Method for Constructing Artificial Intelligence Player with Abstractions to Markov Decision Processes in Multiplayer Game of Mahjong

Moyuru Kurita* and Kunihito Hoki†
*Heroz Inc., †The University of Electro-Communications

***Abstract*—We propose a method for constructing artificial intelligence (AI) player of mahjong, which is a multiplayer imperfect information game. Since the size of the game tree is huge, constructing an expert-level AI player of mahjong is challenging. We define multiple Markov decision processes (MDPs) as abstractions of mahjong to construct effective search trees. We also introduce two methods of inferring state values of the mahjong using these MDPs. We evaluated the effectiveness of our method using gameplays vis-à-vis the current strongest AI player.**

***Index Terms*—Mahjong, Game Abstraction, Markov Decision Process, Retrograde Analysis, Value Function.**

I. Introduction

MAHJONG is a popular game in Asia and has been played over a hundred years with different rule sets according to the country or region. Most rule sets of mahjong share common properties that makes developing AI players challenging, e.g., the number of players is three or four (mostly four), size of the game tree is huge, size and number of information sets are large, and uncertainty strongly influences gameplay. Though the performance of AI players has exceeded human experts in most two-player perfect information games and some multiplayer imperfect information games, this has not been the case for mahjong.

We propose a method of constructing an AI mahjong player and demonstrate that its performance is better than current AI players. We abstract the game of mahjong and treat it as multiple Markov decision processes (MDPs). We considered the averaged behavioral strategies of a variety of experts to replace three of the four players with a chance player who makes decisions based on static probabilities[1]. We introduce four MDPs as abstractions of mahjong and formulate a value functions by using these MDPs. The action probabilities of the chance player acting on behalf of three players are inferred from game records of experts and the authors' experience. We also verified the performance of greedy players who always choose an action of the greatest value.

This paper is organized as follows. We explain the rules and features of mahjong in Sec. II. We review related research in Sec. III and explain the contributions of our research in Sec. IV. We briefly outline our method in Sec. V and give further details of it in Sec. VI. We discuss the performance evaluation of our method using gameplays vis-à-vis the existing current strongest AI player in Sec. VII.

This research is conducted to further develop the contents of research on AI player of mahjong released at a domestic conference [2], organized the theoretical framework of the method, added a new computer experiment, and summarized it newly.

Corresponding author: Moyuru Kurita (email: mkmjai1@gmail.com).

[1]A chance player is also known as "Nature" [1].

II. Basic Knowledge

A. Outline of MDP

MDPs have been widely used to model various sequential-decision-making problems under uncertainty. While each model represented as an MDP is quite simple, these models encompass a wide range of applications [3]. An extensive-form game of N players can also be regarded as an MDP of a player when the behavioral strategies of the other players are announced in advance to the player. This subsection provides the basic components of an MDP for satisfying certain properties (i.e., finite sets of states and actions, discrete time, finite horizon, terminal reward only, degenerate initial distribution, and stationary deterministic policies). For the sake of simplicity, the components are specialized to represent MDPs introduced in this paper. Readers interested in a general description of an MDP should consult the literature (e.g., see the textbook [3]).

An MDP is described using time step, finite state set, finite action set, probability that specifies one-step dynamics, and terminal reward or payoff that specifies optimality criteria. Time step $\tau \in \{0, 1, \ldots\}$ indexes the sequential decision-making points or the terminal point. The decision maker (hereafter, player) occupies a state $s \in S$ at every time step τ. The set S contains initial state s_{ini} and can be divided into a non-terminal state set S_{c} and terminal state set S_{e}. The player has to choose an action $a \in A(s)$ when the player is in $s \in S_{\text{c}}$ at τ and occupies next state $s' \in S$ with the conditional probability $P(s'|s, a)$ at $\tau + 1$. The player receives terminal reward $U(s)$ when he/she is in $s \in S_{\text{e}}$. The player is always in s_{ini} at time step 0 and the one-step dynamics specified by $P(\cdot|s, a)$ always brings the player from s_{ini} to a terminal state $s \in S_{\text{e}}$ at τ that does not exceed an upper bound of time step τ_{UB} for any sequence of actions without letting the player occupy the same state again.

A policy or strategy of an MDP specifies the decision-making rules of a player. Such a policy is a function π of a state, satisfies $\pi(s) \in A(s)$ for any $s \in S_{\mathrm{c}}$, and determines the probability $P^{\pi}(\omega)$ that the MDP generates a sample sequence $\omega = s_0 a_0 s_1 a_1 \cdots s_{T'}$ satisfying $0 \leq T' \leq \tau_{\mathrm{UB}}$, $s_0 = s_{\mathrm{ini}}$, $s_1, \ldots, s_{T'-1} \in S_{\mathrm{c}}$, $s_{T'} \in S_{\mathrm{e}}$, and $a_i = \pi(s_i)$ $(i = 0, \ldots, T' - 1)$. It also determines the expected terminal reward $V^{\pi}(s)$ when the player is in $s \in S_{\mathrm{c}}$. The expected value $V^{\pi}(s)$ is regarded as an evaluation of policy π and optimal policy π^* that satisfies $V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s)$ for any $s \in S_{\mathrm{c}}$. The action value $Q^{\pi}(s, a)$ when the player chooses $a \in A(s)$ in $s \in S_{\mathrm{c}}$ satisfies

$$Q^{\pi}(s,a) = \sum_{s' \in S_{\mathrm{e}}} U(s') P(s'|s,a) + \sum_{s' \in S_{\mathrm{c}}} V^{\pi}(s') P(s'|s,a). \quad (1)$$

When the player chooses $a = \pi(s)$ in $s \in S_{\mathrm{c}}$, $Q^{\pi}(s, a) = V^{\pi}(s)$ holds. We use the abbreviated superscript symbol "$*$" instead of "$\pi*$" as V^*. We even omit these superscripts from V if the policy is not necessarily optimal.

This paper categorizes actions of mahjong and MDPs and states of MDPs into several different types (see Appendix for detailed descriptions). This paper assumes that the state transition of an MDP according to probability $P(\cdot|s, a)$ is made by a chance player.

B. Rules of Mahjong

There are variations in the rules of mahjong, but this section outlines the most basic mahjong rules commonly used in Japan (see [4]). Mahjong is a game played by four people who use four sets of 34 tiles. These 34 tiles are different kinds, and the total number of tiles is 136. Each player starts with 25,000 points. One gameplay of mahjong is a sequence of multiple hands[2], and the points move from player to player by each hand. A standard method of earning points is to form a winning hand earlier than the other players. A typical winninng hand consists of four combinations of three tiles satisfying specific conditions (each combination is called a mentsu) and one pair of the same kind of tile. The final rank of each player is determined by the final points of the game.

Figure 1 shows an overhead view of the tiles during a gameplay and the three different locations of the tiles:

1) Each player's hand is invisible to the others. Immediately to the right of the bottom player's hand is the tile (invisible to the others) obtained from the shuffled drawing pile.
2) Part of the hand is disclosed by the left player. When a player takes a tile discarded by another player, the player discloses a mentsu containing the tile.
3) Each player's tile row is visible and arranged in the order of discard.

Next is a decision point at which the bottom player discards one of the 14 tiles at location (1) to the second row of location (3).

In addition to the four players, we consider a chance player who introduces contingency into the gameplay. The actions of the chance player are classified into the following two types.

[2]A hand also means a set of tiles owned by a player

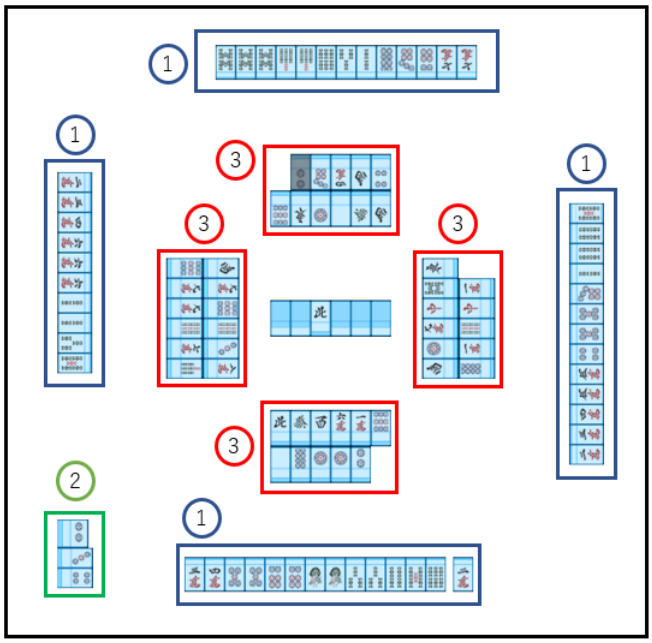

Fig. 1. Overhead view during gameplay.

