meanwhile, a player can also obtain a tile when another player discards it, in the form of "pong" (碰), "kong" (杠), or "chow" (上), where the combination is either 3 or 4 of the same suit and rank, or 3 tiles of the same suit and in sequential order of ranks, respectively, and the player must reveal this combination. In addition, there are many winning combinations. A player tries to make his deck of tiles one. Players will never know which move (that is, whichever tile they discard) is a better one, until when someone achieves a winning combination, as known as "Hu" (胡). At the end of a game, we will calculate the score of the winning combination and decide who is losing the scores to the winner. In particular, if the winner obtains his last tile from another player to win, that player will be losing the scores to the winner. On the other hand, if the winner obtains the tile to win by himself, all players will be losing an equal score to the winner. The most undesirable situation occurs when the winner has obtained all his 'Meld' combinations, which refers to tiles that he reveals in the form of "pong", "kong", or "chow", from the same player, and the last winning tile by himself. Normally, everyone will be losing an equal score, but in this scenario, the scores that the other 2 players lose will be counted towards the loss of that player who has given all the "Meld" combinations to the winner. Therefore, the instructions have already illustrated the complexity of the game.

Nature of Mahjong. In short, Mahjong is a multi-agent game with imperfect information from each player's perspective, where each player only knows 13 tiles on-hand out of 144 tiles in total. It induces a large hidden state space which tells the uncertainty. Much importantly, this game introduces "sparse rewards", which means that non-zero rewards are given only at the end of the game, and there is no immediate feedback on whether each action choice made was good (or not) [10].

Objectives. Despite such complexity and uncertainty, human players still make a decision promptly, and therefore, this project aims to simulate a mahjong game by designing a strategy for an agent in the game that optimizes its hand tiles to win. Much research work has implemented similar agents based on the Japanese Mahjong. Since most rule sets of Mahjong share common properties that makes developing artificial intelligence (AI) players challenging [9], in this project, I am going to implement an agent in the Cantonese-style Mahjong, given its higher complexity of rules.

2 RELATED WORKS

There are multiple existing agents and algorithms which are capable of strategizing in a multi-agent Mahjong's gameplay.

MJX and MJai [10]. MJX is a popular open-source simulator of Japanese (Riichi) Mahjong games. Its rules are implemented based on the Tenhou online platform and tested by human experts. Although it relies on deep reinforcement learning algorithms with high computational processes, comparing with Mjai which is implemented in Ruby, MJX still maintains a fast speed during simulation, because it is implemented using a lower-level programming language like C++ instead of Python or Ruby. To address the challenge at the implementation, where we have less support in the community for a lower-level programming language, MJX adapts to the implementation with the use of Python, so that it reduces the time for researchers to adapt to the programming language, and thus, gives them more time to focus on their actual work while at the same time preserving a fast runtime. MJX also provides a

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

user-friendly interface to support further researching work. In addition, MJX uses a stateless protocol, so it never needs to store the current state of the game in memory during simulation. It greatly reduces the space complexity of the simulation. It is important to take reference of the advantages especially in terms of time and memory complexity.

Optimizing Hand Tiles [11]. A good strategy should respond well to different stages of a game. In an early stage of a Mahjong game, it is suggested to decide whichever suit a player is going to gather to "Hu", based on the ShangTing ("Hu") distance. By deciding a tile type which the player should aim for, it increases the winning probability by reducing the "Hu" distance, making the tiles on-hand more similar to a winning combination. There are many proven methods to evaluate the ShangTing distance, so an agent knows whichever move reduces the distance the most and thus most likely to lead to a winning hand. However, there is a massive number of permutations and combinations of Mahjong tiles. As a result, a systematic approach will be used and discussed further in the project. Typically, it analyzes hand tiles with a "5-block" hypothesis, which breaks down 13 tiles into 4 "Meld" combinations, which is a combination of 3 tiles (either 3 or 4 of the same rank or in a sequential order of ranks), as well as a "Joker" combination, which is a potential pair.

Monte-Carlo Tree Search [11]. In the later stages of a game, given the choices to "Hu" becomes much limited, the Monte-Carlo tree search algorithm is typically used to traverse from the current set of hand tiles (of a player) into different possible winning outcomes so as to guide the agent to take a move which maximizes the winning score. An advanced tree search could also prevent an opponent from winning from a tile which we discard. It uses statistical data to perform these essential steps: selection, expansion, simulation, and backtracking. In the simulation phase, an agent is suggested to predict an opponent's actions using the tree search.

Optimizing Strategies from Metrics. To decide whether a move is an optimized strategy (for example, whether a player should discard a tile, or whether a player should take a tile from an opponent from "Chow"/"Pong"/"Kong"/"Hu"), the paper [11] suggests 4 metrics which could be used in this project as an evaluation of our own agents:

Table 1: Agent Statistics Description

Metric	Description
"Hu" rate	The winning rate of an agent.
"DianPao" rate	The probability of an agent losing to another particular agent (excluding losing to Self-Drawn).
"Hu" points per game	An average score of winning points of an agent from multiple games.
Average total score	An average value of the agent's total score.

With the above approaches and metrics, we can decide the best move even when the outcome and the winning score are unknown during a game.

