CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

MAHJONG

AI/ANALYZER

A graduate project submitted in partial fulfillment of the requirements

For the degree of Master of Science in Computer Science

By

Dongchen Xu

December 2015

The graduate project of Dongchen Xu is approved:

Jeffrey Wiegley, Ph.D.	Date
Adam B. Kaplan, Ph.D.	Date
Richard Lorentz, Ph.D. , Chair	Date

California State University, Northridge

TABLE OF CONTENTS

ABSTRACT

MAHJONG

AI/ANALYZER

By

Dongchen Xu

Master of Science in Computer Science

Mahjong is a traditional tile game in East Asia. Although randomness and luck affect playing, calculation and analysis can increase the chance of winning. The main aim of this project is to develop an AI that helps a player to play mahjong with reasonable suggestions of the moves to be done in the game. From the help of the program, a player will achieve a higher rate of winning. The challenges in this project are developing an effective algorithm, and having it applied on a game platform that have enough number of players. Tenhou.net is online game platform selected to test the AI's strength versus human players. Game record from famous players are also used to prove the effectiveness of the AI.

I. INTRODUCTION

Mahjong is a popular game in East Asia area. Like some other games, mahjong needs skills, calculation of expectation, strategy and luckiness which makes it fun to play with other people.

However, because of the calculation can be hard, players may make mistake in the game. An AI that helps players in the game can greatly increase the speed of calculation. Players are more likely to win a game with the assistant of the AI. Also, the AI can help new players with a faster learning pace and less frustration. So programming an AI that does the calculation and analysis is useful to different level of players. As a programmer, it is interesting and challenging in game solving techniques.

In this dissertation, we will discuss on mahjong and the AI program have done by me in the aspects below:

1. The rules of mahjong
2. The algorithms that calculates and gives out suggestion in the program
3. The test results in different environments
4. The limitations and future goals of the project

II. MAHJONG GAME

From a learning curve point of view, mahjong is not an easy game. To understand the game concepts and the designing of algorithms, necessary terms and the game processes are explained in this chapter.

A. Game description

Usually, a mahjong game needs four players to start. There are also special rules for three players playing the game, but it is not discussed here.

Players initialize their starting hands by taking four tiles at a time, three times then draw an extra tile from the pre-built bank of tiles. Every player should have 13 tiles when the player is not on their turn. After finishing the initialization, players draw first, and then discard a tile each time in turn to collect a certain set of tiles. Whoever is the first to finish the collection is the winner of the round. Based on the difficulty of the collected set, the winner earns various amounts of points from other players. Please refer to appendix “point calculation” section for more information.

There is a dealer in every round. The dealer is the first player to draw and to discard. The dealer has an advantage of 1.5 times point earnings comparing to the other players. However, when other players get the winning tile by themselves, which will earn point from all other players including the dealer, the dealer will have to pay 1.5 times more than usual.

B. Game goal

A player should have more points than other players when the game ends by finishing collecting a certain set of tiles before all other players have done so to earn points from other players. Depending on the game situation, sometimes it is necessary to minimize point loss by defending.

Usually the first two players in scoring rank out of the four players are the winners in the game.

C. Game strategy

Like some other card or tile games, mahjong requires heavy calculation on probability and expected value. Because of the game's feature, a player can only obtain limited information needed to perform an accurate analysis in a game. However, there are still strategies that a player can do to get better results in the game.

1. Attack when possible, defend when necessary

The win and loss of a mahjong game is defined by the scoring rank at the end of the game. So if we want to achieve at least the 2nd rank when the game ends, we definitely need to win points from other players. There is barely any chance to win without gaining points from other players unless some opponents play badly and keep losing point to other players. It is always better to do everything we can to ensure our victory than hoping opponents make mistakes.

However, mahjong also depends on luck. We are not always able to obtain a good hand to win in a round. Because of the rule, if we discard a tile which an opponent needs to win, we have to pay all the points to the opponent by ourselves, while it can be 1/3 to 1/4 of the total depends on situation when the opponents obtain the tile by themselves.

Please check the “point calculation” section for details.

Think of this scenario: we are currently 3rd rank in the game. Our first target should be to overtake the 2nd player. Unfortunately, we discard a tile which he (2nd rank player) needs and lose 12,000 points to him. But we are lucky to obtain 12000 points from the 4th rank player. It looks like we return to where we were before losing 12,000 points, but actually:

Player(us): ±0

Player (which is at 2nd rank): +12000

Therefore, another 12000 points difference is made between the 2nd rank player and us.

Even if gaining points greatly helps us in the ranking, losing points hurts too, so sometimes defense can help us. By turning defensive, we may benefit from:

i. Greatly reduce our chance to discard a tile which an opponent needs, so there is a decreased chance we will have to pay the points ourselves.

ii. Since we are less likely to discard the tile which an opponent needs, they will have a lower chance to win, the round may end in a draw, and if we cannot have a ready hand at the end of the round, only a minor point penalty will occur (3,000 points penalty divided equally between all players who don’t have a ready hand).

iii. Some other opponents may discard the tile, so they have to pay the full amount, and we lose nothing.

A good player has a clear idea of when to attack and when to defend. Deciding what to do in a specific situation is a very interesting part of this game.

2. Different strategies in different phases of the game

In a round, the more turns it goes by, the more information a player can obtain. So we can divide the round into 3 phases:

Phase 1: Starting phase (0~6 turns)

At the beginning of the round, usually, a player should try his best going for an ideal hand. The ideal hand can be either a fast but cheap one, a slow one with more value, or somewhere in between which gives the best expectation. Since there is almost nothing that can be used to determine opponents' hand information, defensive moves are rarely used

Phase 2: Middle game phase (6~12 turns)

Players are getting close to a winning hand, so he have a better expectation of what pattern he will obtain. With more tiles discarded on the table, the player has a better knowledge of the number of certain tiles left in the bank. So he or she can intend to avoid building a hand with the tiles that their amounts are severely limited. In the case that some opponents have a faster hand and already show obvious signs of holding a ready hand (i.e. special move "riichi"), by reading the tiles that opponent have discarded, player can have a prediction of the tiles the opponent will probably need.

Phase3: Late game phase (turn 12 ~ end)

In this phase, more players can obtain a ready hand, so a player has to decide that if he still wants to compete in this round and go for a winning hand; or he may decide to turn to defensive play if he feels that his own hand is not worthy enough. Since there are nearly half of the total tiles revealed on the table, player will have a fairly good guess of the tiles the opponent needs. Also, since the rules forces that a player cannot use a tile that he discarded earlier to win from other players, the tiles that opponents have discarded provide a good guideline for determining which tile is safer at the moment.

D. Game detail

For more game details, please refer to the section “mahjong game introduction” in the appendix.

III. CORE ALGORHITHMS

A. Logic

This program does the analysis based as following:

Manual input hand information case:

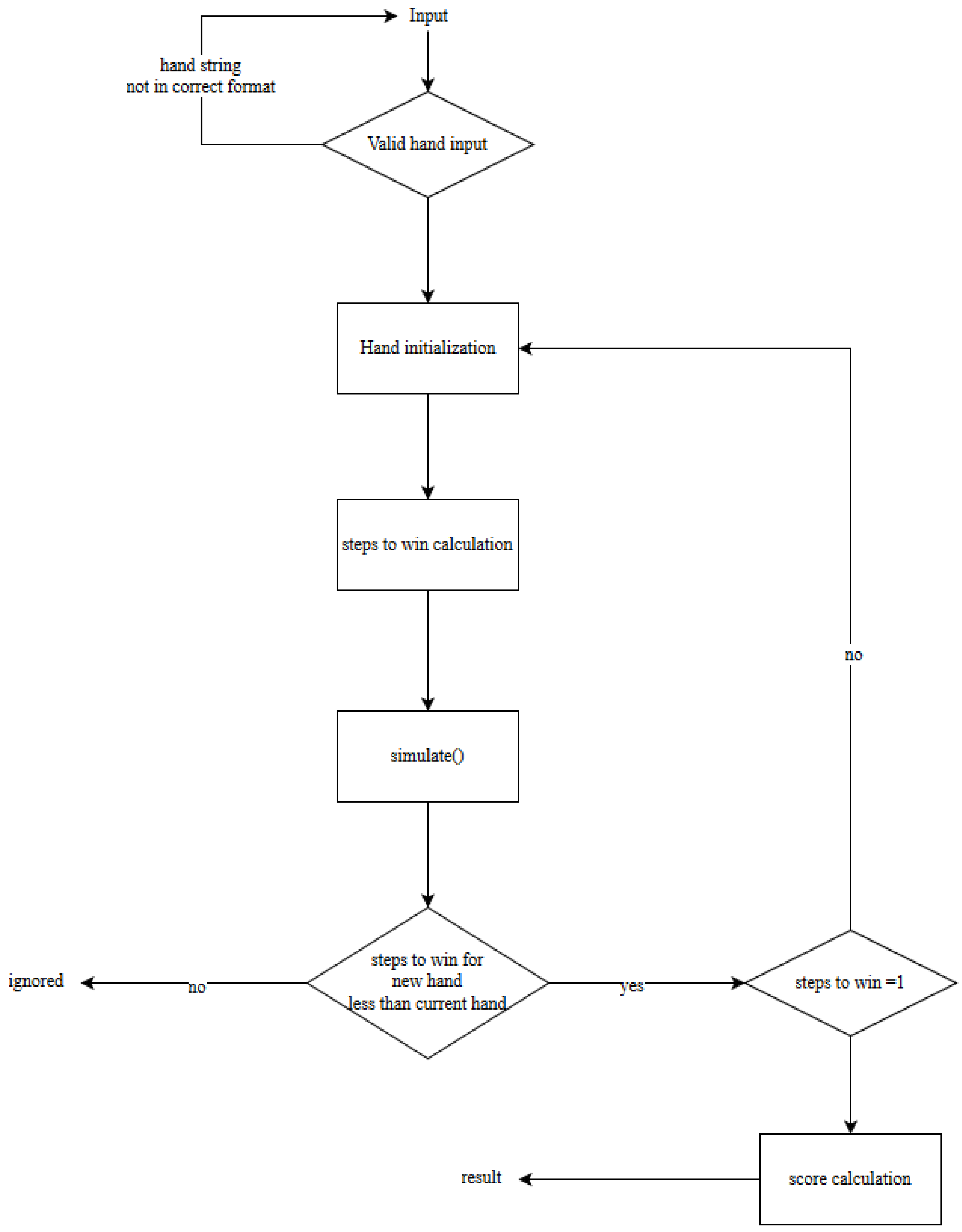
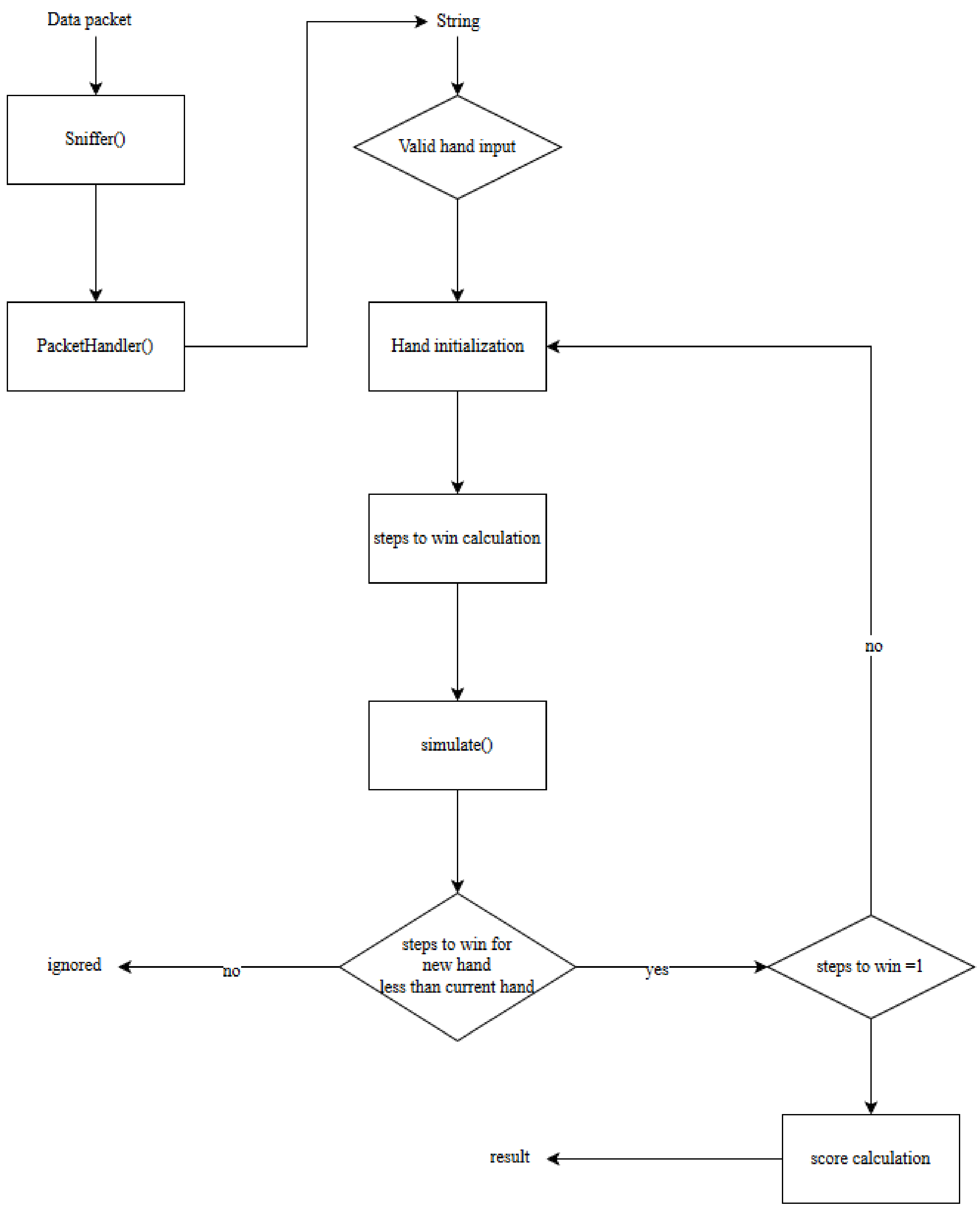


Figure 1, logic for manual input

While input from internet packet sniffing only have the differences as:



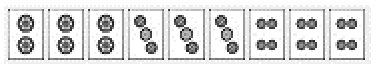
There is additional logic for the online playing with real player, please see the section "game strategy" below.

B. Processing data

Source of input is either from the sniffed data package generated by the sniffer class, or a manually inputted string from a user by using the command line. Either way, the final input is in the form like:

Numerical value + category sign

The numerical value will be the same as the target tile's numerical value. In the case of wind tiles, they are numbered as: east-1, south-2, west-3, north-4, white-5, green-6, red-7. For the category signs: characters-m, bamboos-s, circles-p, and wind tiles-z. These symbols are the same as what tenhou.net is using for easier programming. For example, bamboo 9 will appears as 9s in the input. There is an additional function added for users to input in a faster way: for tiles in the same category, user can input all their numerical values at once with the corresponding category sign after them. For example:



The pattern above could be entered as 222333444p by using this method.

In the sniffing data package case, the TCP packages received from tenhou.net for hand initialization have a format like:

<INIT seed="0,0,0,1,0,8" ten="250,250,250,250" oya="1" hai="21,79,51,1,11,48,70,3,124,110,23,60,126"/><U/>

In the data above, there are several important keywords:

INIT: the flag of hand initialization. The rogram clears the current hand information and reinitializes a new start hand when it appears in the data.

Ten: the current score for players in the game. The closer to the end of game, the clearer that minimum points needed for winning. For example, a player holding 25,000 points and the 2nd rank player is now having 33,000 points. The minimum requirement for the player to win is to get 4,000 points directly from 2nd player. So that any hand worth less than 4,000 points can be treated as zero because even if a player gets them, it’s still a loss.

Oya: indicating that if the player is the dealer or not.

Hai: the initial hand information. Notice that in the actual game logic, all 134 tiles are numbered. To convert it into our game input, simply divide it by 4 then plus 1, and ignore the remainder. The tile categories will be ordered as character (1~9)-circle (10~18)-bamboo (19~27)-wind (28~34). For example, the tile 11 above will be the 3rd character which is “3m” in the input to the program.

In a player turn, the data will be represented as:

<D p="110" />

which is much easier to understand, “drop tile 110”.

<T50/>.

“T” is stand for drawing.

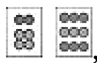
With these data inputs, the program obtains enough information for analyzing and providing suggestions.

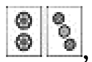
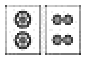
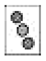
After receiving an input string, tile objects are created at once, then a hand object is created from the tile objects. Hand object initialization also calculate its steps to win and stores the result in a local variable. In the case a player drops a tile then draws another, the program pauses before the player gets the tile and recalculates the steps to win after the player gets it.