- HANDDISTRIBUTION: The chance player distributes hands from the draw pile to each player. Each player receives a hand composed of 13 tiles, and one player receives an additional tile as a_{DRAW}. This player is called the parent player.
- DRAW: The chance player distributes one tile from the draw pile to a player, which is not revealed to the other players.

The information sets of each player are categorized into two types (type I and type II). Any information set of type I follows $a_{\mathrm{HANDDISTRIBUTION}}$ or a_{DRAW}. The player dealt a_{DRAW} is the player that chooses an action from one of the following action types.

- DRAWWIN: The player declares a win when his/her hand (13 + 1 tiles) satisfies specific conditions. Then the player discloses the hand and earns points depending on the hand from the other players. All players then discard their hands.
- DISCARD: The player discards a tile therefore, the size of the hand is kept at 13. The tile is now revealed to the others.

Type II follows a_{DISCARD} or $a_{\mathrm{TAKE\&DISCARD}}$ (explained below) of a player i who discarded a tile of kind h, where the other players sometimes gain the right to choose one of the following action types. Let j indicate a player who gained this right.

- TAKEWIN: Player j declares a win when his/her hand (13 tiles) and i's discarded tile of kind h (1 tile) satisfy specific conditions. Then j earns points from i, and all players discard their hands.
- TAKE&DISCARD: Player j assembles a mentsu using a discarded tile of kind h (take), discloses the mentsu, then discards another tile. Take behaviors are classified into a few classes such as pon (also known as pung)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TG.2020.3036471, IEEE Transactions on Games

and chi (also known as chow) depending on the mentsus assembled.

- PASS: Player j does not declare anything. If all players who gained the right pass, the next action is A_{Draw} of the player next to i.

These action types form the bulk of branching points in the hand. Each hand starts with $a_{\text{HANDDISTRIBUTION}}$, and the hand ends when one of the players chooses an action in A_{WIN} (a union of A_{DRAWWIN} and A_{TAKEWIN}), or when the number of tiles in the draw pile decreases to a specific number. Figure 2 illustrates some branches of the gameplay from a_{DISCARD} of player i to a_{DISCARD} of player $i+1$ (player $i=5$ means $i=1$). A hand consists of about 60 of such parts.

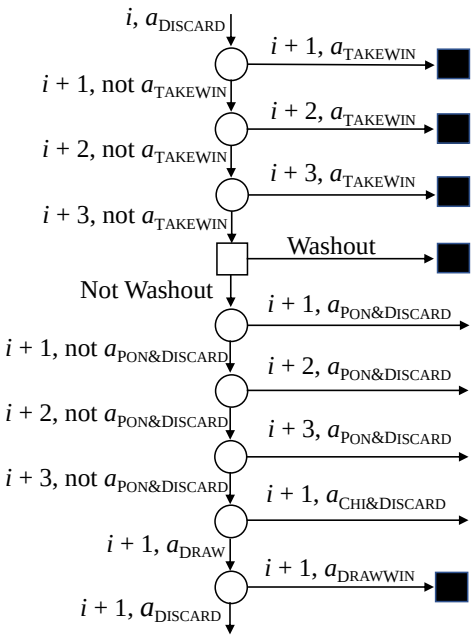

Fig. 2. Some branches of hand. White circles represent information sets in which players choose actions. Black squares represent sets of endings of hand. White square represents branch of washout determined by rules.

The rules specify whether the next hand starts or the entire game ends when a hand ends. The chance player determines one parent from four players for the first hand, and one parent for each subsequent hand is specified by the rules. If the rule called tonpu-match is applied, then a player usually plays four to six hands in a gameplay, and the player usually plays one or more hands as a parent.

We now describe several important terms in mahjong we use in this paper.

- tenpai: When a hand (13 tiles) can become a winning hand with an additional tile, the hand is called tenpai.
- shanten-number: The minimum number of tiles that need to be exchanged to make the hand tenpai.

C. Features of Mahjong

Mahjong's gameplay consists of playing multiple hands in a row. The game situation before $a_{\text{HANDDISTRIBUTION}}$ can be explained from only a small amount of shared information ϕ_{hand} (points of four players etc.). Also, four behavior strategies and the shared information determine the expected value of the final ranking of each player. Since it is possible to represent the game situation before $a_{\text{HANDDISTRIBUTION}}$ by ϕ_{hand} and obtain a sufficient number of expert's record, the final ranking in this game situation is easily predicted by regression. Therefore, it is reasonable to represent a hand as a truncated partial game. The game tree handles the end of the hand (i.e., the beginning of the next hand) as terminal nodes to which the expected values of the final rank are given. Similar methods of treating the entire game as a continuous truncated partial game are used in other games. For example, it is common to play one game of n-point-match backgammon as an individual game based on the reward of the match-equity table [5].

Let $P(\text{RANK}(x,i)|\phi_{\text{hand}})$ be the probability that event $\text{RANK}(x,i)$, i.e., player i acquires rank x for $i, x \in \{1,2,3,4\}$, occurs under the condition ϕ_{hand}. Then the expected value of i's payoff at ϕ_{hand} is given by

$$U^i_{\text{hand}}(\phi_{\text{hand}}) = \sum_{x\in\{1,2,3,4\}} P(\text{RANK}(x,i)|\phi_{\text{hand}}) U_{\text{game}}(x). \quad (2)$$

Here, $U_{\text{game}}(x)$ is the payoff of rank x, which is defined by the rules of the tournament (normally, the higher the rank, the higher the payoff).

We roughly estimate the number of information sets (i.e., decision points) of a player in a truncated partial game of one hand by ignoring actions in A_{TAKEWIN} and in $A_{\text{TAKE\&DISCARD}}$. $A_{\text{HANDDISTRIBUTION}}$ covers about 10^{11} ways to distribute hands to a player. The number of legal actions in A_{DISCARD} is about ten. After that, the player can see about 30 kinds of tiles at each a_{DISCARD} of the other three players. Then, the player can see about 30 kinds of tiles at a_{DRAW}. We call a partial gameplay from discard to next draw of the same player a turn. Since the number of turns to play one hand is about 20 at most, we obtain a rough estimate by

$$10^{11} \times (30^4 \times 10)^{20} \approx 10^{150}. \quad (3)$$

The exponent value is a little smaller than that of the state space in the game Go [6].

A hand falls into five scenarios from i's point of view.

1) win: i chooses an action in A_{WIN}. Usually this is the most favorable scenario.
2) lose: Another player chooses an action in A_{TAKEWIN} against a tile discarded by i. Usually this is the most unfavorable scenario.
3) other win: Another player chooses an action in A_{DRAWWIN} or in A_{TAKEWIN} against a tile discarded by another player different from i. It is difficult for i to control the emergence of this scenario as will.
4) tenpai washout: The hand ends due to a shortage in the draw pile when i has a tenpai hand.
5) noten washout: The hand ends due to a shortage in the draw pile when i does not have a tenpai hand.

We ignore other scenarios because they are rare. Choosing one of these scenarios according to the current game situation is one of the most important strategies for playing mahjong.

III. Previous Research

Due to research over the past 20 years, AI has exceeded human ability in many two-player zero-sum games with perfect information, e.g., backgammon [7], checkers [8], chess [9], shogi [10], [11], and Go [12]. One of the techniques that has played a central role in the development of these AI players is heuristic search using the property that two players share symmetric information [13]. However, heuristic search has not been powerful in games with three or more players and imperfect information. The reason for this is that it is difficult to construct a search tree that is easy to finish searching and effective for representing proper game situations.

There are also interesting research results from two-player games with imperfect information. Counterfactual regret minimization (CFR) is a powerful technique based on self-play for constructing a strong player of a game belonging to such a class [14]. In fact, ϵ-Nash equilibrium of heads-up limit Texas hold'em, which has about 10^{14} decision points for a player, was obtained using CFR$^+$, a variant of CFR [15]. Moreover, an expert-level AI player of heads-up no-limit Texas hold'em, which has more than 10^{160} decision points, has been developed using tree search with bet abstraction and deep learning of counterfactual values [16]. In research other than on poker AI, an expert-level AI of Scrabble has been developed using a selective move generator, simulations of likely game scenarios, and the heuristic search algorithm B^*[17].

Relatively few studies have been reported on multiplayer imperfect-information games such as mahjong. Even in such games, one of the research objectives may also be to compute approximations to some of the Nash equilibrium points. A case study on limit Texas hold'em with three players was conducted [18] in which an AI player based on CFR outperformed other AI players, although this method loses the theoretical guarantees of two-player zero-sum games. However, applying CFR variants to other multiplayer games is not easy. Implementation of a mahjong player based on CFR is difficult because the size of the game tree is too large to search, and the abstraction for reducing the search space is unknown.

Another research objective in multiplayer imperfect-information games is to construct an AI player by using heuristic methods, which are known to be effective in two-player perfect-information games. There are AI players in multiplayer Texas hold'em. Poki, which is an AI player of Texas hold'em involving multiple players, adopts a betting strategy based on heuristic evaluation of hand strength [19]. Commercial software called Snowie is considered to have the same strength as experts, but its algorithm is unpublished.