Reinforcement Learning [8]. Reinforcement Learning is a type of machine learning algorithm other than supervised or unsupervised learning. Suphx, which is an AI system for Mahjong games, suggests that before performing Reinforcement Learning, a model should be trained through supervised learning, which causes the requirement of large sample sizes. Suphx uses deep convolutional neural networks. It is trained from the logs of professional human players. Afterwards, the model is boosted through self-play reinforcement learning with the networks as the policy. A popular policy, the Gradient Algorithm, is suggested in the paper [8]. Several techniques of reinforcement learning are also proposed, which could be used in this project:

- Global Reward Prediction
- Oracle Guiding
- Parametric Monte-Carlo Policy Adaption (pMCPA)

In addition, from the model structure of Suphx, we understand the importance to implement a pipeline of more than one models, which comprise of:

Table 2: Model Descriptions

Model	Description
The Discard model	To decide which tile to discard in normal situations.
The Riichi Model (a.k.a. the "Hu" model)	to decide whether to declare "Hu".
The Chow model	To decide whether/what to make a "Chow".
The Pong model	To decide whether to make a "Pong".
The Kong model	To decide whether to make a "Kong".

From Suphx, we know how trustable a reinforcement learning model could be in a simulated Mahjong game, despite its limitation of datasets from feedbacks of professional players and its computational challenge to train a new model from scratch. Therefore, in this project, I decided to rely on pre-trained models, so that I could get reliable results of strategies of a reinforcement learning without the need to consume a large amount of computational resources.

RLCard [7]. RLCard is an open-source toolkit for reinforcement learning in the domain of card games, which also supports Mahjong, with a user-friendly interface. It is a great tool for building agents in imperfect information games with reinforcement learning. Therefore, it is used in this project.

An Existing Reinforcement Learning Model. Meowjong [12] is the first Sanma AI which is based on deep Reinforcement Learning (RL). It uses RL to determine an action in each round of a game. It is implemented according to the Japanese (Riichi) Mahjong. Its baseline model was trained with a highly complex Convolutionary Neural Network (CNN) using the Supervised Learning approach. The training data were game records from the high-quality "Houou" table on Tenhou, specifically 50,000 rounds from 2019. 10% of this data was used for validation, with an additional 5,000 rounds from 2020 reserved for testing. Meowjong consists of different models representing different possible actions such as "chow", "pong", "kong",

discarding a tile, or declaring "Hu". Eventually, the model is continuously improved through self-play with the RL approach from the baseline. In the phase where RL is being used, a model used 400 episodes where every 10 of them are being evaluated. Meanwhile, 500 rounds were being played in the experiment in each cycle of Winds (i.e., a full cycle of East, South, West, and North). From the above experimentation, RL is believed to be computationally far more expensive than solely supervised learning.

Markov Decision Processes (MDP) [9]. A MDP process is utilized to model sequential decision-making processes under uncertainty. It demonstrated probabilistic outcomes and a finite set of states, actions, transition probabilities, and terminal rewards. They all help to determine an optimal strategy by maximizing the expected payoff over a sequence of actions. An MDP process is typically an unsupervised model, which needs not to prepare for a dataset in advance.

3 BACKGROUND

To understand how humans make decisions under high uncertainty, it is important to investigate how one establishes a strategy to a complex real-world problem. Mahjong is an **extensive form** and **imperfect information** game, with vast hidden space which makes predicting an opponent's move harder. Through a simulation, I am going to visualize decision-making processes in a Cantonese-style Mahjong game.

Problem Statement

Uncertainty. Given high uncertainty, it is very difficult to strategize a best response. The **information set** is huge. There are up to 10^{21} hidden states [7]. Thus, deciding a move with a **dominant strategy** is barely possible. In this situation, a **mixed strategy**, which consists of probabilities of multiple actions, is formulated for each agent in each round. Theoretically, it is desirable to minimize the **regret value** from a **Nash Equalibrium**, to make sure all agents playing a strategy which will not cause them to prefer another one due to a higher utility in the outcome. The **regret value** $R_i(a_i, t)$ in round t with action a_i could be calculated below. It minimizes the difference of utility values between hypothetical actions and an actual action.

$$R_i(a_i, t) = \frac{1}{t-1}\left[\sum_{1 \le t' \le t-1} u_i(a_i, a_{-i,t'}) - u_i(a_{i,t'}, a_{-i,t'})\right]$$

Although it is a **finite game**, it is computationally nearly impossible to formulate or even to visualize the entire game tree. Each player could play any 13 tiles in each round, not to mention the ability to call for special actions, such as "pong", "kong" or "chow". A player never knows what tiles an opponent hold. Some professional player do know by deducing from the discarded tiles. It is still a guess, where the player will never know the exact set of tiles another player has on-hand. This entails huge uncertainty from the large information set in each round, making computing an optimized strategy which guarantees no regret for a player very difficult. Furthermore, a score is only evaluated once a player wins. It is only possible to backtrack (from a game tree) when an outcome where someone wins occurs. Especially in an early stage of a game, a game tree is not a feasible option to strategize moves.

Technical Challenges. In addition to the game's theoretic nature, it is also computationally heavy to derive the best response. From existing researches, supervised learning and reinforcement learning are 2 commonly known techniques to strategize a move for a player accurately. However, given the complicated set of rules and calculation of scores, it has already taken much computational resources to decide if a hand is a winning combination (once someone calls "Hu"), for example, not to mention the machine learning process. The lack of feedback from professional players also causes the lack of training data from real games. It makes supervised learning difficult. Thus, **supervised learning seems not to be a feasible option** in this project. Reinforcement Learning models do not need to be trained with a dataset, which is an ideal choice in this project. However, since it relies on the learning from game interactions, it needs to be simulated with a high number of games. Experiments in Meowjong [12] showed that it typically requires very high computational requirement. Each model is trained on 4 NVIDIA P100 GPUs with 64GB memory in total, and takes from 10 hours to 30 hours. Therefore, Reinforcement Learning is also not applied at the early stage of this project. Unless sufficient computational requirements have been satisfied, it is very difficult to strategize an action in a Mahjong game with a Reinforcement Learning algorithm. Given the limitations in computation, therefore, I decided to utilize those pre-defined mathematical approaches namely the ShangTing distance and a Markov Decision Process.