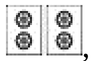
C. Steps to win calculation

A winning hand needs a pair and four groups of triples or straights. To calculate how many steps a hand needs to turn to a winning hand, different cases are discussed in this section.

All tile combinations other than a three-tile group can be summarized as:

, isolated tiles, we cannot form anything by using them.

, pre-straight, could add on one more tile to form a straight, not only limited to succeeding tile, tiles like  could also count as a pre-straight since we can add a  to it. Assume we choose any numerical tile as the pivot, any other tile with the numerical value ranging -2~+2 to the pivot can successfully build a pre-straight with the pivot.

, a pair, can also be treated as a pre-triple.

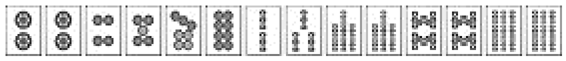
As discussed above, it takes two steps to build a three-tile group from an isolated tile. Since we have four groups, it will be at maximum eight steps to build all of them. Adding on the pair we need, nine steps in total.

The steps to win calculation is:

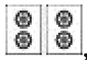
Int step = 9, which is the maximum possible steps to build a winning hand

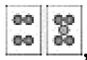
```
For i=0 to # of total tiles in hand
    If the tile is used, do nothing
    If tile i , i+1, i+2 can make a triple or straight, mark them as used, steps -=2
    Else if tile i , i+1 can make a pair or a pre-group, mark them as used, steps -=1
    Else, tile i is an isolated tile, mark it as used
```

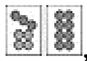
However, we need four three-tile groups at most. It may happens that we have too many pre-groups that doesn’t contribute further to the steps to win. For example:

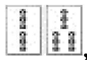


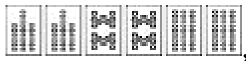
By the logic above, the steps will be subtracted as:

, minus 1.

, minus 1.

, minus 1.

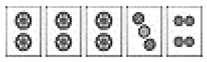
, minus 1.

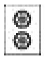
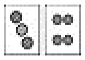
, minus 4(two straight).

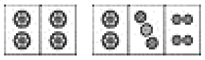
So steps will be 9-1-1-1-1-4=1.

If we want to choose two of three pre-straights, and make them to straights, at least two more steps are needed in this case. Therefore, we set up another modifier to the steps to win: useful pre-group which equals 5, including the pair we need. When we successfully built a three-tile groups, useful pre-group will substract one. If we have more pre-groups than the useful pre-groups, a plus modification applies to steps to win which equals the difference between the number of pre-groups and useful ones.

There is a special case:



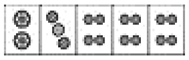
By using the algorithm, the triple of  is detected first, and  is treated as a pre-straight. This provides a -3 steps in STW calculation. However, the correct way of reading these tiles is:

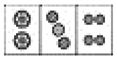
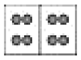


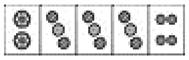
Which is a pair plus a straight, providing -3 steps in STW calculation and clears the requirement of a pair.

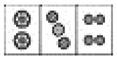
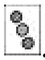
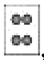
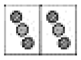
This special case won't affect STW calculation until the winning determination. Since the program is treating this 5-tiles group without a pair, it may detect a winning hand as a ready hand if the hand contains this pattern within. The solution is straight forward: try detect this special pattern in STW calculation and forcing the program to treat it as a pair plus a straight.

Similar case: the similar cases will not result in same problem.



 is detected first, and  treated as a pair, which is the correct way of reading these tiles.



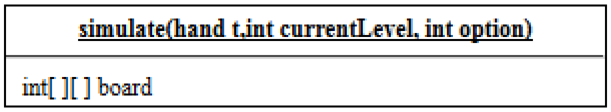
 is still detected first, since the program will try to looking for the next different tile while "looking" at . Since the next different tile is , a straight is detected. The remaining  is treated as a pair, which is correct.

Efficiency analysis:

The easiest way to determine steps to win is to do a breadth first search of the tree of hands to find the first winning hand. It results in a tree search with worst case 7 levels, and each node will have 17*34 children, which are $\sum_{n=0}^{7} 476^n \approx 5.5 * 10^{18}$ nodes. By performing the direct analysis using the algorithm above, only one traversing of the hand is needed, which is bounded by 14 calculations.

D. Recursive tree search

The program has a simulate () method that simulates the "drop then draw" move like a player does.



simulate(hand t,int currentLevel, int option)

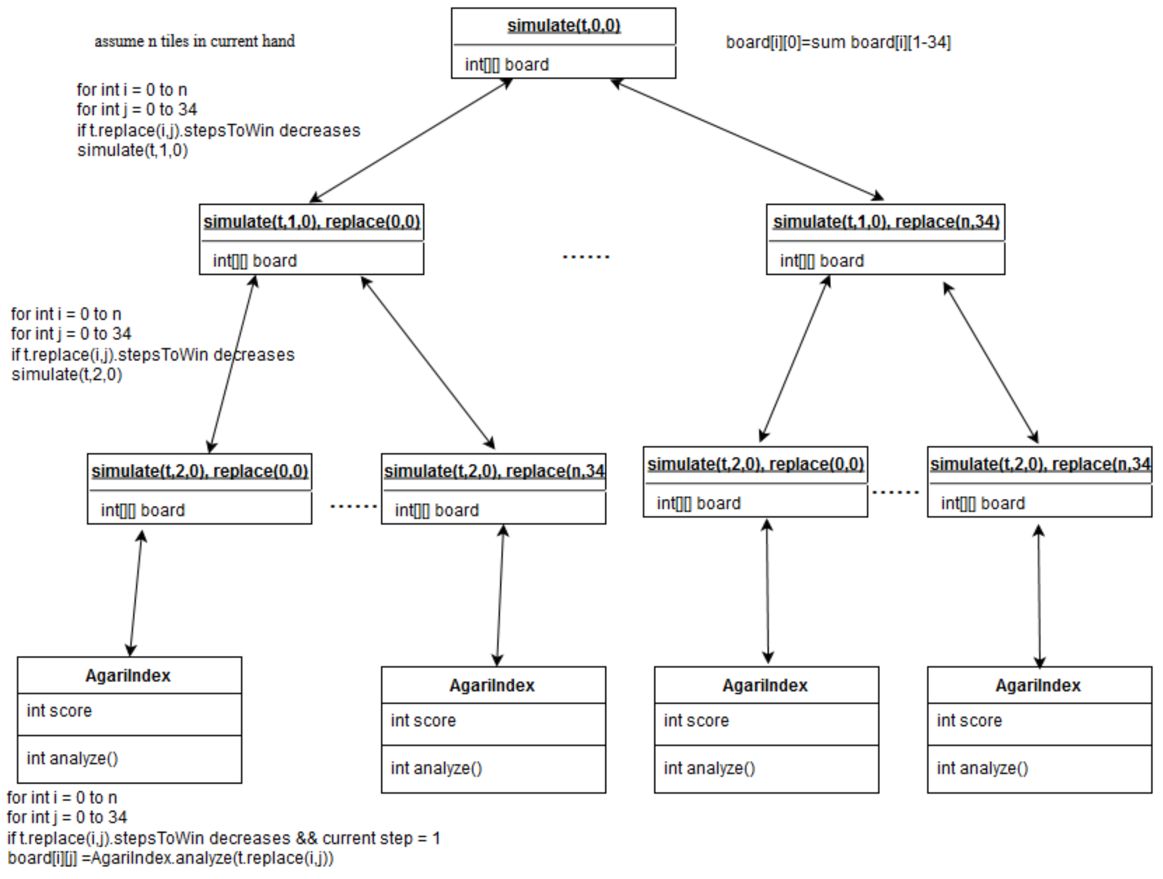
int[ ][ ] board

The pseudo code is:

```
For int i = 0 to # (tiles in hand)
      For int j = 0 to 34(total kinds of tiles)
            replace $i^{th}$ tile with tile #j
            recalculate steps to win
            if new steps to win < current steps to win, simulate(new hand)
```

"tile #j" means create a tile based on its order in the list, since we have a constructor for tiles with an integer parameter (i.e. 1 means character 1, 10 means circle 1, etc).