Besides poker games, an expert-level AI player of Skat has been constructed based on heuristic search algorithms of perfect-information games. Search algorithms have been used in the game using game-state inference and static evaluation obtained by regression using game records [20]. It is interesting to build AI players based on such heuristic search algorithms in other multiplayer games and imperfect information, but it is difficult to construct an effective search tree. In fact, it has been reported that an AI player of The Settlers of Catan applying Monte-Carlo tree-search methods is not as strong as human players [21].

There has been research on AI players of mahjong. There is an open-source beginner-level player based on the Monte-Carlo simulation called manue[3]. To model actions of opponent players statically, it uses inferred probabilities that an action in A_{DISCARDS} (a union of A_{DISCARD} and $A_{\text{TAKE\&DISCARDS}}$) by a player induces a win of another player. Bakuuchi is another player that carries out Monte-Carlo simulations. Early Bakuuchi uses such probabilities with higher accuracy, Eq. (2), to evaluate each simulation at the end of the hand and simulation policies learned from game records [22]. In that study, they reported that point dependency on the policy is inappropriate and had reached only the intermediate level. Note that a recent Bakuuchi, which is unpublished, has reached the advanced level. Bakuuchi is the first AI player that achieved 7 dan in tenhou. It highest dan achieved is 9 dan, and it is stronger than a majority of human players. To the best of our knowledge, no tree has yet been discovered to search for better decisions.

Our method abstracts mahjong to construct effective search trees to appropriately handle various game situations. Game abstraction is known as an effective means of reducing a huge search space of an extensive-form game with imperfect information [23]. For example, the effectiveness of information and action abstraction is shown in the aforementioned poker and patrolling security games [24].

IV. Contributions

The contributions of this paper are as follows.

(1) We define an abstraction of mahjong, *Inclusive Policy Solitary Mahjong* $\mathcal{M}$, which is an MDP that is expected to be effective to evaluate short-term behavior strategies to compete on the most favorable scenario win. Three other players are replaced with a static environment, and the decision-making player goes through the process of $\mathcal{M}$ and ends with a win, lose, other win, tenpai washout, or noten washout scenario.

(2) We introduce several features in machine learning that are expected to be representative of long-term behavior strategies of a hand and be useful for inferring state values. The features are computed using three other MDPs. Three other players are replaced with a static environment, and the decision-making player goes through each process and ends with a few specific scenarios.

(3) We propose a method for constructing an AI player using (1) and (2). We present the experimental results of 3557 gameplays with a state-of-the-art AI mahjong player in which our AI player achieved a significantly higher average rank. We also present that our player makes each decision in a few seconds using a realistic computational resource.

V. Outline of Proposed Method

We discuss the action values of mahjong by separating the cases in which a hand ends immediately. Let us consider the first few actions from information set u_0. Recall that most actions belong to three types, Win, Discards, and Pass.

[3]Hiroshi Ichikawa https://github.com/gimite/mjai-manue

We first consider action type WIN. After such an action, a, a hand ends without any action of the other players. When player i at u_0 takes a, the action value is

$$Q^{\text{org}}(u_0, a) = U^i_{\text{hand}}(\phi_{\text{hand}}). \tag{4}$$

The superscript 'org' means original game of mahjong. We compute $U^i_{\text{hand}}(\phi_{\text{hand}})$ using Eq. (2), where $P(\text{RANK}(x, i)|\phi_{\text{hand}})$ is inferred using a multi-class logistic regression model, as in a previous study [22].

Next, we consider action type DISCARDS. Such an action, a, is accompanied by discarding a tile, and the hand also ends immediately if another player chooses a_{TAKEWIN} against the tile. Let us assume that the other players determine actions according to static probability and treat them as if they are also the chance player. When i at u_0 takes a, we approximate the action value as

$$\begin{aligned} Q^{\text{org}}(u_0, a) \quad &\approx \quad P(\text{TAKEWIN from } i|u_0, a) \\ &\times \quad U_{\text{TAKEWIN from } i}(u_0, a) \\ &+ \quad P(\overline{\text{TAKEWIN from } i}|u_0, a) \\ &\times \quad U_{\overline{\text{TAKEWIN from } i}}(u_0, a). \end{aligned} \tag{5}$$

The probability $P(\text{TAKEWIN from } i|u_0, a)$ and corresponding expected payoff $U_{\text{TAKEWIN from } i}(u_0, a)$ can be inferred using orthodox machine learning methods because a hand immediately terminates if a is followed by the a_{TAKEWIN} of another player. We discuss these methods in Sec. VI-C.

We then consider action type PASS, i.e., $a \in A_{\text{PASS}}$. When i at u_0 takes such a, we approximate the action value as

$$Q^{\text{org}}(u_0, a) \approx U_{\text{PASS}}(u_0, a). \tag{6}$$

The value $U_{\text{PASS}}(u_0, a)$ is the corresponding expected payoff.

After separating the cases of immediate ends of a hand, we need to compute $U_{\overline{\text{TAKEWIN from } i}}(u_0, a)$ and $U_{\text{PASS}}(u_0, a)$ to estimate the action value at u_0. Our method uses two models to compute these values. In both models, three other players are considered as a chance player. These models treat the abstract game state s, which is an approximate expression of an i's information set that will be realized after (u_0, a), as tuple $s = (u_0, q, h, c, t)$. Here, q represents a future hand envisioned at u_0 of i, h is a kind of tile that i will encounter in the future, c is a type of state described below, and t is the number of tiles discarded by i since u_0. The tile of type h is one of the most important judgment factors when a mahjong player makes a decision. When game state s is an abstract representation of an information set of type I (see Sec. II-B), the tile is the one distributed by the chance player just before making the decision. On the other hand, when s is that of type II, the tile is the one discarded by another player just before making the decision. In this way, these models explicitly keep track of i's hand and one most recently raised tile, and any other features in the game situation (u_0, a) are assumed to be almost unchanged. We omit u_0 of s in Sec. VI-A and VI-B because it is constant and appears too many times.

To set up the first model, in addition to using the state representation, we define inclusive policy one-player mahjong $\mathcal{M}$, which is an MDP and takes into account as many scenarios as possible. This MDP requires comprehensive search and designed to predict a hand's ending with a relatively small number of steps with high accuracy.

To set up the second model, in addition to using the state representation, we define several one-player-mahjong games, which are different MDPs, and take into account different small subsets of all scenarios. The estimation of action values by one of these one-player mahjong games is not accurate because each subset is restricted. However, these one-player-mahjong games are amenable to long-term computation and can be used to provide good features to predict the scenario of hands with a relatively large number of steps.

VI. PROPOSED METHOD

This chapter is organized as follows. In Sec. VI-A, we define multiple MDPs as mahjong abstractions and formulate their action-value functions. We then represent the action values of game situation by means of these MDPs in Sec. VI-B. In Sec. VI-C, we describe methods of calculating probability, payoff, and other parameters of the MDPs. The probability and payoff functions are summarized in the Appendix. In Sec. VI-D, we describe an efficient search algorithm of the MDPs.

A. Abstraction to MDPs

Consider player i at information set u_0 of a hand, which is a truncated partial game of mahjong. We abstract the hand rooted at u_0 to an MDP in four ways. Here, i is the agent who makes decisions, and decision making of the others are represented by transitions probabilities of states. This section defines four MDPs and formulas that approximately represent the expected value of i's payoff.

1) Inclusive Policy Solitary Mahjong $\mathcal{M}$: MDP $\mathcal{M}$ covers various scenarios from i's point of view. The flow of $\mathcal{M}$ is schematically shown in Fig. 3. Type c of state s in $\mathcal{M}$ indicates one of the following:

- `Discard`: Player i at s of this type can choose a_{DRAWWIN} to gain payoff $U_{\text{DrawWin}}(q, h)$ only if q and h satisfy conditions used in the mahjong rule sets as if q and h respectively were a hand (13 tiles) and a kind of tile obtained from the drawing pile. If i does not, then i has to choose an action in A_{DISCARD} to discard a tile from q, or discard a tile of kind h.
- `Take`: Player i at s of this type can choose a_{TAKEWIN} to gain payoff $U_{\text{TakeWin}}(q, h)$ or an action in $A_{\text{Take\&Discard}}$ only if q and h satisfy conditions used in the mahjong rule sets as if q and h respectively were a hand (13 tiles) and a kind of tile discarded by the other players. If i does not, then i chooses a_{PASS}.
- `Fold`: Player i at s of this type chooses either a_{FOLD} or a_{NOTFOLD}. If i chooses a_{FOLD}, i gains payoff $U_{\text{Fold}}(q, t)$.