4 MODELING THE RESEARCH

As a result, this research is going to approach the following questions:

(1) Is there a so-called "optimal" strategy for a Mahjong player?
(2) What is "optimal"?
(3) Without an "optimal" solution for self reward, how to minimize losses?
(4) Like in Mahjong, how human makes decisions under high uncertainty?

5 METHODOLOGY

Implemented Rules

To determine whether a player wins, a 5-block hypothesis [11] is applied, where a player has to gather successfully 4 melds and a pair. To recap, a "meld" is a combination of either 3 or 4 tiles of the same suit and rank, or 3 tiles in the same suit with a sequential order of ranks. A standard hand allows a player to win, but with 0 point. Meanwhile, a player who wins with the following combinations with a higher score. A score will be accumulated if the player's hand matches more than one patterns. There are 2 tables below which show the winning combinations and the corresponding scores.

Table 3: Winning Combinations and Scores

Winning Combination	Score (points)
Standard Hand	0
7 Flowers	3
8 Flowers	Maximum possible
Thirteen Orphans ("十三么")	13

Table 4: Winning Combinations and Scores (Contd.)

Winning Combination	Score (points)
Pure Terminals (“清么九”)	10
Mixed Terminals (“混么九”)	6
All-in-one Suit (“清一色”)	7
Nine Gates (“九指連環”)	10
All Honors (“字一色”)	8
Mixed Suits (“混一色”)	3
All Pongs (“對對胡”)	3
Four Concealed (“四暗刻”)	Maximum possible
All Sequences (“平胡”)	1
Big Three Dragons (“大三元”)	8
Small Three Dragons (“小三元”)	5
Big Four Winds (“大四喜”)	13
Small Four Winds (“小四喜”)	6
Eighteen Arhats (“十八羅漢”)	13
Blessing of Heaven ("天胡")	Maximum possible
Blessing of Earth ("地胡")	Maximum possible

Furthermore, there are some patterns which give an additional point once the player wins, but they do not guarantee to win by solely themselves. They must be accompanied by at least one winning combination above.

Table 5: Patterns and Additional Points

Patterns	Additional Points
No Open Tiles (“門前清”)	1
Self-drawn (“自摸”)	1
Matching Flower	1
Full Flower Set	1
Matching Wind	1
Dragon in a Meld	1
Winning on the Last Tile ("海底撈月")	1

Note: Please refer to Appendix 1 for detailed explaination of each combination.

Structure of a Tile

A tile in mahjong has 2 properties: the suit and the rank. To see more examples of tiles, please refer to Appendix 1. An "encode" function has been implemented to convert tiles that contain suits in string format and ranks in either string or integer format into a single integer, by using ordinal encoding. It is essentially helpful to integrate with a Markov Decision Process or even a machine learning model for decision-making. A "decode" function has also been implemented to convert decisions made by the computer back into a human-understandable format.

Calculating the Resulting Utility

In a Mahjong game, a score is only evaluated when a deck of a player wins. Therefore, rewards from most states in the game are sparse. After calculating the score from a winning hand, 2 mechanisms are applied. Firstly, traditional players normally provide bounds of a score. Given $+\infty$ scores upon achieving certain winning combinations, an upper bound simplifies the calculation process. Because some players will try to win as soon as possible, without any considerations to maximize his possible scores, causing the game to be less exciting, players typically penalize occasions such as winning without a point by defining a lower bound of the winning score. In this project, since agents are assumed to be rational and mostly self-interested which maximizes its own rewards, only an upper bound is applied to a winning score. Secondly, there is a tradition to penalize fake declarations of "Hu". Some players might mistakenly declare "Hu" but his deck is not yet ready to win. Once a player declares "Hu", the game ends. Some players might even falsely declare on purpose, ruining others' chances to win, especially when they know their slim chance to win or their high chance to lose on high score. Therefore, such occasion is penalized in this project by giving the player a negative score with its greatest possible magnitude (i.e., $-\infty$ or $-$upper bound).

Playing Order

Unseen tiles are arranged into 4 walls, each of which consists of 18 tiles. Players play in an anticlockwise order while drawing tiles in a clockwise order. It also defines where the front and the back of the tiles.

Figure 1: Anti-clockwise Playing Order [13]

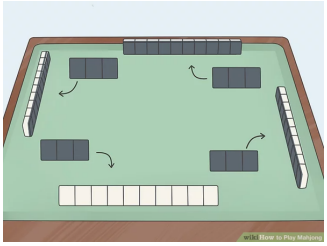
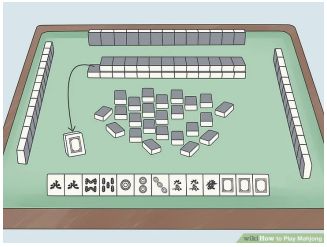


Figure 2: Clockwise Drawing Order [13]



Action Space

In each round, unless declaring "Hu" (to win), a player either takes a special action - "chow", "pong" or "kong" - or draws a tile from the front of the unseen tiles. If a player takes the action to "chow", a discarded tile (from another player) is taken so the player forms a meld in sequential rank, and the meld is revealed. Now, the player has less hand tiles, because some tiles have already formed a complete meld which is part of a potential winning hand. After taking "chow", the player discards a tile. If a player takes the action to "pong", a discarded tile is taken and a meld is revealed, with 3 exact same tiles. The player then discards a tile. Similarly, if a player takes

the action to "kong", a discarded tile is taken and a meld is revealed, with 4 exact same tiles here. However, the player uses an additional tile to form a meld. As a result, the player draws a tile from the back of the unseen tiles and discards a tile from its hand. In case of a flower tile is drawn, unless 7 or 8 flowers are gathered in total, which guarantees a winning combination immediately, the player must reveal the flower and draw a tile from the back of the unseen tiles. In summary, these are the actions which a player is able to choose in each round:

- Declaring "Hu" to win,
- Taking a special action - "chow", "pong", or "kong",
- Discards a tile

The Baseline

A default player is added to the simulation as a baseline in the comparison of their performances. It first decides whether its deck (with its hand tiles and open melds) can form a winning combination. If so, it declares "Hu". Otherwise, it prioritizes special actions. In case of multiple special actions being available in a round, this agent firstly considers "chow", then "pong", and finally "kong". If no special actions are taken, it discards the newly drawn tile immediately by default. This strategy makes its hand tiles unchangeable and thus less likely to win.

Strategizing with the ShangTing Distance

An agent relies on the ShangTing distance for decision making. When a deck has not entered the "Listen" phase where it waits for the last remaining tile to win, an effective strategy is proven to break down a deck according to a 6-block hypothesis, which involves the following tile types:

- a complete Meld,
- a compound Joker - an incomplete potential Meld.

Given the breakdown, the ShangTing distance [11] is used as a metric to determine the similarity between the current set of hand tiles and a possible winning ("Hu") combination. It is formulated from a set of 3 equations below, where N denotes the distance, m denotes the number of melds currently on-hand, n denotes the number of jokers, and q denotes the number of pairs.

$$N = 8 - 2 * m - n(m + n \leq 5) \tag{1}$$

$$N = 3 - n(m + n > 5, q > 0) \tag{2}$$

$$N = 4 - n(m + n > 5, q = 0) \tag{3}$$

The minimum value of a ShangTing distance is 1, which implies the current hand in "Listen", waiting for the final tile to win. The maximum value, however, is 8, implies a total mess in the current hand. Thus, the goal is to minimize the ShangTing distance.

Strategizing with a Markov Decision Process

Given the heavy computation of a supervised learning or a reinforcement learning algorithm, a Markov Decision Process is used to decide what actions have to be taken in each round. An MDP compares transitional probabilities in the action space from the current state to the subsequent hypothetical state. Inspired by Suphx, an agent uses 4 models in a MDP, which are the Chow model, the Pong model, the Kong model and the Hu model, with a customized logic to determine whether choosing to discard a tile is the best response in a round. To address the issue caused by sparse rewards at non-winning game states, base rewards have been assigned to each possible action. Discarding a tile loses 0.1 units of utility. Hence, it is a penalty which causes the model to prioritize other actions. Taking the action "chow" gives 0.5 units of reward. Taking the action "pong" gives 1 unit of reward. Taking the action "kong" gives 1.5 units of reward. Declaring "Hu" gives 10 units of base rewards, and the potential winning score on top of that. Therefore, this model prioritizes declaring "Hu", then "kong", "pong", "chow", and finally discarding a tile if no further actions are available in a round.

Visualizing the Game

This project visualizes an entire game's progress with a plain text-based .txt file. A well-defined visualization aids debugging. However, by implementing a Graphical User Interface (GUI) at the early stage of the project, it takes up much effort apart from the research of the problem so it seems unnecessary. Given the inability to print a large amount of text content on the terminal, the method of printing logs to a separate .txt file is chosen. It allows offline debugging and observation results from the game logs. Comparing with a .json format, a .txt format requires minimal knowledge of parsing and understanding the content itself.

6 RESULTS

From a series of simulations, the performances of different types of agents (including the default player, the ShangTing player, and the MDP-based player) are evaluated and compared with the following 4 metrics, as a recap:

- "Hu" rate
- "DianPao" rate
- "Hu" points per game
- The average value of the agent's total score

To determine the most effective metric, the experiment starts with simulations consisting of different number of games. These are the number of games simulated in 5 different sets of simulations: [100, 200, 300, 400, 500]. From the graph, the effect of the number of games towards a specific metric on different types of agents is clearly visualized below (at the next page) respectively.

7 DISCUSSION

An Analysis of the Results

"Hu" Rate. Since it is a probabilistic outcome, it ranges between 0 and 1. It is a normalized metric which gives a fair comparison under the same scale regardless of the winning points each player scores in certain games. From the comparison with "Hu" rate, it is summarized that the agent adopting a MDP shows greatest potential of winning. In general, all agents with a particular algorithm outperforms the one with the default strategy, which is to discard the newly drawn tile in each round, because the default strategy does not change the deck of the hand tiles nor have an attempt to make it more similar to a winning combination.