By performing code above, a tree structure will be created as:


assume n tiles in current hand
simulate(t,0,0)
int[][] board
board[i][0]=sum board[i][1-34]
simulate(t,1,0), replace(0,0)
int[][] board
simulate(t,1,0), replace(n,34)
int[][] board
simulate(t,2,0), replace(0,0)
int[][] board
simulate(t,2,0), replace(n,34
int[][] board
simulate(t,2,0), replace(0,0)
int[][] board
simulate(t,2,0), replace(n,34
int[][] board
AgariIndex
int score
int analyze()
AgariIndex
int score
int analyze()
AgariIndex
int score
int analyze()
AgariIndex
int score
int analyze()

```
for int i = 0 to n
for int j = 0 to 34
if t.replace(i,j).stepsToWin decreases
simulate(t,1,0)
```

```
for int i = 0 to n
for int j = 0 to 34
if t.replace(i,j).stepsToWin decreases
simulate(t,2,0)
```

```
for int i = 0 to n
for int j = 0 to 34
if t.replace(i,j).stepsToWin decreases && current step = 1
board[i][j] =AgariIndex.analyze(t.replace(i,j))
```

Then, the best choice can be decided by comparing the board[0~#of tiles][0] of the root node.
the max value tile means the tile with best value expectation.

AgariIndex is a class that contains all winning patterns in treemap form. By using the hand information, we know how the hand is divided into groups that fulfill the requirements of winning. Also, the number of straights, pairs, triples and quads are recorded down for pre-calculation purpose. When calling “evaluation” function, these abstracted data are useful in simplifying calculation. For more information of pattern matching, please see reference page for detail.

As we can see, by each level of the recursion, we replaced one tile of the hand to another which results in steps to win decrease by 1, while those moves that do not contribute are ignored. When we obtain a hand with steps to win equal to 0, we have successfully built a winning hand, and the score calculation applies.

When the recursion tree goes down to the leaf, we can actually calculate the score for the hand by using the pattern matching in the agariIndex class. This value is added to the scoreboard(i ,j) position. And scoreboard(i ,0) will be the sum of the rest of this row. After we return to the root node, the i^{th} tile with its scoreboard(i , 0) have the maximum value will be the suggested tile to drop.

However, due to the time limit for the online playing, the tree search process may not be able to go that far. So an option is added to the simulate method. The default searching depth is 2 levels in the tree, additional levels are searched according to the value of the option. If we cannot hit leaf level to get an actual score for the hand, the minimum score for the current hand is returned instead. In that case, we are only be able to retrieve a “shortest path”, which only takes probabilities into consideration, while the ideal case is using expected values instead.

E. Expected value of a hand

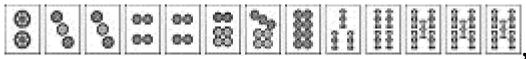
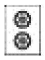
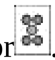
Since the game will use 134 tiles in total, the chance of drawing a specific tile can be simply calculated as:

$$P = \frac{\#\ of\ target\ tile\ remaining\ in\ the\ bank}{\#\ of\ total\ tiles\ in\ the\ bank}$$

After we have the winning hand at the bottom level of the recursion, the expected value is calculated as:

$\sum_{i=0}^{t} P * score(i)$, where t is the number of total possible winning hands.

For example:

, with the tile need for building a winning hand  or .

Let’s assume each of the tile we needed still have 3 in the bank, and the bank total is 30 tiles.

By drawing , P = 10%, score = 4000

By drawing , P = 10%, score = 2000

Then the expected value is:

$$R = 4000*10\%+2000*10\% = 600$$

600 will be the returning value to the upper level of recursion. The total expected value calculation as it show in the leaf node will be:

$$R*\prod_{i=0}^{stw} P(i),$$

where P(i) is the probability of drawing the ith tile needed to build a winning hand.

F. Game strategy

In the algorithm above, all calculations are done on a local information basis. That is, “how to discard a tile to obtain a winning hand with its expectation of speed and score at its maximum”. This seems greedy in a multiplayer game. However, it is not like a move in a zero sum game where it can be accurately evaluated. In a mahjong game, the information from opponents is usually not enough to make an assured evaluation of the situation. So we have to somehow “guess” opponents current progress of building a hand. So additional conditions will affect how the program will give out advice.

1. Speed versus score.

There are two main “triggers” to cause the program to significantly bias toward one side of speed or score.

The condition we should choose speed over score can be summarized as: we cannot/don’t need to build a high score hand. As an extreme example, the game is at south cycle game 4, and we are the 1st rank already. In this case, as long as we obtain a winning hand, the 1st rank is confirmed. So the score of hand means nothing to us anymore. In this case, simply set the calculated score equal to 1, so we find a “shortest path” to achieve the result of speed over score. In another case, still in the south cycle game 4, and we are 3rd rank 10,000 points falling behind 2nd rank player. Under this situation, we should definitely choose score over speed since if we cannot get a hand which is worth over 10,000 points or directly get 5,000 points from the 2nd rank player, our rank won’t change and it will end in a loss. So that by setting any winning hands worth less than 5,000 points to 0, we actualized the approach of score over speed to ensure a best chance of victory.

2. Defensive play

Defensive play will be activated after an opponent has a ready hand. We determine an opponent already has a ready hand under these situation:

1. The opponent declares ready hand.

2. The opponent has put 3 three tile groups as open hand.

For situation 2, if an opponent puts 4 three-tile-groups as open hand, then he is 100% percent to have a ready hand, since there is only one tile in his closed hand, and he needs a same tile to make the last tile a pair to win. However, since it takes turns to use “special moves” to build the 3 opened three-tile-groups, we assume he already has a pre-three-tile-group or a pair already. Therefore, his hand is at most 2 steps to win, which is same as 1 step till a ready hand. For safety purposes, we just treat it as if he has a ready hand.

Under defensive play mode, the program will use a fixed value as the threshold. If our hand has a better opportunity than the threshold, the program will act the same as normal mode, giving out instructions to achieve a best hand. If it doesn’t, the program will provide instructions to defense. The discard priority on defensive play mode is:

1. Opponent’s discarded tile

By the rules, opponent cannot use the tile that he discarded earlier to win from other players.

2. the 4th tile of a kind of wind tile

Since wind tiles can only be used to build pairs, triples, or quads, if the tile to be discarded is the 4th one, there is nothing that can be made from it. So it is also safe.

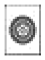
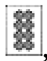
3. The tiles that someone else has discarded in the same turn

Please refer “Declare winning” section in the appendix. Since the opponent cannot declare win by the rules, it is also safe.

4. 3rd tile of a kind of a wind tile

This is not 100% safe, but the opponent will have very few chances in need of this tile. Since if he chooses to do so, if it is the case that this tile is in the 14 tiles which are not to be drawn, he'll never win this round.

5. Suji tile

Suji tile means the +3 or -3 tile of a pivot tile. For example, circle 4 has circle 1 and circle 7 as suji tiles. There are three group of suji tiles: 147, 258 and 369 in all three categories of numerical tiles. Consider the fact that pre-straights are easier to build comparing to other pre-groups, and players tend to build a "two-way" pre-straight such as 34, so we can use both 2 and 5 to finish the 3-tile-group. Moreover, if a player holds a long pre-straight like 23456 which can be treated as a pre-straight plus a finished group, he may use 1, 4 and 7 to finish building. Since this kind of hand building is effective, it has a higher chance to occur by players' choices. Because the rule of "furiten", a player cannot call "ron" on other players' discarded tiles if his winning tiles contain any tile he discarded earlier. Assume the opponent has discarded a , the suji tiles of  and are considered safer than other tiles since either opponent holding pre-straights other than 23 and/or 56, or he has enterred a "furiten" status and cannot declare "ron" on and . In 's case, the only possibility that the opponent can make use of it are holding a and waiting for another one to finish a pair to win; or holding two 's and waiting another to build a triple to win, which appears less likely to happen than pre-straights. will also be a safe tile but at a lower safety level since there is possibility that the opponent is holding and , so only 1 and 9 in this case are considered as a high level safety tile to be discarded. The rest of suji tile will have a priority lower than priority 6.

6. Wind tile that won't contribute multiplier to the opponent

   Since these tiles are considered to be useless to the opponent, there is little chance of him to holding these tiles.

7. The suji tiles mentioned in priority 5
8. Wind tiles
9. End point numerical tiles (1,9)

   In almost all the cases for defensive moves, these tiles are enough to provide a good guideline. If the round is still not finished after all tiles above are used up, a random tile will be discarded.

IV. CASE ANALYSIS AND STATISTICS

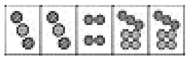
In this chapter, different cases are discussed to explain how the algorithm works to help a player finding the best tile to discard.

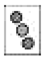
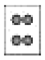
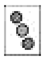
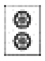
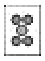
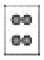
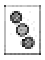
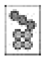
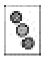
A. Separate hand analysis

Under this mode, the program calculates the best tile to discard without any information other than the hand itself. Therefore, the result will reflect only the best result the player may achieve from the current hand, which provides a good instruction of how to build a winning hand effectively. The cases below are ordered by complexity. If a hand appears to be less than 14 tiles, the rest of the tiles are treated as "opened hand" and provides no extra multipliers. For all unrevealed tiles, we assume they are in the bank, since even in real game, we have no enough information to tell exactly if the tiles we need are in the bank or held by opponents.