`Discard` and `Take` respectively correspond to i's information sets following a_{DRAW} of the chance player and a_{DISCARD} of the other players in mahjong. Though an information set corresponding to `Fold` does not exist in mahjong, we introduce this type of state for simplification.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TG.2020.3036471, IEEE Transactions on Games

MDP $\mathcal{M}$ terminates immediately if i chooses either a_{DRAWWIN}, a_{TAKEWIN}, or a_{FOLD}. Otherwise, the chance player chooses actions, which are categorized as follows.

- LOSE: $\mathcal{M}$ terminates at probability $P(a_{\text{LOSE}}|q,h,t)$ after i's action of A_{DISCARDS}, where h is a kind of tile discarded by the action and i gains payoff $U_{\text{Lose}}(h)$. If $\mathcal{M}$ does not terminate, t increases by one. Then $\mathcal{M}$ terminates if $t = T$, and i gains payoff $U_{\text{Washout}}(q)$. Otherwise, the state transfers to a state of `Fold` (T is a parameter of $\mathcal{M}$, which is described in Sec. VI-C).
- OTHERWIN: $\mathcal{M}$ terminates at probability $P(a_{\text{OTHERWIN}}|q,t)$ after i chooses a_{NOTFOLD} and gains payoff U_{OtherWin}. Otherwise, the chance player chooses an action in $A_{\text{OTHERDISCARD}}$.
- OTHERDISCARD: The chance player chooses a tile of kind h at probability $P_{\text{D}}(h|q)$ and the state transfers to a state of `Take`.
- DRAW: The chance player deals a tile of kind h at probability $P_{\text{D}}(h|q)$ as if h were served from the shuffled drawn pile after i chooses a_{PASS} at a state of `Take`, and the state transfers to a state of `Discard`.

a_{LOSE} and a_{OTHERWIN} respectively result in a transition to a terminal state corresponding to lose and other win scenarios in the mahjong, whereas a_{DRAWWIN} and a_{TAKEWIN} result in a transition to a terminal state corresponding to a win scenario. The a_{FOLD} results in a transition to a terminal state that deals with lose and other win scenarios using $\mathcal{M}_{\text{fold}}$, as described below.

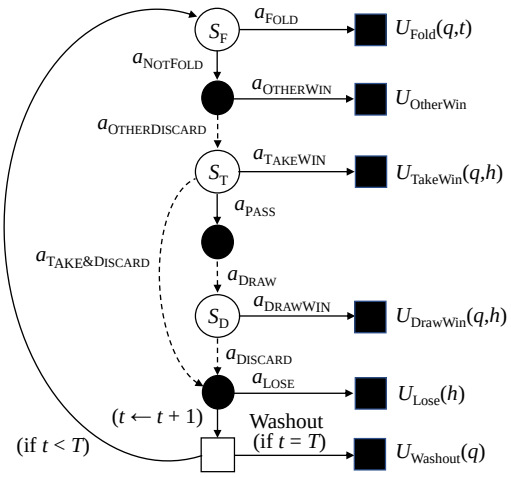

Fig. 3. Some branches of $\mathcal{M}$. White circles represent states in which player i chooses actions. Black circles are also branching points, neither of which is decision point of player i. Black squares represent sets of endings of $\mathcal{M}$. White square represents branch of washout determined by rules.

When the c of s is `Discard`, i.e., $s = (q, h, \texttt{Discard}, t)$ and h corresponds to a kind of tile obtained from the drawing pile, the action-value function is as follows. For (q,h) satisfying the conditions to choose a_{DRAWWIN}, we have $Q(s, a_{\text{DRAWWIN}}) = U_{\text{DrawWin}}(q,h)$. MDP $\mathcal{M}$ terminates with this action. For action a in A_{DISCARD}, we have a tile of kind h' to be discarded through a, a new hand q' after discarding a tile of kind h' from q or the tile obtained from the drawing pile of kind h, and

$$\begin{aligned} Q(s,a) &= P(a_{\text{LOSE}}|q',h',t)U_{\text{Lose}}(h') \\ &+ P(\overline{a_{\text{LOSE}}}|q',h',t)V_{\overline{\text{Lose}}}(q',h',t+1). \\ V_{\overline{\text{Lose}}}(q',h',t+1) &= \begin{cases} V(s'_{\text{F}}) & t+1 < T \\ U_{\text{Washout}}(q') & t+1 = T \end{cases} \end{aligned} \tag{7}$$

Here, $s'_{\text{F}} = (q', \text{null}, \texttt{Fold}, t+1)$.

When the c of s is `Fold`, i.e., $s = (q, \text{null}, \texttt{Fold}, t)$, and the action is a_{NOTFOLD}, we have

$$\begin{aligned} Q(s, a_{\text{NOTFOLD}}) &= P(a_{\text{OTHERWIN}}|q,t)U_{\text{OtherWin}} \\ &+ P(\overline{a_{\text{OTHERWIN}}}|q,t)\sum_{h\in H} P_{\text{T}}(h|q)V(s_{\text{T}}), \end{aligned} \tag{8}$$

where $s_{\text{T}} = (q, h, \texttt{Take}, t)$, h is the kind of tile that is analogous to the other players' discard in mahjong, and H is the set of all tile kinds.

When the c is `Take`, i.e., $s = (q, h, \texttt{Take}, t)$, the action-value function is as follows. For (q,h) satisfying the condition to choose a_{TAKEWIN}, we have $Q(s, a_{\text{TAKEWIN}}) = U_{\text{TakeWin}}(q,h)$. MDP $\mathcal{M}$ terminates with this action. For action a_{PASS}, we have

$$Q(s, a_{\text{PASS}}) = \sum_{h'\in H} P_{\text{D}}(h'|q)V(s'_{\text{D}}), \tag{9}$$

where $s'_{\text{D}} = (q, h', \texttt{Discard}, t)$. For action a in $A_{\text{TAKE\&DISCARD}}$, we have

$$\begin{aligned} Q(s,a) &= P(a_{\text{LOSE}}|q',h',t)U_{\text{Lose}}(h') \\ &+ P(\overline{a_{\text{LOSE}}}|q',h',t)V_{\overline{\text{Lose}}}(q',h',t+1), \end{aligned} \tag{10}$$

where h' is a kind of tile discarded through a. The formulas of payoff functions are outlined in Sec. VI-C.

2) Folding Solitary Mahjong ($\mathcal{M}_{fold}$): MDP $\mathcal{M}_{\text{fold}}$ covers two scenarios (lose and other win) to represent the folding strategies of i. Folding is a behavior strategy in which i abandons the most favorable win scenario and avoids the most unfavorable lose scenario of the current hand. All Actions of DRAW and DISCARDS of the other players are ignored to simplify the game.

In $\mathcal{M}_{\text{fold}}$, under the probability of lose $P_{\text{Lose}}(h)$ and payoff of lose $U_{\text{Lose}}(h)$, i is only allowed to discard h from i's hand. The action type of i is only DISCARD, any state in $\mathcal{M}_{\text{fold}}$ is of type `Discard`, and the number of tiles in i's hand decreases monotonically from the initial state because actions of DRAW are ignored. There are two types of actions of the chance player, which represents an average expert player, as follows.

- LOSE: $\mathcal{M}_{\text{Fold}}$ terminates with probability $P_{\text{Lose}}(h)$, where h is the kind of tile discarded just before, and i gains payoff $U_{\text{Lose}}(h)$. If i discarded tiles of the kind h more than once from the initial state, this type is not selected.
- OTHERWIN: When $\mathcal{M}_{\text{fold}}$ does not terminate by a_{LOSE}, it terminates with constant probability α (we tentatively set $\alpha = 0.1$), and i gains payoff U_{OtherWin}. Otherwise, the state transfers to another state of type `Discard`.

MDP $\mathcal{M}_{\text{fold}}$ also terminates when i discards all or T_{fold} tiles and i gains payoff U_{OtherWin}. For the sake of simplicity, we

assume a natural condition $U_{\text{Lose}}(h) < U_{\text{OtherWin}}$ holds for all h. Under this assumption, if i has tiles of a certain kind that have been discarded once or more, i should always discard one of these tiles.

Let q be i's hand, h be the kind of tile to discard, and t be the number of discards from the initial state. Action-value functions are formulated in terms of s = (q, t) without specifying both c (the constant state type `Discard`), and h (always the kind discarded through an action in A_{DISCARD}). When action a is an action that discards h, we have

$$\begin{aligned} Q(q,t,a) &= \begin{cases} Q_{\text{new}}(q,t,a) & \text{if } a \text{ discards a new kind } h \\ Q_{\text{prev}}(q,t,a) & \text{otherwise} \end{cases} \\ Q_{\text{new}}(q,t,a) &= P_{\text{Lose}}(h)U_{\text{Lose}}(h) \\ &+ (1-P_{\text{Lose}}(h))\alpha U_{\text{OtherWin}} \\ &+ (1-P_{\text{Lose}}(h))\bar{\alpha}V(q',t+1) \\ Q_{\text{prev}}(q,t,a) &= \alpha U_{\text{OtherWin}} + \bar{\alpha}V(q',t+1). \end{aligned} \tag{11}$$

Here, $\bar{\alpha} = 1-\alpha$, and q' is a hand where one of tiles of kind h is subtracted from q.