"Hu" Points per Game. This metric has an arbitrary range because it simply measures the winning scores in each winning game. Like

Figure 3: "Hu" Rate vs Number of Games

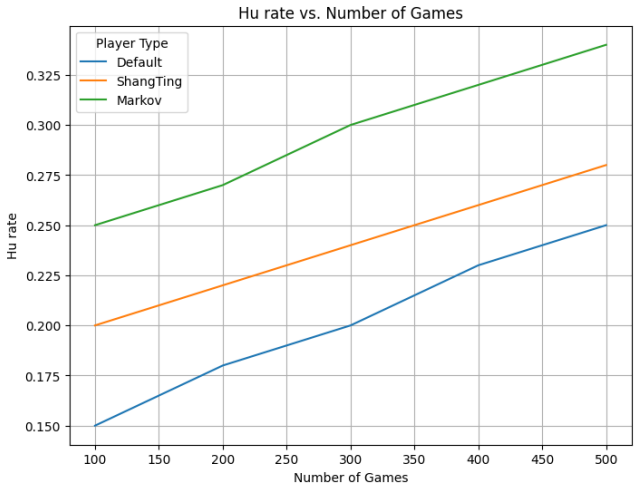


Figure 4: "Hu" Points Per Game vs Number of Games

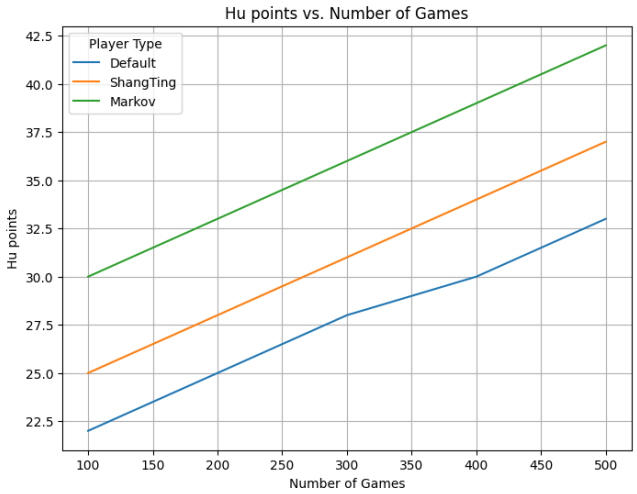


Figure 5: "DianPao" Rate vs Number of Games

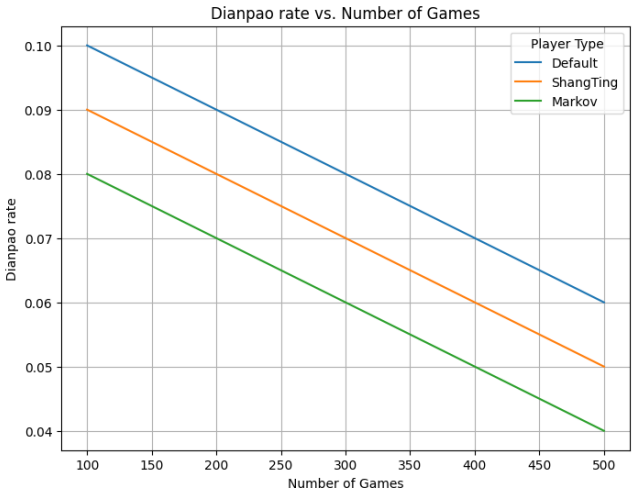
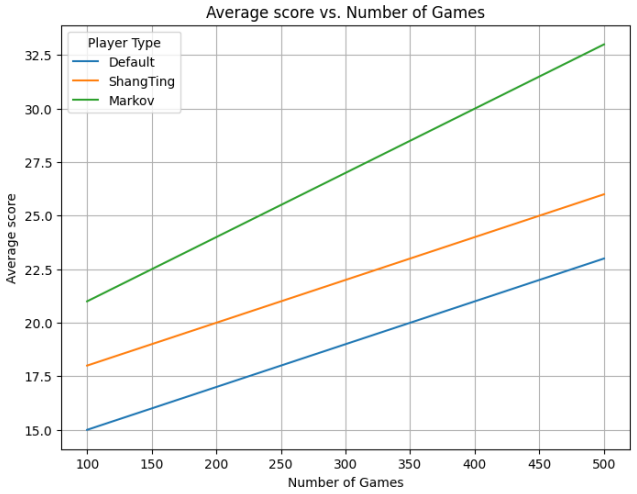


Figure 6: Average Score vs Number of Games



the "Hu" rate, it showed the same trends across different number of simulated games.

"DianPao" Rate. Since it is also a probabilistic outcome, it also varies between 0 and 1. However, unlike the above metrics, it evaluates the probability which an agent will lose to another player in a game. Thus, it measures the error rate of an algorithm. The results clearly showed that the agent with a default strategy is the most likely to lose to another player, since it does not consider whether the tile it is going to discard is a potential missing piece of another player's winning hand. It demonstrated the strongest ability of an MDP-based player to avoid losing to another player, which matches what it is concluded with other metrics.

Average Score. It is an average value of all metrics used. It combines all the advantages and weighted the trade-offs of those metrics. The results also visualize that the strongest agent is the one based on MDP, the second is the ShangTing distance-based agent, and the weakest agent is the one applying the default strategy. Interestingly, based on the changes of slopes on the curve, the performance of an MDP-based player has been visualized with a large growth here. Since an MDP is an unsupervised machine learning algorithm, it constantly improves its models by learning from experiences, from opponents' attempts in the same game, and results across multiple games. Therefore, this metric provides a fair comparison among different types of players across different number of simulated games and unintentionally reveals interesting insights.

The Best Agent. With all the metrics, it is generally concluded that the **MDP-based agent is the best agent**. The reason is its strong ability to improve itself. It iteratively computes a value of each state until fully improved (or called "convergence"):

$$V(s) = \max_a \sum_{s'} P(s' \mid s, a)\left[R(s, a, s') + \gamma V(s')\right]$$

From the equation, s is the an encoded state of the game; a is the chosen action from the action space A; P is the transition probabilities which evaluates the likelihood of reaching a new state after taking an action; and R is the resulting reward. It takes a list of past moves as a history for its decision-making process. It is formulated as an example illustrated below:

```
history = [
    {
        "state": {
            "hand": [0, 1, 2],  # Encoded tiles
            "open_melds": [],
            "discarded_tiles": [36, 37],
        },
        "action": "discard",
        "action_tile": 2,
        "next_state": {
            "hand": [0, 1],  # Tile 2 removed
            "open_melds": [],
            "discarded_tiles": [36, 37, 2],
        },
        "reward": -0.1  # Discouraged action
    }
]
```

From the above piece of code snippet, it is known that an MDP algorithm takes both the current state and its subsequent state with a list of encoded hand tiles, revealed melds, and discarded tiles. It also stores the action taken through the transition between 2 certain game states. This list provides a well-defined source from the game for the model's self improvement, by observing the probabilities between different transitions of states. Therefore, as the best agent, the MDP-based agent showed its strong ability of self-improvement.