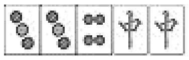
(All these cases below will be replaced with a picture in the final paper)

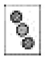
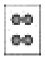
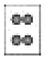
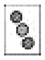
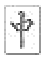
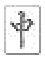
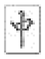
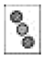
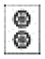
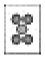
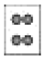
Case 1: Choices with same score but different number of usable tiles



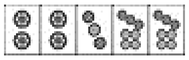
Player may drop a  or a  to have a ready hand. Based on analysis, they have the same multiplier (1 from tanyao). However, by dropping  and having  and  as the last tiles we need to win, we have an advantage of 4 more tiles in the bank rather than discarding  and having  and  as the last tiles we need to win. So  is suggested.

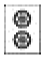
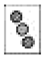
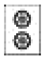
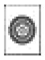
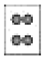
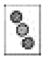
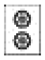
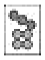
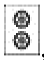
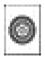
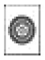
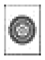
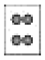
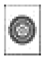
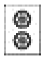
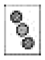
Case 2: Choices with different scores and different numbers of usable tiles



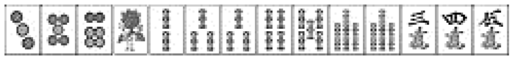
Player may drop a  or a  to have a ready hand. Based on analysis, they are worth a different number of points. Discarding  and having  and  as the last tiles we need to win may result in one more multiplier by having  as the winning tile since a triple of  is worth one more multiplier. By dropping  and having  and  as the last tiles we need to win, we have four more tiles in bank advantage but with no multiplier at all. So  is suggested.

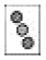
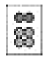
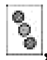
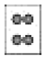
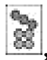
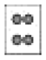
Case 3: Choices involve with probability of no multiplier



Player may drop a  or a  to have a ready hand. Based on analysis, they have the same multiplier if player wins (1 from tanyao). By discarding  and having  and  as the last tiles we need to win, we have 4 more tiles in bank advantage than discarding  and having  and  as the last tiles we need to win. However, in the case of discarding , if we happen to draw  as the last tile, there is no multiplier since in order to apply the one multiplier from "tanyao", it requires no numerical tile 1, 9 or any wind tiles in the hand. So  provides no contribution to winning. Moreover, we have to discard  if we draw it. Based on the rules, we enter a "furiten" status which we cannot use  from other players as the winning tile. This roughly reduced our chance to win by 3/4. So in this case, the two choices have the same score, same tiles in bank (since  doesn't count) but discarding  has the extra risk of "furiten",  will be the advice.

Case 4: Tie breaker



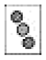
Player may discard a  or a  to have a ready hand. By discarding , we have a ready hand worth 1 multiplayer from "pinhu" with winning tiles  and , 8 tiles in total in the bank. By discarding 6m we have a ready hand worth 2 multipliers from "sanshoku". The hand has  as the winning tile, 4 tiles total in the bank. Now we have the choice between:

1,000 points and 8 tiles in the bank

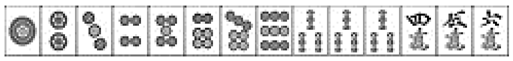
2,000 points and 4 tiles in the bank

They are same in the expectation calculation. We can actually treat them as a tie. However, we still have to break the tie to give out a suggestion. Here, I make the program choose the one with higher probability. There are two reasons:

1. The choice with higher probability is faster, there's less chance to be interfered by other players.
2. The choice with more winning tiles will be less possible to have all winning tile being held by other players.

Based on the reason above,  is given as suggestion.

Case 5: Hand with a good number of winning tiles but no multiplier.



Player may discard 🀒 or 🀡 to have a ready hand. By discarding 🀒, we obtain a hand worth 2 multiplier from "iizu" if we can draw an 🀠. And there are 4 🀠's in the bank. If we discard 🀡 instead, we have a ready hand worth 0 multiplier but have 1p, 🀜 and 🀟 as the winning tiles which adds up to 9 tiles. By now, discarding 🀒 is definitely a better choice since we cannot really obtain a winning hand by discarding 🀡 since we will have no multiplier. We can perform the "riichi" option to add 1 multiplier to the hand to clear the minimum 1 multiplier requirement if we want to discard 🀡 and go for a winning hand. However we can do the same move for the case that we discard 🀒 instead, which we will have 3 multipliers in total. So the comparison becomes:

1,000 points and 9 tiles in the bank

4,000 points and 4 tiles in the bank

The second choice is still better based on expectation. So 🀒 is the suggestion.

There are many other cases, but they can be generally summarized to become similar as the cases above.

B. Game record analysis

On tenhou.net, game histories are recorded and there are championships of famous and skilled players. Here I will let the program calculate for the result and compare with the move that the professional players made and explain the reason if the moves are different.

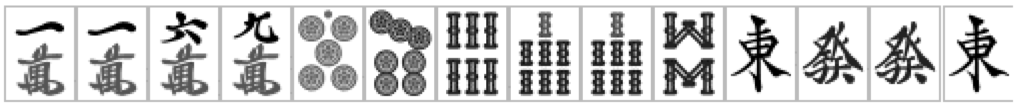
Game selected: 2nd Tenhou famous player championship, Game 1, table 1, the record URL can be found in the appendix.

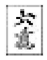
Here we will use the view from player Fukuji Makoto:

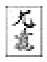
Game description: South cycle, game 4 round 0.

Score and ranking: 1st: 45,300, 2nd (player): 38,400, 3rd:25,600, 4th: 10,700

Initialization:

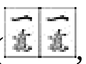


Player choice: 

Program choice: 

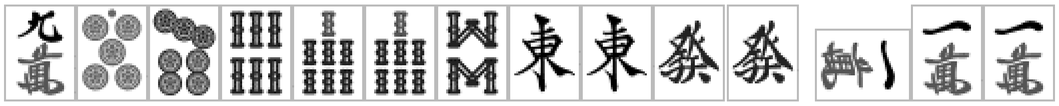
Player is now on 2nd rank, if we decided to go for 1st rank, 3,450 points are needed to directly get from 1st rank player, which the closest score is 3,900(multiplier 3). It is doable with riichi, hatsu triple and 1 dora. However, since we need “riichi” to fulfill the requirement of multiplier 3, it becomes not so practical since we will lose the flexibility to choose who we want to obtain the points from because of the existence of the “furiten” rule. Therefore, if we are going for a winning hand, the fastest approach should be used so that we can keep our 2nd place when the whole game ends.

Since 2nd place is considered as winning, the program actually did choose the fastest approach to discard 9m.

The player Fukuji Makoto discarded 6m instead, which is an intermediate move between fastest approach and defensive play. Since we already have enough pre-group (,

🀝🀟, 🀕🀖, 🀖🀗, 🀀🀀, 🀅🀅) and an easy multiplier from the hatsu triple in the future to clear the requirement of 1 multiplier minimum, we can start discarding tiles that is more possible to be used by other players without affecting our speed much. These tiles may become winning tiles in the future. So by discarding them earlier, it is safer while we are building our hand.

Turn 2:

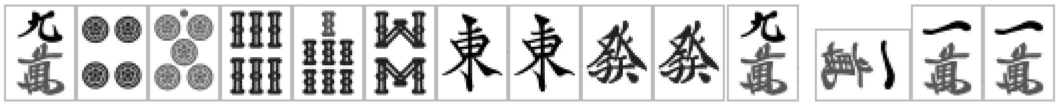


Program choice: 🀏

Player choice: 🀖

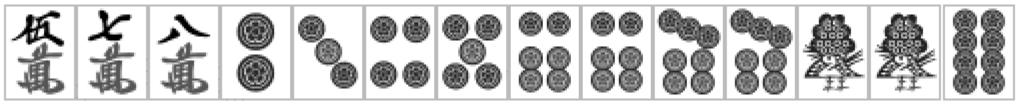
Same reason as first move.

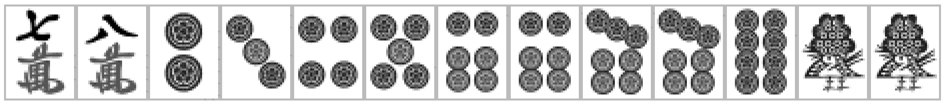
Turn 5:

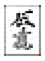
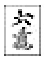
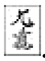


Here player was lucky to draw another 🀏 to make it a pair. If we discard 🀏 as program suggested, we lose the chance to make it a pair. However, between having the extra pair of 🀏, we are not advancing with fewer steps to win. Moreover, both final score and number of usable tiles are same. The only difference is since we have the pair of 🀏, we have one useless pre-group, and 🀀 is a slightly safer tile to discard than 🀏. Because it is the 5th turn and this round is still in an early phase, it doesn't make a significant difference.