The optimal policy is to discard the tiles in ascending order of $f(h)$, i.e.,

$$f(h) = \frac{P_{\text{Lose}}(h)(U_{\text{OtherWin}} - U_{\text{Lose}}(h))}{1-(1-P_{\text{Lose}}(h))\bar{\alpha}^{n_h}}, \tag{12}$$

where n_h is the number of tiles of kind h in i's hand in the initial state, and the optimal value $E_{\text{Fold}} = V^*(s_{\text{ini}})$ is given by

$$\begin{aligned} E_{\text{Fold}} &= P_{\text{Lose}}(h_1)U_{\text{Lose}}(h_1) \\ &+ (1-P_{\text{Lose}}(h_1))\left[1-\bar{\alpha}^{n_{h_1}}\right]U_{\text{OtherWin}} \\ &+ (1-P_{\text{Lose}}(h_1))\bar{\alpha}^{n_{h_1}}P_{\text{Lose}}(h_2)U_{\text{Lose}}(h_2) \\ &+ \cdots \\ &+ \left[\prod_{k=1}^{K}(1-P_{\text{Lose}}(h_k))\right]\bar{\alpha}^{\sum_k n_{h_k}}U_{\text{OtherWin}}. \end{aligned} \tag{13}$$

Here, h_k $(k = 1, \cdots, K)$ is in ascending order of Eq. (12), and K is the number of tile kinds in i's hand in the initial state. The optimal policy ends up with the scenario lose with the probability

$$\begin{aligned} P_{\text{FoldLose}} &= P_{\text{Lose}}(h_1) \\ &+ (1-P_{\text{Lose}}(h_1))\bar{\alpha}^{n_{h_1}}P_{\text{Lose}}(h_2) + \cdots. \end{aligned} \tag{14}$$

Let $U_{\text{LoseAverage}}$ be the optimal value under the condition of lose termination, which is discussed in Sec. VI-B. Equation (13) can be transformed using $U_{\text{LoseAverage}}$ as follows

$$\begin{aligned} E_{\text{Fold}} &= P_{\text{FoldLose}}U_{\text{LoseAverage}} \\ &+ (1-P_{\text{FoldLose}})U_{\text{OtherWin}}. \end{aligned} \tag{15}$$

3) Winning Solitary Mahjong ($\mathcal{M}_{win}$) and Tenpai Solitary Mahjong ($\mathcal{M}_{tenpai}$): MDPs $\mathcal{M}_{\text{win}}$ and $\mathcal{M}_{\text{tenpai}}$ are specialized for representing win and tenpai strategies, respectively. Both are expected to have a smaller search space than $\mathcal{M}$. Terminal states that do not have direct relations to the purpose of win for $\mathcal{M}_{\text{win}}$ or tenpai for $\mathcal{M}_{\text{tenpai}}$ are ignored. Specifically, terminal states induced by a_{LOSE}, a_{FOLD}, and a_{OTHERWIN} are removed from $\mathcal{M}$ in both MDPs. Moreover, the payoff of washout $U_{\text{Washout}}(q)$ in $\mathcal{M}_{\text{win}}$ does not depend on q, which we write as U_{NotWin}. Also, player i of $\mathcal{M}_{\text{tenpai}}$ is unable to take an action in A_{WIN}. We omit formulas of action values, but they are derived by replacing zero with probabilities corresponding to those actions.

B. Value Inference Using Multiple MDPs

In this section, we introduce two methods of inferring values of legal actions of mahjong using multiple MDPs introduced in the previous section. The first method simply adopts the optimal value V^* of $\mathcal{M}$ to calculate the approximate values in Eqs. (5) and (6) as

$$\begin{aligned} U_{\overline{\text{TakeWin from } i}}(u_0,a) &= V^*(q,\text{null},\texttt{Fold},1) \\ U_{\text{Pass}}(u_0,a) &= V^*(q,\text{null},\texttt{Fold},0) \end{aligned} \tag{16}$$

where u_0 is player i's information set and q is i's hand after action a. Note that parameter t indicates the number of tiles player i discarded from i's information set u_0.

The second method uses the results of optimal-policy evaluations using $\mathcal{M}_{\text{win}}$, $\mathcal{M}_{\text{tenpai}}$, and $\mathcal{M}_{\text{fold}}$. Let Z be a set of hand scenarios $\{\text{win}, \text{lose}, \text{other}, \text{tenpai}, \text{noten}\}$. This method calculates the approximate values in Eqs. (5) and (6) as

$$\begin{aligned} U_{\overline{\text{TakeWin from } i}}(u_0,a) &= V(q,1) \\ U_{\text{Pass}}(u_0,a) &= V(q,0) \\ V(q,t) &= \sum_{z\in Z} P(q,t,z)U(q,t,z) \end{aligned} \tag{17}$$

We calculate $P(q,t,z)$ in Eq. (17) using the product of probabilities p_{win}, p_{washout}, p_{tenpai}, and p_{lose} (see Table I) as

$$\begin{aligned} P(q,t,\text{win}) &= p_{\text{win}}(q,t) \\ P(q,t,\text{tenpai}) &= p_{\overline{\text{win}}}(q,t)p_{\text{washout}}p_{\text{tenpai}}(q,t) \\ P(q,t,\text{noten}) &= p_{\overline{\text{win}}}(q,t)p_{\text{washout}}p_{\overline{\text{tenpai}}}(q,t) \\ P(q,t,\text{lose}) &= p_{\overline{\text{win}}}(q,t)p_{\overline{\text{washout}}}p_{\text{lose}}(q,t) \\ P(q,t,\text{other}) &= p_{\overline{\text{win}}}(q,t)p_{\overline{\text{washout}}}p_{\overline{\text{lose}}}(q,t) \end{aligned} \tag{18}$$

These probabilities are inferred by logistic regression using features that are the results obtained from optimal-policy evaluations of these MDPs and game records (see Sec. VI-C for more detail). To explain their features, let us introduce the following symbols: $V_{\text{win}}(q,t)$ and $P_{\text{win}}(q,t)$ are results obtained from optimal-policy evaluation of $\mathcal{M}_{\text{win}}$, where the former is an optimal state value of $(q, \text{null}, \texttt{Fold}, t)$ and the latter is the probability that i in this state finally chooses an action in A_{Wins}; $P_{\text{tenpai}}(q,t)$ is the probability that i in $(q, \text{null}, \texttt{Fold}, t)$ of $\mathcal{M}_{\text{tenpai}}$ will have a tenpai hand when it terminates under an optimal policy; and $P_{\text{FoldLose}}(q,t)$ and $U_{\text{LoseAverage}}(q,t)$ are values from Eqs. (14) and (15), where the initial hand of $\mathcal{M}_{\text{fold}}$ is q and T is adjusted according to t. The features used for the regressions are as follows.

- $p_{\text{win}}(q,t)$:
  - $\text{logit}(P_{\text{win}}(q,t))$
  - The number of players declaring riich (riich is discussed in Sec. VI-E).

  - $1-\prod_j(1-p^j_{\text{tenpai}})$. Here, j runs over all players who are not i and do not declare riich.
- p_{washout}:
  - The number of players declaring riich.
  - $1-\prod_j(1-p^j_{\text{tenpai}})$. Here, j runs over all players who are not i and do not declare riich.
- $p_{\text{tenpai}}(q,t)$:
  - $\text{logit}(P_{\text{tenpai}}(q,t))$
  - The number of players declaring riich.
- $p_{\text{lose}}(q,t)$:
  - $\text{logit}(P_{\text{Lose}}(q,t))$
  - The number of actions in $A_{\text{Take\&Discard}}$ i has chosen since $A_{\text{HandDistribution}}$.

Here, p^j_{tenpai} is an inferred probability that player j is tenpai at u_0 (see Sec. VI-C for more detail).

TABLE I
Probabilities in Eq. (18).

Symbol	Description
$p_{\text{win}}(q,t,u_0)$	Probability that the hand ends with player i's win scenario under the condition that the information set is u_0 of player i, i's hand is q, and i discards t tiles from u_0.
$p_{\text{washout}}(u_0)$	Probability that the hand ends with washout scenario under the condition that the information set is u_0 of i and i does not win the hand.
$p_{\text{tenpai}}(q,t,u_0)$	Probability that the hand ends with tenpai washout under the condition that the information set is u_0 of i, the hand ends with washout, i has hand q, and i discards t tiles from u_0.
$p_{\text{lose}}(q,t,u_0)$	Probability that the hand ends up with i's lose scenario under the condition that the information set is u_0 of i, i's hand is q, i discards t tiles from u_0, i does not win the hand, and the hand does not end with washout.