Comparing the Mechanism of Other Agents. Apart from the best agent, undeniably other agents also improved constantly in terms of the "Hu" rate, "Hu" points per game, and "DianPao" rate, but the improvements are not as desirable as the MDP algorithm. Those agents do not improve well because the logic is generally hardcoded, which means that they perform the same logical flow for decision-making in every round. The ShangTing distance algorithm consists of a set of 3 hard-coded mathematical formulas which decide the best action to be taken. The default strategy is hard-coded to discard the newly drawn tile in every round. On the contrary, the ShangTing distance algorithm, for example, demonstrates other advantages. It does not make any prior assumption about what actions should be prioritized if they are available in a certain game's state, whereas the MDP algorithm and the default strategy prioritizes special actions over discarding a tile. Despite the availability to "chow", "pong", "kong", or even declaring "Hu", sometimes player need to carefully decide about the consequences. For example, by not rushing to declare "Hu", an agent could potentially aim for a higher potential winning score. By not revealing melds, tiles could be used flexibly in different melds in a potential winning hand, because otherwise tiles in an opened meld are not changeable and information is revealed to opponents. Therefore, despite the inability to improve an algorithm by itself, there are some other advantages demonstrated, and we need a fair enough metric to balance between the trade-offs.

Limitations

Robustness of a Strategy. Given a complicated set of scoring rules, the strategies might not consider everything robustly. Unlike the MDP algorithm which determines its best response from a set of transition probabilities, other strategies might not strategize to form a Thirdteen Orphans winning combination given any kinds of hand tiles. A Thirdteen Orphans combination is composed of 13 individual tiles and a pair. However, both the ShangTing distance algorithm and the default strategy play by forming melds under the 6-block hypothesis. Since the Thirdteen Orphans combination is not based on the composition of melds, those types of strategies might miss this winning combination which has a high score. Hence, from a particular example, some strategies might not demonstrate a desirable robustness.

Complexity of Rules. Rules in a Mahjong game are generally highly complex, but the implementation in this project might not demonstrate the most computational efficient solution. Given multiple nested iterations and sets of sorting logic which facilitates deterministic processing, the time complexity to determine if a winning hand matches the 5-block hypothesis takes $O(nlog(n))$. It takes $O(nlog(n))$ to sort the tiles. Then it iterates over $O(n)$ tiles to locate possible pairs. It finds potential melds by recursing over $O(n/3)$ levels (or $O(n/4)$ levels if it is a "kong" combination). Hence, the total cost in the worst scenario involves recursively exploring $n!$ permutations of the tiles to find valid melds, leading to a factorial time complexity $O(n!)$. Finally, it takes linear $O(n)$ time to count the number of melds and pairs. With pruning and early exits, the actual cost could be potentially lower and closer to exponential $O(k^n)$, where k is the branching factor which refers to the number of meld candidates in each step. However, it is still undesirable if the simulation scales up. To sum up, given an exponential time complexity of $O(n!)$, or $O(k^n)$ as an average case with pruning, it is uncertain whether the implementation of rules could be improved so we could reserve more computational resources on the strategies themselves.

Uncertain Reliability. This project relies heavily on existing algorithms, however, how trustable are they in the problem of resolving uncertainties in a Mahjong game? The default strategy is obviously not a desirable strategy because it never approaches to winning. The ShangTing distance algorithm uses pre-defined mathematical formulas, which forms a ground basis of trustability. However, what about the MDP algorithm? Since this machine learning algorithm is applied in a black-boxed manner, we never know how well it converges. Although the simulation is visualized with 4 metrics, how "good" are they in estimating the performances of agents? It is never known, unless a comparison of a large set of metrics is made. In such situation, how trustable is the solution of a strategy guaranteed?

8 FUTURE WORK

GUI. A nicely-looking **Graphical User Interface** which simulates the game visually aids efficient research.

Training Data. Trained with sufficient data, a **supervised learning** algorithm from game logs played by professional players will make the simulation's results more trustable.

Computational Resources. With more advanced computational settings, such as an advanced GPU and abundant memory, **reinforcement learning** could be applied to give more trustable results and to generalize a strategy which is closer to optimality.

Testing. Even though unit testing is conducted during the Test-Driven Development, especially in the implementation of the set of complicated scoring rules, an **exhaustive testing** approach (including gray-box and black-box testing) is essential to make sure every piece of logic runs as expected.

Accuracy of the Implemented Rules. The rules are implemented based on my own understanding of the Cantonese Mahjong game. However, it is uncertain whether the rules are complete and accurate. Thus, it should be **verified by several professional human players** so that the results of this research become more reliable.

9 CONCLUSION

From the comparison of performances from different agents in the simulation of a Cantonese style Mahjong game, it is theoretically possible but practically not ideal to find an "optimal" strategy for a Mahjong player which causes the least regret. To minimize losses, a player could either rely on proven mathematical algorithms (such as the ShangTing distance) or deducing the best response in the current game's state from the history of past actions made by each player. The latter is also more practical as a human being to make decisions under high uncertainty like Mahjong.