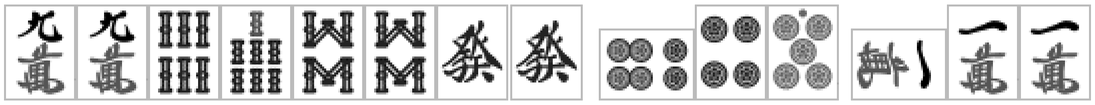
Turn 5: opponent's turn to discard:

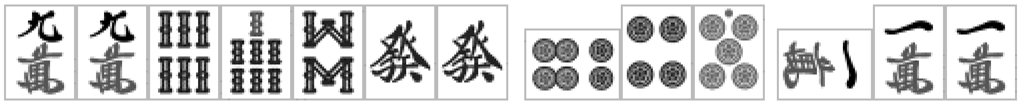


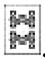
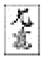
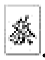
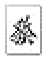


Here, the dealer discarded  and declared “riichi” with winning tile  and . Since we are in 2nd place, the program turns into defensive play.

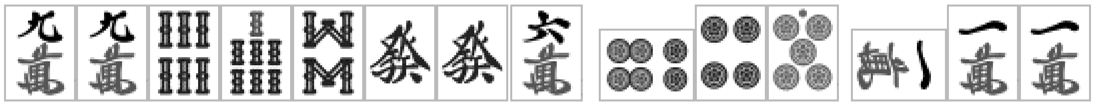
Turn 7:

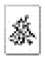


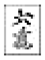


Here, player obtains a ready hand by performing “chii” move then discard . We have 2 kinds of winning tiles:  and . However, only  triple will grant us the necessary multiplier to declare a win. So the total number of winning tiles is 2, and the hand is worth 2000 points (multiplier 2, hatsu, dora*1). However, since the program turns into defensive play, “chii” move will not be executed.

Turn 10:



Program choice: 

Player choice: 

6z is relatively safer than any other tiles in the hand. So it is the suggestion from program. The player choose to maintain a ready hand and discard 6m, and unluckily becomes the one have to pay 11,600 points to the dealer.

From this case, it shows that the even if the program shows “extreme” in some situation in reference to its “one or the other” choice between attack and defense, the suggestions it’s giving out are fairly effective and turns out to be better sometime than the real professional player.

C. Real game analysis on tenhou.net

	pt	rate	final score	1st	2nd	3rd	4th	chance of winning in a round	chance of discarding a tile opponents need	chance of performing special moves	chance of "riichi"	total games	# of players
new	0	1488	-21	0	0.263	0.322	0.542	0.139	0.194	0.294	0.152	0	133329
Rank 9	0	1494	-22	0.207	0.241	0.26	0.376	0.204	0.164	0.323	0.183	4	28302
Rank 8	0	1499	-23	0.226	0.243	0.258	0.35	0.212	0.159	0.327	0.187	8	14606
Rank 7	16	1505	-24	0.234	0.247	0.255	0.336	0.216	0.157	0.329	0.189	14	16757
Rank 6	17	1515	-25	0.239	0.249	0.255	0.33	0.219	0.156	0.33	0.191	22	10084
Rank 5	17	1526	-14	0.245	0.25	0.254	0.32	0.222	0.155	0.332	0.193	30	7086
Rank 4	25	1533	-17	0.246	0.249	0.253	0.316	0.222	0.155	0.333	0.192	39	6425
Rank 3	39	1547	-1	0.247	0.254	0.251	0.311	0.224	0.154	0.333	0.192	50	6591
Rank 2	31	1547	-25	0.245	0.251	0.252	0.317	0.223	0.156	0.337	0.192	66	5889
Rank 1	22	1492	-921	0.218	0.235	0.253	0.371	0.207	0.17	0.35	0.179	235	7801
1 Dan	194	1562	-489	0.239	0.244	0.251	0.329	0.221	0.164	0.352	0.186	381	15250
2 Dan	381	1643	275	0.254	0.25	0.248	0.305	0.229	0.157	0.346	0.183	596	18285
3 Dan	563	1726	1324	0.264	0.254	0.246	0.288	0.233	0.15	0.344	0.179	880	14500
4 Dan	727	1825	2012	0.264	0.255	0.246	0.283	0.232	0.144	0.346	0.174	1224	10010
5 Dan	890	1922	2640	0.263	0.256	0.248	0.275	0.228	0.137	0.346	0.171	1629	5958
6 Dan	1015	1989	3371	0.262	0.259	0.252	0.267	0.226	0.13	0.355	0.168	2056	3129
7 Dan	1166	2072	4122	0.262	0.261	0.254	0.261	0.224	0.125	0.357	0.167	2376	1843
8 Dan	1253	2143	5968	0.264	0.261	0.253	0.259	0.222	0.121	0.364	0.164	3213	599
9 Dan	1304	2202	6987	0.262	0.263	0.255	0.257	0.221	0.118	0.366	0.162	3791	119
10 Dan	1475	2243	9677	0.273	0.263	0.252	0.255	0.222	0.114	0.363	0.163	3354	15
Phoenix	2200	2253	15475	0.276	0.26	0.242	0.264	0.225	0.125	0.356	0.163	5625	9

The table above is the statistics of players’ average data in different ranks. The ranks are in an increasing order.

Tests on tenhou.net with real players are also conducted. So far, 49 games are played. The statistics are below:

Rank distribution:

1st	32.6%
2nd	22.4%
3rd	22.4%
4th	22.4%

Chances of winning a round: 26.6%

Chances of discarding an opponent’s winning tile: 15.2%

Although it is maybe too early for a conclusion since the data is showing a biased result because of the sample size is not large enough, we start to know the characteristic of the AI and can have an evaluation of it.

By looking at the rank distribution, it appears to be biased to the 1st rank instead of uniformly distributed on all ranks. This could be the result of the “extreme” type of moves. Because there is no “intermediate” thinking in the program, it will suggest in an extreme way (attack or defense, no third choice in between). The total winning percentage is 55% (1st and 2nd rank), so it is doing fine so far. However, more game need to be played to prove its effectiveness. The confidence interval for current data is 41.07%~68.93% on 95% confidence level.

In the aspect of building a hand, 26.6% single round winning rate is above average. It may result from an unbalance of the “speed versus score” design. However, the high winning rate of the game may suggest extraordinary luckiness. Final conclusion should be done after enough tests are made.

In the aspect of defensive play, 15.2% to drop an opponent's winning tile is fairly average, while top players can decrease it to 11%.

V. Conclusion

From the test cases and statistics, the program has met the expectation in:

1. Effectively build a winning hand with taking both speed and final score into consideration.
2. Read the data packets from network communication and make use of them to give out suggestions to help players playing.
3. Defensive moves are effective against human players.

However, the program still have the limitation and can be improved on:

1. Recursive search method can be improved in efficiency to perform a full search of the game tree in the game versus human player(since there is a 5 second limit for each turn)
2. Some data packets from tenhou.net are still unable to read because the representation of 3-tile-group as an open hand of opponent in the data packet is unknown. Although there is a 4~5 digit number is used for the representation, the correspondence of tile number (which is 0 to 133, each one represents a tile in the game) to the patterns number (the 4~5 digit number used in the data packet) is not a polynomial style calculation. The only thing we can do is to have enough tests so the correspondence can be recorded. Then a fully automated test is possible since we have knowledge of all data packets used in the game.

In the future, the program will be redesigned so:

1. A more user-friendly interface will be made for suggestions from the program.

2. With a more effective tree search algorithm, the simulation function is not limited to only advance when having a decreased steps to win, but can also apply to the case when the new hand have a same steps to win or even increased steps to win. So every possible case is discovered to perform an extremely accurate analysis.
3. Automated tests are available so the program can adjust the threshold value of turning defensive play itself. Which will achieve a better performance against human players.

With the works to be done in the future, this program will be a very effective tool for players at any level to play mahjong better. I hope this project can help more people developing their interests on mahjong.