We calculate $U(q,t,z)$ in Eq. (17) as

$$\begin{aligned} U(q,t,\text{win}) &= \frac{V_{\text{win}}(q,t)-(1-P_{\text{win}}(q,t))U_{\text{NotWin}}}{P_{\text{win}}(q,t)} \\ U(q,t,\text{lose}) &= U_{\text{LoseAverage}}(q,t) \\ U(q,t,\text{other}) &= U_{\text{OtherWin}}. \end{aligned} \tag{19}$$

We calculate $U(q,t,\text{tenpai})$ and $U(q,t,\text{noten})$ on the basis of mahjong rules and tenpai probability p^j_{tenpai} of the other players $j \neq i$.

C. Parameters and Probabilities Used in MDPs

This section briefly describes methods for determining parameters and probabilities used in the MDPs. These methods include other heuristics that are not explained in this section. Readers interested in a more detailed description should consult our domestic conference paper [2], technical report [25], and program codes hosted in GitHub[4].

We first explain the collection of game records we used to obtain empirical probability distribution of gameplays generated by expert players. Tenhou[5] is a famous Japanese online mahjong website. There, players are ranked by the dan system, where players with better game results get higher dan. Games played on Tenho fall into four main groups. Only players holding 7-dan or higher (about 1% of active players) can participate in the highest-level group of highest level. Game records of this highest group are available online and used for adjusting parameters such as weights used by logistic regression.

Let the decision maker of these MDPs be player i, and i's current information set of mahjong be u_0 as before. The first parameter to be described is T. Let T_{max} be the maximum number of i's future actions in A_{Discards} until the current hand ends assuming that no player will choose actions in $A_{\text{Take\&Discard}}$. We set T to T_{max} for $\mathcal{M}$. For the other MDPs, we set T to $\lceil T_{\text{max}}\sigma_{\text{ratio}}\rceil$. We determine ratio σ_{ratio} on the basis of logistic regression using the same features as those used for p_{washout} and label $T_{\text{measured}}/T_{\text{max}}$, where T_{measured} is the number of future actions in A_{Discards} until the current hand ends. The training data (a set of feature and label pairs) are sampled from the recorded information sets of players who did not win at the corresponding hands.

The next parameters to be described are those used to infer probabilities related to the lose scenario. Some probabilities, such as $P(\text{TakeWin from } i|u_0,a)$ in Eq. (5), can be determined by $P(j \text{ TakeWin from } i,\phi_{\text{hand}}|h,t,u_0)$, i.e., the joint probability that another player j chooses a_{TakeWin} when i discards a tile of kind h by a after t actions in A_{Discards} from u_0 and the hand ends immediately with game situation ϕ_{hand}. We abbreviate the probability as $P^j_{\text{Lose}}(\phi_{\text{hand}}|h,t,u_0)$. Because j's hand must be tenpai when j chooses a_{TakeWin}, the probability can be factorized as

$$\begin{aligned} & P^j_{\text{Lose}}(\phi_{\text{hand}}|h,t,u_0) \\ = \;& P(j \text{ TakeWin from } i,\phi_{\text{hand}}|h,t,u_0,j \text{ is tenpai}) \\ \times \;& P(j \text{ is tenpai}|h,t,u_0). \end{aligned} \tag{20}$$

We infer the conditional probability $P(j \text{ TakeWin from } i,\phi_{\text{hand}}|h,t,u_0,j \text{ is tenpai})$ in two different ways. When j has chosen no action in $A_{\text{Take\&Discard}}$ since $A_{\text{HandDistribution}}$, it is inferred in such a way as to further factorize the probability and draw histograms in terms of scores of TakeWin from game records. When j has chosen one or more actions in $A_{\text{Take\&Discard}}$, it is inferred in such a way as to enumerate all possible tenpai hands for j. When a player has chosen actions in $A_{\text{Take\&Discard}}$ twice, the number of possible tenpai hands is in the of 100 thousands, and enumerating all of them does not significantly affect the total calculation time. When the number of such actions that j has chosen is one, it is not realistic to enumerate all tenpai hands. However, it is possible to enumerate the remaining seven tiles by ignoring one mentsu.

We also infer the conditional probability $P(j \text{ is tenpai}|h,t,u_0)$ by assuming it satisfies $P(j \text{ is tenpai}|h,t,u_0) = P(j \text{ is tenpai}|t,u_0)$ and using logistic regression similar to that in a previous study [22], but the major difference is that the regression model is fitted for each number of j's past actions in $A_{\text{Take\&Discard}}$ and for each number of j's past actions in A_{Discards} from $A_{\text{HandDistribution}}$ to u_0. We verified that this probability

[4]https://github.com/critter-mj/akochan
[5]http://tenhou.net/

rarely depends on h. We abbreviate the probability as $P^j_{\text{tenpai}}(t, u_0)$. This is also used for $p^j_{\text{tenpai}}(u_0)$ in Sec. VI-B, where t is set to that of s_{ini}.

The remaining probabilities are as follows. Probabilities that the chance players of MDPs choose an action $a \in A_{\text{DRAW}}$ or $A_{\text{OTHERDISCARD}}$ according to static probability, which are proportional to the number of i's invisible tiles of the kind h. That is,

$$P_{\text{D}}(h|q, u_0) = P_{\text{T}}(h|q, u_0) = \frac{m(h, q, u_0)}{\sum_h m(h, q, u_0)}, \tag{21}$$

where $m(h, q, u_0)$ is the number obtained by assuming that i is in a virtual situation where he/she has exchanged the minimum number of tiles unilaterally and continuously to realize hand q from u_0. The probability that the chance player chooses a_{LOSE} is inferred by assuming it satisfies

$$P(a_{\text{LOSE}}|q, h, t, u_0) = \sum_{j, \phi_{\text{hand}}} P^j_{\text{Lose}}(\phi_{\text{hand}}|h, t, u_0). \tag{22}$$

This is also used for $P_{\text{Lose}}(h|u_0)$ of $\mathcal{M}_{\text{Fold}}$, where t is set to that of s_{ini}. The probability that the chance player chooses a_{OTHERWIN}, i.e., $P(a_{\text{OTHERWIN}}|q, h, t, u_0)$, is inferred by assuming it satisfies $P(a_{\text{OTHERWIN}}|q, h, t, u_0) = P(a_{\text{OTHERWIN}}|t, u_0)$ and using logistic regression, where the number of other players declaring riich and estimated tenpai probabilities of the other players at u_0 are used as features. Recorded information sets of players who did not win or lose at the corresponding hands are used, and the probability that the other players declare win is learned for each t. We verified that this probability rarely depends on h.

We adjust parameters to represent payoff functions as follows. Let us assume that $\phi_{\text{DrawWin}}(q, h, u_0)$ is shared information ϕ_{hand} that represents an immediate end of the current hand by i's action a_{DRAWWIN} at in a virtual situation. The virtual situation is i's information set u_0 but hand q is substituted for i's hand, and h is a kind of the tile obtained from the drawing pile by a_{DRAWWIN}. Then $\phi_{\text{DrawWin}}(q, h, u_0)$ is determined in accordance with the rule sets and provides the definition of payoff function $U_{\text{DrawWin}}(q, h)$ as

$$U_{\text{DrawWin}}(q, h, u_0) = U^i_{\text{hand}}(\phi_{\text{DrawWin}}(q, h, u_0)) \tag{23}$$

using Eq. (2). In the same way, we assume that $\phi_{\text{TakeWin}}(q, h, u_0, j)$ is ϕ_{hand} that represents an immediate end by i's action a_{TAKEWIN} from player $j \neq i$ at the virtual situation, where h is a kind of the tile presented by a chance player as if it were j's discarding action $a \in A_{\text{DISCARDS}}$. Then $\phi_{\text{TakeWin}}(q, h, u_0)$ is determined in accordance with the rule sets and provides the definition of payoff function $U_{\text{TakeWin}}(q, h, u_0)$ as

$$U_{\text{TakeWin}}(q, h, u_0) = \frac{1}{3} \sum_j U^i_{\text{hand}}(\phi_{\text{TakeWin}}(q, h, u_0, j)). \tag{24}$$

We define payoff function $U_{\text{Lose}}(h, t, u_0)$ by omitting obvious u_0 dependency as

$$U_{\text{Lose}}(h, t) = \frac{\sum_{j, \phi_{\text{hand}}} P^j_{\text{Lose}}(\phi_{\text{hand}}|h, t) U^i_{\text{hand}}(\phi_{\text{hand}})}{\sum_{j, \phi_{\text{hand}}} P^j_{\text{Lose}}(\phi_{\text{hand}}|h, t)}. \tag{25}$$