REFERENCES

[1] A Visualized Guide of Hong Kong style Mahjong [n.d.]. "https://play.agames.hk/index_2/game/mj/mjindex02.php"

[2] An image of all Flower tiles [n.d.]. https://upload.wikimedia.org/wikipedia/commons/d/d7/Canton_Mahjong_flower.jpg

[3] An image of Honor tiles [n.d.]. https://www.redbubble.com/i/sticker/Mahjong-Game-Honor-Tiles-%E9%BA%BB%E9%9B%80%E7%95%AA%E5%AD%90-Winds-Dragons-%E6%9D%B1%E5%8D%97%E8%A5%BF%E5%8C%97%E4%B8%AD%E7%99%BC%E7%99%BD-I-Love-Mahjong-Cantonese-by-PawaPotto/159228090.EJUG5 Mahjong Game Honor Tiles 麻雀番子-Winds, Dragons 東南西北中發白| I Love Mahjong | Cantonese | Sticker.

[4] An image of ordinary tiles [n.d.]. https://commons.wikimedia.org/wiki/File:Mahjong_eg_HK_carving.jpg

[5] An image of the All Sequences combination [n.d.]. "https://baike.baidu.com/item/%E5%B9%B3%E5%92%8C/625185"

[6] An image of the Eighteen Arhats combination [n.d.]. "https://zh.wikipedia.org/wiki/%E5%9B%9B%E6%A7%93"

[7] Yuanpu Cao Songyi Huang Ruzhe Wei Junyu Guo Xia Hu Daochen Zha, Kwei-Herng Lai. 2020. *RLCard: A Toolkit for Reinforcement Learning in Card Games.* Technical Report. Department of Computer Science and Engineering Texas AM University College Station, Simon Fraser University.

[8] Qiwei Ye Guoqing Liu Chao Wang Ruihan Yang Li Zhao Tao Qin Tie-Yan Liu Hsiao-Wuen Hon Junjie Li, Sotetsu Koyamada. 2020. *Suphx: Mastering Mahjong with Deep Reinforcement Learning.* Technical Report. Microsoft Research Asia, Kyoto University, University of Science and Technology of China, Tsinghua University, Nankai University.

[9] Kunihito Hoki Moyuru Kurita. 2021. Method for Constructing Artificial Intelligence Player With Abstractions to Markov Decision Processes in Multiplayer Game of Mahjong. In *IEEE TRANSACTIONS ON GAMES, VOL. 13, NO. 1.*

[10] Nao Goto Shinri Okano Soichiro Nishimori Shin Ishii Sotetsu Koyamada, Keigo Habara. 2022. Mjx: A framework for Mahjong AI research. In *2022 IEEE Conference on Games (CoG).*

[11] Chunyan Gan Xiaoman Yang Xiaochuan Zhang, Liu Liu. 2022. Research On Mahjong Game Strategy Combining Hand tiles Optimization and Situation Search. In *2022 34th Chinese Control and Decision Conference (CCDC).*

[12] Xiangyu Zhao and Sean B. Holden. 2022. Towards a Competitive 3-Player Mahjong AI using Deep Reinforcement Learning. In *2022 IEEE Conference on Games (CoG).*

[13] "How to Play Mahjong, wikiHow."" [n.d.]. "https://www.wikihow.com/Play-Mahjong"

APPENDIX 1 - MAHJONG RULES

Tiles in Mahjong

There are in total 144 tiles. There are 6 categories:

- the Characters ("萬子"),
- the Dots ("筒子"),
- the Bamboos ("索子"),
- the Honors ("番子"),
- the Flowers ("花").

The Characters, the Dots and the Bamboos have tiles with ranks 1 to 9. Each tile has 4 copies.

Figure 7: Ordinary Tiles [4]

The Honor tiles consists of the **Winds**: East ("東"), South ("南"), West ("西") and North ("北"); as well as the **Dragons**: "中", "發", and "白". Each tile of this category has 4 copies as well.

Figure 8: Honor Tiles [3]

Meanwhile, there are 8 flower tiles in the game. A full set of flowers consists of: Spring ("春"), Summer ("夏"), Autumn ("秋"), Winter ("冬"), while another full set consists of: "梅", "蘭", "菊", "竹". These are traditional flowers in China.

Figure 9: Flower Tiles [2]

Scoring Rules in Mahjong

There are many winning combinations, as mentioned in the report above. Here I will try to list the major ones which are used in this project.

Winning with Flowers. Regardless hand tiles, if a player draws in total 7 flower tiles, he/she can choose to declare "Hu" and win with 3 points. If a player draws in total 8 flowers, which are all of the flowers available in the game, he/she can choose to win with the maximum possible score (depending on the upper limit of points pre-defined).

Figure 10: Winning with 7 or 8 Flowers [1]

Thirdteen Orphans ("十三么"). It is formed with all the terminals, which are tiles with ranks in 1 or 9, in the Characters suit, the Bamboo suit and the Dots suit, and one of each honor tile, with a duplicate of any tile here. The deck in total should have 14 tiles to win.

Figure 11: The Thirdteen Orphans Combination [11]

Standard Hand. A 5-block formation is the basis of a winning deck, which incur no extra points upon winning. It is formed with 4 melds and 1 pair. A meld is formed with 3 or 4 exact same tiles, or 3 tiles of the same suit with a sequential order in their ranks. A pair is formed with 2 exact same tiles. There should be in total at least 14 tiles (including the one which the current player has just drawn, or another player has just discarded) to win.

Pure Terminals ("清么九"). It is formed with purely all terminals, with the 5-block hypothesis as the basis.