REFERENCES

Tadao yamaoka. "How to use index method to pattern matching a hand". http://hp.vector.co.jp/authors/va046927/mjscore/mjalgorism.html , Retrived Dec.2015

APPENDIX

A. Mahjong game introduction

1. Tiles

In a normal game, a total of 136 tiles will be used, including: 36 character tiles (Mantsu), ranging from one to nine, four tiles each; 36 circle tiles (Pintsu), ranging from one to nine, four tiles each; 36 bamboo tiles (Soutsu), ranging from one to nine, four tiles each. These three categories are usually called numerical tiles because they do have numerical values with them. 28 wind tiles (Jihai), consists east(ton), west (sha), south (nan), north (pei), red (chon), green (hatsu), white (haku), four tiles each. Some game will include four red tiles of 5’s (two circle 5, one character 5, one bamboo 5) replacing normal 5’s as permanent bonus tiles. Please check “bonus tiles” in “scoring pattern” section for more information.

2. Point counter

Finger-long sticks made from plastic or wood act like chips in gambling. Unless specified, each player will have 25,000 points in representation of one 10,000 point stick, one 5,000 point stick, nine 1,000 point sticks and ten 100 point sticks. The trade, exchange and count is usually done manually. However, some newer models of automatic mahjong tables have microchip-installed counters, so the counting task can be done automatically.

3. Dice

Two six-sided dice will be used in the game. Their main function is to decide the position where players start to draw tiles. Usually considered as an anti-cheating process.

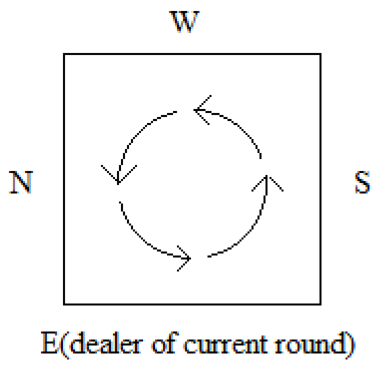
E. Start a game

1. Players

A normal game will need four players to start. There are special rules for 3 player games but it will not be the main discussion in this article.

2. Decide seat and draw order

In an old fashioned method, the seating order was decided in a very complicated way. Nowadays, the majority of games will decide order as following: get four tiles from the bank, east, west, south and north, one of each. Players will choose one freely from them. The player with the tile "east" will automatically become the dealer for first round. The rest of the players will sit counter-clockwise according to the dealer as the order east-south-west-north. Also, the "self wind" will be the same as players' seating directions. The drawing order is exactly the same as the seating order.



3. Building the bank of the tiles

Shuffle the tiles, and put them as top-down pairs with the backside faced-up. There will be 17 pairs in front of each player.

4. Cycles and rounds

A game is usually one cycle (ton-fuu sen, east-wind game) or two cycles (ton-nan sen, east-south game). Each player will take a turn as dealer one or two times based on the game length.

Two main parts are used to represent the game process: cycle status (kyoku) and game status (bonba). For example, the very first round of game will be represented as below:

East cycle game 1 - round 0
cycle status game status

If the dealer wins or the dealer has a winning hand when the round draws, the dealer player remains unchanged and the game status will increase by one. If anyone other than the dealer wins or the dealer doesn't have a winning hand when the round end in a draw, the cycle status will increase by one. For example, assuming game is now on the east cycle game 3 round 0.

If the dealer is unchanged:

East cycle game 3 round 0 → East cycle game 3 round 1

If the dealer is changed:

East cycle game 3 round 0 → East cycle game 4 round 0

Usually the dealer will put some 100 point counters which number equals to the current game status as an indicator. Whomever wins the round will receive a modification of game status multiply 300 points in addition to their primitive score for the winning hand. The cycle status will also determine “cycle wind”. For east cycle, the cycle wind is east tiles, a triple of east tile will worth one multiplier. Similarly south cycle will have south tiles as cycle wind. Please check the “score calculation” section for more information of how the cycle status and game status will affect scoring.

5. Start drawing tiles

Roll the dice and draw tiles clockwise. The starting drawing place will be decided as:

E mod 4(side) + E (tile offset on the side selected), where E is the sum of the two dice

1 is always representing the dealer for this round. Accordingly, 2 for south, 3 for west and 0 for north. For example: 6 as a result of dice rolling sum. 6 mod 4 equals 2, which will be representing the south player. And again, the sum 6 also indicating start drawing at the 6th pair of tiles in front of the south player.

At the initialization, players will draw 4 tiles (2 pair) at the same time in the order of seating, repeat the process 3 times so each player will have 12 tiles in their hand. Then draw the last one tile in order. Usually the dealer will draw the first and fifth tiles together since he will be the first one to draw and there is nothing to stop him doing so. However, in some computer game, for the ease of design, the dealer will draw after every player gets their last tile, which will not make any difference in the game result.

After all players finish initializing their starting hand, the top tile of the 3rd pair counting reversely from the end of bank will be revealed by the player who sits nearest to it to indicate the bonus tile of this round. The bonus tile will be the next succeeding tile of the indicator tile. For example: indicator tile bamboo 5, actual bonus tile will be bamboo 6. If the indicator is 9 in any category, use 1 in same category for the actual bonus tile. For wind tile, two indicating cycle will be used: east-south-west-north-east, white-green-red-white. Also, the last seven pairs of tiles in the bank will be isolated from the main bank, these tiles will be called “ou-hai (king tiles)”. They are not supposed to be drawn except “kan” special move can result in drawing from the tail. If player draws from them by a “kan” move, resupply king tiles to 14 tiles from the tail of main bank.

6. Game under process

Each player will first draw a tile, insert that tile to their hand, then discard a tile. A players will always have 14 tiles after drawing, or 3x+2 if they have done a “special move”.

7. Special moves

Not limited from drawing from the bank, a player can take the discarded tile of the current player in turn. If the next player draws a tile, he loses the chance to declare that special move.

There are five kinds of special moves: chii, pon, kan, ron and tsumo.

Chii: take the discarded tile from the player sits to your left and build a straight of 3 tiles, put them as open hand, then discard a tile. If other players declare pon/kan on the same discarded tile, this “chii” move is negated.

Pon: Take the discarded tile from any player and build a triple of 3 tiles, put them as open hand, then discard a tile. Can negate other players’ “chii” move.

Kan: Take the discarded tile from any player and build a quad of 4 tiles, put them as open hand, draw a tile from the end of the bank, reveal another bonus indicating tile next to the existing bonus indicating tile, then discard a tile. Could also happen if a player draws all four tiles himself, he can do exactly the same process but only show two out of four tiles instead , and it doesn't count as the tiles in opened hand(such as the tiles used in a 3-tile group from a "chi" move. Can negate other players' "chii" move.

Ron: Take the discarded tile from any player and declare winning, put all tiles as open hand. Negates any other special moves.

Tsumo: After drawing, player declares winning and puts all tiles as open hand. Since it happens after the current players drawing phase, no other special moves can be made.

F. Riichi

Could be treated as another kind of special move that doesn't involve other players. While Player doesn't have any opened hand, he could declare "one step to win" by discarding the tile horizontally instead vertically. If no one else declares "ron" on this tile, player pays 1,000 points to actualize this "riichi", these points will be given to the player who wins the round. If this round ends in a draw, the points will be taken into next round. After declaring "riichi", the player's hand is locked, he cannot replace any tile other than the current drawing tile. Under "riichi" status, player can only declare "ron" or "tsumo" till end the round, or "kan" if he happens to draw another tile to make a quad from a triple he already had, otherwise, he has to discard any tile he draws immediately.

The advantages of "riichi" are:

“riichi” itself is worth one multiplier, which clears the requirement of minimum one multiplier needed to declare winning. Of course, it contributes to a higher point earning from this extra multiplier.

If a player wins the round, not only the current bonus tiles, but the other tiles which pair up with the current bonus indicating tiles(the tiles under current bonus indicating tiles) will also count as new bonus indicating tiles, which doubles the number of bonus tiles, and may contributes to a higher possible points earning.

Warning other players by the declaration. Player with a cheap hand may turn to defense so that player will no longer be a competitor in this round.

The disadvantages of “riichi” are:

Player loses the ability to change the hand to a higher valued one.

Player loses the ability to turn to defense.

Other players become aware. It is less likely other players will discard the tile which needed to win.

G. Declare winning

A winning hand should satisfies the conditions below:

A regular winning hand consists of a pair as “head” (atama), and four groups of three tiles whether they are in open hand or not. These groups of tiles could be either a straight or a triple. A quad can also count as a group of three. There are special patterns that don’t meet this requirement but still considered as a winning hand, please check “scoring patterns” for more information.

The hand should have at least 1 multiplier to declare winning without counting the “bonus tile”. For the multipliers, please see “scoring patterns” section.

While a player’s last tile needed for winning includes any tile he has discarded earlier, he cannot perform “ron” special move to use other players’ discarded tile as his last tile. However, when he gets the last tile from his own drawing, he can still declare winning.