Let us assume that $\phi_{\text{washout}}(u_0, b)$ be ϕ_{hand} that represents a future washout of u_0 and determined by whether each player's future hand is tenpai. Here, $b = (b_1, \ldots, b_4)$ is a tuple of four Booleans (b_j is true only when player j is tenpai). Then q is i's future hand from u_0, $B(q)$ is a set of all possible b (note that b_i always indicates whether q is tenpai), and payoff function $U_{\text{Washout}}(q, u_0)$ is defined as

$$\begin{aligned} U_{\text{Washout}}(q, u_0) &= \sum_{b \in B(q)} \Big[U^i_{\text{hand}}(\phi_{\text{washout}}(u_0, b)) \\ &\qquad \prod_{j \neq i} p^j(u_0, b_j) \Big] \\ p^j(u_0, \text{true}) &= p^j_{\text{tenpai}}(T, u_0) \\ p^j(u_0, \text{false}) &= 1 - p^j(u_0, \text{true}). \end{aligned} \tag{26}$$

Payoff function $U_{\text{Fold}}(q, t)$ is defined using Eq. (13), as $U_{\text{Fold}}(q, t) = E_{\text{Fold}}$. To calculate E_{Fold}, the optimal policy of $\mathcal{M}_{\text{fold}}$ is evaluated using $s_{\text{ini}} = (q, t = 0)$ and $T_{\text{fold}} = T - t$. Payoff function $U_{\text{OtherWin}}(u_0)$ is defined by omitting obvious u_0 dependency as

$$U_{\text{OtherWin}} = \frac{\sum_{j,k,h,\phi_{\text{hand}}} P^{jk}_{\text{Win}}(\phi_{\text{hand}}|h) U^i_{\text{hand}}(\phi_{\text{hand}})}{\sum_{j,k,h,\phi_{\text{hand}}} P^{jk}_{\text{Win}}(\phi_{\text{hand}}|h)}, \tag{27}$$

where both indexes j and k run for all players who are not i,

$$\begin{aligned} & P^{jk}_{\text{Win}}(\phi_{\text{hand}}|h, u_0) \\ = \; & P(j \text{ TAKEWIN from } k, \phi_{\text{hand}} \\ & \quad |k \text{ DISCARDS } h \text{ after } u_0, j \text{ is tenpai}) \\ \times \; & P(j \text{ is tenpai}|k \text{ DISCARDS } h \text{ after } u_0). \end{aligned} \tag{28}$$

for $j \neq k$, and

$$\begin{aligned} & P^{jj}_{\text{Win}}(\phi_{\text{hand}}|h, u_0) \\ = \; & P(j \text{ DRAWWIN}, \phi_{\text{hand}}|j \text{ DRAW } h \text{ after } u_0, j \text{ is tenpai}) \\ \times \; & P(j \text{ is tenpai}|j \text{ DRAW } h \text{ after } u_0). \end{aligned} \tag{29}$$

These probabilities are inferred as in Eq. (20).

D. Outline of Search Algorithm of MDPs

Our search algorithm to compute the expected final rank of a player at an information set has computational complexity proportional to the number of states of $\mathcal{M}$. Even ignoring actions in $a_{\text{TAKE\&DISCARD}}$, there are about 10^{11} patterns of a player's hand, and it is not realistic to search all states related to each hand. It is therefore desirable to reduce a sufficient number of states and actions of $\mathcal{M}$ so that the search algorithm ends with a realistic computational resource and the error of expected final rank does not increase.

For such reductions, we focus on states and actions related only to hands that can realize tenpai with a relatively small number of tile exchanges. We construct a set of such hands in the following four steps: (1) consider a graph in which a vertex represents a hand, an edge represents a tile exchange, and it takes into account all possible hands and tile exchanges, (2) enumerate paths with length n or less connecting the current hand q_0 and tenpai hand, (3) construct the set of hands $Q_{\text{S0}}(n)$

by enumerating vertices along all the paths including two terminals (i.e., q_0 and a tenpai hand), and (4) construct the set of hands $Q_{\mathrm{S}}(n,m)$ consisting of all q satisfying the condition that q' in $Q_{\mathrm{S0}}(n)$ exists such that m or fewer mentsus of q are revealed by taking from q.

Two integers n and m are parameters that control the size of the search space. Parameter n must be greater than or equal to the shanten-number of q_0 because the space must have some tenpai hands. As n and m are larger, the final rank prediction is expected to be more accurate. In our experiment, we adjusted these parameters according to the shanten-number of q_0 so that the AI player can make each decision in a few seconds with light-weight desktop computers. In this way, the size of $Q_{\mathrm{S}}(n,m)$ is controlled to be about 50,000. The search algorithm ignores any action that results in a hand not belonging to $Q_{\mathrm{S}}(n,m)$. Our search algorithm is based on retrograde analysis [26] with which the state values are determined from states with larger t.

E. Dealing with Popular Rules

In this section, we describe how our AI player deals with popular rules. A dora is a tile that increases the points of a hand if it is in the winning hand. A dora tile is selected using a dora indicator tile, which is chosen by the chance player with $A_{\textsc{HandDistribution}}$. This choice is shared by all players. The payoff of win or lose is determined in accordance with dora tiles.

Riich declaration is an action that can be chosen by a player who formed a tenpai hand without choosing any action in $A_{\textsc{Take\&Discard}}$ since $A_{\textsc{HandDistribution}}$. The player who declared riich is unable to change hands but is able to earn more points when he/she wins. We deal with riich declarations by adding hands after the declaration in $Q_{\mathrm{S}}(n,m)$ and modifying the payoff of win $U_{\mathrm{DrawWin}}(q,h)$ if q is the hand after the declaration. In addition, the folding tendency, i.e., other players tend to fold their hands when one declares riich, is reflected by modifying the values of $P(\overline{a_{\textsc{OtherWin}}}|q,t)$ and $P_{\mathrm{T}}(h|q,t)$ according to q.

VII. Experiment

This section presents the results of gameplays vis-à-vis existing AI players. We constructed the AI player with the proposed method as follows. When the shanten-number of the hand is zero or one, we use Eq. (16) to evaluate the values of legal actions. When the shanten-number of the hand is two or three, we use Eq. (17) to evaluate these values. In both cases, the player is greedy, i.e., the action with the highest value is selected. We tentatively set $\alpha = 0.1$ in Eq. (11). When the shanten-number of the hand is greater than three, we adopt a simple rule-based strategy. The rules used in this strategy basically determine whether to decrease the shanten number to win or fold the current hand. To decrease the shanten-number, the rules state to choose one of the isolated tiles to discard. To fold the hand, the rules state to choose a tile on the basis of value estimation using $\mathcal{M}_{\mathrm{fold}}$. The three AI players are one Bakuuchi and two copies of manue. The version of Bakuuchi is the one that achieved its highest dan and ratings (R2206) in Tenhou[6] and is stronger than that published in a previous paper [22]. Table II lists the results from 3557 gameplays of mahjong with the tonpu rule[7]. Because manue is clearly weak, we focused on to the difference between two ranks of our AI player and Bakuuchi for each gameplay and observed that the mean and deviation of the difference were 0.0574 and 1.822, respectively. Given the sample size was 3.56×10^3 and sample standard deviation was 1.822, the standard error of the sample mean 5.74×10^{-2} was 3.05×10^{-2}. Therefore, it is derived that the sample mean was positive with the one-tailed significance level of 0.03. This indicates that the performance of the AI player constructed with the proposed method reached the world's highest level.

TABLE II
Experimental results of 3557 gameplays of mahjong with tonpu rule. 1st to 4th columns show empirical probability obtained from results corresponding to each final ranking.

	1st	2nd	3rd	4th	Average Ranking
Our AI player	0.33	0.28	0.21	0.17	2.23 ± 0.04
Bakuuchi	0.32	0.27	0.21	0.20	2.29 ± 0.04
manue	0.17	0.22	0.29	0.31	2.74 ± 0.02

VIII. Conclusion and Future Work

We proposed a method of constructing a state-of-the-art AI mahjong player. With this method, multiple MDPs are introduced related to scenarios of a hand. When the shanten-number of the hand is less than two, MDP $\mathcal{M}$ plays an essential role for estimating actions values in mahjong situation. It takes into account as many scenarios as possible, and the analysis results are directly used for evaluating actions. When the shanten-number of the hand is two or more, we use the results of $\mathcal{M}_{\mathrm{win}}$, $\mathcal{M}_{\mathrm{tenpai}}$, and $\mathcal{M}_{\mathrm{fold}}$. These MDPs are focused on a few specific scenarios, and the analysis results are used as features for inferring state values. We reduced the number of MDP states to the extent that the expected final-rank error does not increase so that the calculation ends in a few seconds.

We presented the results of 3557 gameplays of mahjong with the AI player constructed with the proposed method and two current AI players, i.e., one version of Bakuuchi, the strongest player, and two versions of manue whose source code is published. The results indicate the effectiveness of the proposed method.

We believe that state-of-the-art AI players (e.g. Bakuuchi and the AI player constructed with the proposed method) are reaching the level of the top 200 human Tenhou players. It is interesting to measure the performance of these AI players vis-à-vis top human players. Unfortunately, we cannot do this because three top players have to play with the AI player for more than one month in a row. To recognize the strength difference between two top human players, they have to play a 20-minute game several thousand times.