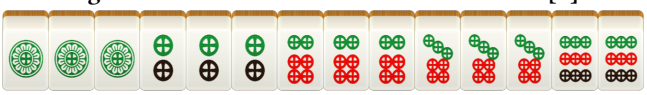
Figure 12: The Pure Terminals Combination [1]

Mixed Terminals ("混么九"). Like the Pure Terminals, this combination is formed with a mixture of honor tiles and terminals, with the 5-block hypothesis as the basis.

Figure 13: The Mixed Terminals Combination [1]

All-in-One Suit ("清一色"). It is formed with the entire deck of the same suit, with the 5-block hypothesis as the basis.

Figure 14: The All-in-One Suit Combination [1]



All Honors ("字一色"). As a variation of the All-in-One Suit, it is formed with all honor tiles in the entire deck. Those could be Wind tiles or Dragon tiles, with the 5-block hypothesis as the basis.

Figure 15: The All Honors Combination [1]

Nine Gates ("九指連環"). As a variation of the All-in-One Suit, it is formed with firstly the terminals being in a combination of 3 or 4 exact same tile, with then the remaining ranks to be in sequential order, with the 5-block hypothesis as the basis.

Figure 16: The Nine Gates Combination [1]

Mixed Suit ("混一色"). As a variation of the All-in-One Suit, it is formed with a mixed combination of one kind of suit (either in the Characters suit, the Dots suit, or the Bamboos suit), as well as with honor tiles, with the 5-block hypothesis as the basis. Because it is less difficult, the winning score will be lower than the All-in-One Suit's.

Figure 17: The Mixed Suits Combination [1]

All Pongs ("對對胡"). It is formed with all melds in either a "pong" or a "kong" combination, and a pair, with the 5-block hypothesis as the basis.

Figure 18: The All Pongs Combination [1]

Four Concealed ("四暗刻"). As a variation of the All Pongs combination, this combination requires winning without prior open melds, which means that a player gathers the winning requirements of the All Pongs combination solely by himself until the "Listen" phase. Given its difficulty, the score is relatively high.

All Sequences ("平胡"). It is formed with all melds in a sequential order of ranks, and a pair, with the 5-block hypothesis as the basis.

Figure 19: The All Sequences Combination [5]

Big Three Dragons ("大三元"). The winning deck should contain all Dragon tiles ("中", "發", and "白") in a complete meld, with the 5-block hypothesis as the basis.

Figure 20: The Big Three Dragons Combination [1]

Small Three Dragons ("小三元"). Like the Big Three Dragons, this combination allows one of the Dragon tiles to be appearing as a pair, while the remainings must be in complete melds, with the 5-block hypothesis as the basis. It gives less points because it is slightly easier to attain than the Big Three Dragons.

Figure 21: The Small Three Dragons Combination [1]

Big Four Winds ("大四喜"). The winning deck should contain all Wind tiles (East "東", South "南", West "西", North "北") in a complete meld, with the 5-block hypothesis as the basis.

Figure 22: The Big Four Winds Combination [1]

Small Four Winds ("小四喜"). Like the Big Four Winds, this combination allows one of the Wind tiles to be appearing as a pair, while the remaining must be in complete melds, with the 5-block hypothesis as the basis. It gives less points because it is slightly easier to attain than the Big Four Winds.

Figure 23: The Small Four Winds Combination [1]

Eighteen Arhats ("十八羅漢"). It is formed with all melds in a "kong" combination. Each meld should be formed with 4 exact same tile. It also follows the 5-block hypothesis as the basis. The winning deck will contain a maximum of 18 tiles in total. Given its difficulty, the score is relatively high.

Figure 24: The Eighteen Arhats Combination [6]

Blessing of Heaven ("天胡"). This combination is formed with the 5-block hypothesis as the basis. The only requirement is that the first player (i.e., the dealer) is able to win with the initial tiles. It is therefore formed with pure luck.

Blessing of Earth ("地胡"). This combination is formed with the 5-block hypothesis as the basis. Like the Blessing of Heaven, it is however determined by whether any of the remaining players (except the dealer) is able to win in the first round with the first tile drawn. It is therefore formed with pure luck.

Additional Points Upon Winning.

Matching Flowers ("正花"). Each flower tile has a rank. If its rank matches your playing order, only upon winning, an additional 1 point will be added to the winning score.

Full Flower Set ("一台花"). There are 2 sets of flowers in the game: ["春", "夏", "秋", "冬"] and ["梅", "蘭", "菊", "竹"]. If a player wins with a full set of flowers, an additional point is added only upon winning.

No Open Tiles. If a player wins without prior open melds, an additional point will be added. Note that this will not be the case of a Four Concealed pattern since the score attained from that pattern has already includes this additional point.

Matching Wind. There are 2 types of Winds in the game: the seat wind and the prevailing wind. The seat wind is simply the playing order. The East Wind will be favorable to the first player, the West Wind will be favorable to the second, and etc. The wind circulates after each game. The prevailing wind also starts with the order of East, South, West, and North. But it circulates every 4 rounds. If a player's winning deck consists of a meld with any kind of a matching Wind, an additional point will be added because of that.

Self-drawn ("自摸"). If a player wins with a tile drawn by himself/herself, an additional point will be added and all remaining players will normally be splitting the loss of scores to the winner, unless in the game there is a player who "feeds" 4 melds to the winner.

Winning on the Last Tile ("海底撈月"). If a player wins with the last tile drawn from the unseen deck, an additional point will be added to the winning deck.

Accumulating Points. All in all, points will be stacked if a player wins with multiple combinations.