If a player refuses to declare winning when any other player discards a tile he needs, he cannot declare winning by use “ron” special move until next time he discards a tile.

By the rule decided before the game, multiple players may or may not declare winning by use “ron” special move to a tile at the same time. If not, only the first player with the drawing order closest to the player who discarded the tile will get the points.

H. Point calculation

There are two main parts in point calculation, base and multiplier.

1. Multiplier

Simply adding all cumulative multipliers (some hand may fulfill multiple patterns but only some of them are counted. Please see “patterns” section in the appendix for more information) from a winning hand, including patterns and bonus tiles. Please check “scoring patterns” section for detail.

2. Base

Every winning hand has a minimum base of 20. Then adding base value based on the tables below, any other situation is not in the table is counted as 0 base point:

Three tile group base value		
	Normal tiles	1,9,or wind tiles
Open hand triple	2	4
Close hand triple	4	8
Open hand quad	8	16
Close hand quad	16	32

Winning tile base value	
Endpoint such as: 8 and 9 in hand,7 needed	2
Middle clip such as: 5 and 7 in hand, 6 needed	2
Head tile such as: 1 in hand, another 1 needed	2

Head tile base value	
Green, white and red	2
Cycle wind tile	2
Self wind tile	2

*Cycle wind and self wind could be same, in that case it will count as 4 point base value.

After all possible base points are counted, perform an upper ceiling operation to its nearest tenth. The calculation for base point is now finished.

The score calculation under 2000 points is generally done by the formula below:

$$Basic\ Score = base * 2^{multiplier+2}$$

After the score reachs 2000 points, the calculation will be simplified by only using the multiplier:

Multiplier less than 6	2000
Multiplier 6~7	3000
Multiplier 8~10	4000
Multiplier 11~12	6000

Multiplier 13+	8000

The point exchange between players is calculated by:

Dealer winning	
Dealer gets the tile himself	All other players pays: basic score * 2
Discard a tile that dealer needs	That player pays by himself: basic score * 6

Non-dealer winning	
Winner gets the tile himself	Dealer pays: basic score *2 Other player pays: basic score
Discard a tile that winner needs	That player pays by himself: Basic score * 4

At same time, game status * 300 will be added to the final score; if player gets the tile he needed himself, these extra points will be equally distributed among other players. Also, winner for the round will take away all credit left in the public poll.

I. Game ends

When the game is in the south cycle game 4, if the dealer of that game doesn't win or hold a ready hand at the moment of drawing, the whole game end and enters ranking phase. Players will rank from 1st to 4th based on how many points they are holding. Under some rules players will receive extra bonus/penalty based on the ranking. The most common rules are "10-20" rule and "5-10" rule. "10-20" means the rank 3rd player pays extra 10,000 points to the rank 2nd player, and rank 4th player pays extra 20,000 points to the rank 1st player. Similar as "5-10" rule.

J. Scoring patterns

Note: most of the patterns in this section have a figure for better understanding, will include them in the final draft.

Some of the patterns are limited to all closed hand only. The patterns below are available no matter the hand contains any opened hand or not unless specified.

Multiplier 1:

Riichi: Successfully performing “riichi” special move.

Ippatsu: After successfully performed “riichi”, perform “ron” special move on any other players’ discard tile before next discard of player’s own, or “tsumo” on next player’s own draw.

Menzen-tsumo: Perform “tsumo“, without any opened hand.

Tan-yao: Winning hand doesn’t contain any numerical tile with its numerical value 1 or 9, or wind tile.

Pin-hu: All the three-tile groups are straights, the winning tile is at an end point of one of them (such as holding 3 and 4, 2 is the winning tile). Player cannot have any opened hand.

Ii-pei-kou: Have two sets of identical straight of same category. Player cannot have any opened hand.

Yaku-hai: Have a triple or a quad of self-wind tiles, cycle-wind tiles, green, red or white.

Dora: Each bonus tile count as an additional multiplier, player have to clear the requirement of "1 multiplier minimum" to get a winning hand in order to apply these additional multipliers.

Rin-shan: Get the winning tile from the tail of the bank by performing a "kan" special move.

Chan-kan: Only happens when one of the other players declares "kan" by adding the 4th tile to an existing opened triple. Player may declare "ron" on that tile.

Hai-tei: Perform "tsumo" by getting the last tile in the main bank.

Kou-tei: Perform "ron" on the last possible discarded tile from another player.

Multiplier 2:

San-shoku: have 3 straights in all three categories but with same numerical value. Only counts 1 multiplier if have any opened hand.

Ittsu: have 3 straight in same category that forms a long straight from 1 to 9, i.e. three straight of 123,456,789. Only counts 1 multiplier if have any opened hand.

Tui-tui: all three-tile groups are triples or quads.

San-an-kou: closed hand contains three triples or quads.

Chi-tui: hand contains seven different pairs, this is a special pattern that doesn't follow the general winning hand rule.

Chan-ta: all three-tile groups and the pair contain at least one of the following: tiles with numerical value 1 or 9, wind tiles.

Hon-rou: all three-tile groups and the pair contain only tiles with numerical value 1 or 9, and wind tiles. Could be in the pattern of “Chi-tui”

San-shoku-dou-kou: have three triples or quads in all three categories but with same numerical value.

San-kan-tsu: have three quads in hand, could be opened hand.

Shou-san-gen: have a pair and two triples or quads of the following: green, red, white.

Daburi(double riichi): declares “riichi” on the first discarding phase. If there is any other player performs a special move other than “riichi” itself before this player does so, it will be just count as regular “riichi”.

Multiplier 3:

Hon-ii-tsu: hand contains only tiles from one kind of numerical tiles, with the rest are wind tiles. Only count 2 multipliers if have any opened hand.

Jun-chan: all three-tile groups and the pair contain at least a tile of numerical value 1 or 9, different from “chan-ta”, no wind tile can be used.

Ryan-pei: hand contains two sets of “ii-pei-kou”. It is always in the pattern of “chii-tui”, but no double count. Only “ryan-pei” counts.

Multiplier 6:

Chin-ii-sou: hand contains only tiles from one kind of numerical tiles.

Multiplier 13(yakuman):

Some extremely rare patterns. Please check "Japanese mahjong yaku" on Wikipedia for more information.

K. Previous game record

http://tenhou.net/6/#json={%22title%22%3A[%22%E7%AC%AC%E4%BA%8C%E6%9C%9F%E3%80%80%E5%A4%A9%E9%B3%B3%E5%90%8D%E4%BA%BA%E6%88%A6%22,%22%E7%AC%AC%EF%BC%91%E7%AF%80%E3%80%80%EF%BC%A1%E5%8D%93%E3%80%80%EF%BC%91%E6%88%A6%E7%9B%AE%22],%22name%22%3A[%22%E2%93%88%E7%A6%8F%E5%9C%B0%E8%AA%A0%22,%22%E2%93%85%E5%A4%9A%E4%BA%95%E9%9A%86%E6%99%B4%22,%22C%E3%81%95%E3%82%93%22,%22%E2%93%85%E7%9F%B3%E6%A9%8B%E4%BC%B8%E6%B4%8B%22],%22rule%22%3A{%22disp%22%3A%22%E8%88%AC%E5%8D%97%E5%96%B0%E8%B5%A4%22,%22aka%22%3A1},%22log%22%3A[[[7,0,0],[38400,45300,25600,10700],[18],[39],[11,11,16,19,52,27,36,37,37,38,41,46,46],[41,%22p111111%22,24,14,19,38,%22c262452%22,22,44,16],[16,37,27,60,41,41,38,60,60,60],[13,13,15,19,21,21,29,34,53,36,37,38,44],[31,33,46,39,41,51,12,15,12],[29,44,13,15,53,13,51,60,34],[11,16,17,19,22,27,29,33,33,34,39,42,43],[14,25,43,37,13,26,42,45,14],[39,11,22,60,43,43,60,42,34],[11,15,17,18,25,26,26,27,27,31,31,32,35],[43,24,22,23,28,13,26,47,34,44],[60,11,32,35,%22r15%22,60,60,60,60,60],[%22%E5%92%8C%E4%BA%86%22,[-11600,0,0,12600],[3,0,3,%2230%E7%AC%A64%E9%A3%9C11600%E7%82%B9%22,%22%E7%AB%8B%E7%9B%B4%281%E9%A3%9C%29%22,%22%E5%B9%B3%E5%92%8C%281%E9%A3%9C%29%22,%22%E8%A3%8F%E3%83%89%E3%83%A9%282%E9%A3%9C%29%22]]]]}&ts=0

L. Source code address:

https://github.com/morisamashinka/mahjong