The models constructed with the proposed method include an action that does not exist in mahjong. That is, an action sequence in gameplay to avoid lose and abandon win (fold)

[6]http://tenhou.net/

[7]This took several months using an ordinary desktop PC.

is represented using a virtual action a_{FOLD}. It is reasonable because an expert player rarely changes her/his strategies in attempting a win scenario from avoiding lose scenario in the middle of the action sequence. However, if possible, it is desirable to handle them comprehensively. Also, because it may be over simplification to deal with the three opponent players as a single chance player, more sophisticated methods may improve the performance of AI players.

Larger but simpler models, such as deep neural networks, would be better to build stronger AI players. However, we have not yet constructed such AI players in mahjong. Due to the large size of the game tree, a larger model will be required compared, for example, to poker models.

ACKNOWLEDGMENTS

The authors would like to thank Naoki Mizukami for his helpful comments and support during the for experiments. This research partly used computational resources provided by HEROZ. This work was supported by JSPS KAKENHI Grant Numbers JP16K00503 and JP18H03347.

REFERENCES

[1] D. Fudenberg and J. Tirole, *Game Theory*. The MIT Press, 1991.
[2] M. Kurita and K. Hoki, "Development of mahjong player on the basis of kyoku abstraction for multiple goals and value inference (in japanese)," in *The 22nd Game Programming Workshop*, vol. 2017, nov 2017, pp. 72–79.
[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.
[4] S. D. Miller, *Riichi Mahjong The Ultimate Guide to the Japanese Game Taking the World by Storm*. Psionic Press, 2015.
[5] K. Woolsey, *How to Play Tournament Backgammon*. The Gammon Press, 1993.
[6] L. V. Allis, *Searching for Solutions in Games and Artificial Intelligence*. Ph.D. thesis, University of Limburg, 1994.
[7] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995.
[8] J. Schaeffer, *One Jump Ahead: Computer Perfection at Checkers (2nd Edition)*. Springer, 2008.
[9] M. Campbell, A. Hoane, and F. hsiung Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57 – 83, 2002.
[10] K. Hoki, D. Yokoyama, T. Obata, H. Yamashita, T. Kaneko, Y. Tsuruoka, and T. Ito, "Distributed-shogi-system akara 2010 and its demonstration," *International Journal of Computer and Information Science*, vol. 14, no. 2, pp. 55 – 63, 2013.
[11] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.
[13] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.
[14] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 1729–1736.
[15] M. Bowling, N. Burch, M. Johanson, and O. Tammelin, "Heads-up limit hold'em poker is solved," *Commun. ACM*, vol. 60, no. 11, pp. 81–88, Oct. 2017.
[16] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017.
[17] B. Sheppard, "World-championship-caliber scrabble," *Artificial Intelligence*, vol. 134, no. 1, pp. 241 – 275, 2002.
[18] N. A. Risk and D. Szafron, "Using counterfactual regret minimization to create competitive multiplayer poker agents," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, ser. AAMAS '10. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 159–166.
[19] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron, "The challenge of poker," *Artificial Intelligence*, vol. 134, no. 1, pp. 201 – 240, 2002.
[20] M. Buro, J. R. Long, T. Furtak, and N. Sturtevant, "Improving state evaluation, inference, and search in trick-based card games," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1407–1413.
[21] I. Szita, G. Chaslot, and P. Spronck, "Monte-carlo tree search in settlers of catan," in *Advances in Computer Games*, H. J. van den Herik and P. Spronck, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 21–32.
[22] N. Mizukami and Y. Tsuruoka, "Building a computer mahjong player based on monte carlo simulation and opponent models," *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 275–283, 2015.
[23] T. Sandholm, "Abstraction for solving large incomplete-information games," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, pp. 4127–4131.
[24] N. Basilico and N. Gatti, "Automated abstractions for patrolling security games," 2011.
[25] M. Kurita and K. Hoki, "Ron-probability inference on the basis of prediction of other players' hands and waitings in mahjong (in japanese)," in *Special Interest Group Technical Reports of IPSJ: Game Informatics (GI)*, no. 5, jul 2017, pp. 1–8.
[26] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen, "Checkers is solved," *Science*, vol. 317, no. 5844, pp. 1518–1522, 2007.

APPENDIX

This appendix provides supplementary tables that summarize the action sets (Table A), inferred probability (Table IV), and inferred expected payoffs (Table V).

This paper focuses on four different domains, i.e., one four-player extensive-form game (mahjong) and four MDPs ($\mathcal{M}, \mathcal{M}_{\text{Fold}}, \mathcal{M}_{\text{Win}}, \mathcal{M}_{\text{Tenpai}}$). The players (including chance players) in these domains deal with a variety of actions, which are categorized into multiple types, as shown in Table II. Each action set contains all actions of the same category. In addition to the action sets in the table, we also use $A_{\text{DISCARDS}} = A_{\text{DISCARD}} \cup A_{\text{TAKE\&DISCARD}}$ and $A_{\text{WIN}} = A_{\text{DRAWWIN}} \cup A_{\text{TAKEWIN}}$. The checkmarks indicate the action types treated in these domains. This table also indicates whether a type of action of the type is chosen by a chance player. The table also shows the number of each type of action by assuming one of the most basic mahjong rules commonly used in Japan.

These MDPs are defined using several probabilities listed in Table IV and payoff functions (i.e., terminal rewards) listed in Table V. Let us consider the most basic MDP (illustrated in Fig. 3) to evaluate player i's actions at information set u_0. Probability $P(a_{\text{LOSE}}|q,h,t,u_0)$ forms the transition probability $P(s'|s,a_{\text{DISCARD}})$ when $s = (q,h,\texttt{Discard},t,u_0)$ holds. This probability can be positive when s' is of type `Discard` or is two terminal states associated with each of the two payoff functions $U_{\text{Lose}}(q,t,u_0)$ and $U_{\text{Washout}}(q,u_0)$. Probabilities $P(a_{\text{OTHERWIN}}|q,t,u_0)$ and $P_{\text{T}}(\cdot|q,t)$ form the transition probability $P(s'|s,a_{\text{NOTFOLD}})$ when $s = (q,h,\texttt{Fold},t,u_0)$ holds. This probability can be positive when s' is of type `Take` or is a terminal state associate with payoff function $U_{\text{Fold}}(q,t,u_0)$. Probability $P_{\text{D}}(\cdot|q,t)$ forms the transition probability $P(s'|s,a_{\text{PASS}})$ when $s = (q,h,\texttt{Take},t,u_0)$ holds. This probability can be positive when s' is of type `Discard`.

TABLE III
Table of action types. 'Org' means original game of mahjong, and 'Chance' means chance player.

Action set	Org	$\mathcal{M}$	$\mathcal{M}_{\text{Fold}}$	$\mathcal{M}_{\text{Win}}$	$\mathcal{M}_{\text{Tenpai}}$	Chance	Size
$A_{\text{HANDDISTRIBUTION}}$	✓					Yes	10^{44}
A_{DRAW}	✓	✓		✓	✓	Yes	34
A_{DISCARD}	✓	✓	✓	✓	✓		34†
$A_{\text{TAKE\&DISCARD}}$	✓	✓		✓	✓		3400
A_{DRAWWIN}	✓	✓		✓			1
A_{TAKEWIN}	✓	✓		✓			1
A_{PASS}	✓	✓		✓	✓		1
A_{FOLD}		✓					1
A_{NOTFOLD}		✓					1
$A_{\text{OTHERDISCARD}}$		✓		✓	✓	Yes	34
A_{LOSE}		✓	✓			Yes	1
A_{OTHERWIN}		✓	✓			Yes	1

† Size is twice as large as notation when rule set of riich is applied.

TABLE IV
Probabilities in MDPs. Checkmarks indicate probabilities used in each domain. Column labeled 'Eq.' shows references of equations that explain corresponding probabilities.

Symbol	Description	$\mathcal{M}$	$\mathcal{M}_{\text{Fold}}$	$\mathcal{M}_{\text{Win}}$	$\mathcal{M}_{\text{Tenpai}}$	Eq.
$P(a_{\text{LOSE}}\|q,h,t,u_0)$	Probability that chance player chooses a_{LOSE} as if hand ended up with lose scenario immediately after i, who discarded t tiles from his/her information set u_0, discards a tile of kind h.	✓	✓			(22)
$P(a_{\text{OTHERWIN}}\|q,t,u_0)$	Probability that chance player chooses a_{OTHERWIN} as if hand ended with other win scenario after player i has discarded t tiles from his/her information set u_0.	✓	✓			
$P_{\text{D}}(h\|q,t,u_0)$	Probability that chance player serves a tile of kind h as if it were brought to player i from the drawing pile after i has exchanged minimum number of tiles to realize hand q from i's information set u_0.	✓		✓	✓	(21)
$P_{\text{T}}(h\|q,t,u_0)$	Probability that chance player presents a tile of kind h to i as if it were discarded by one of other players after i has exchanged minimum number of tiles to realize hand q from i's information set u_0.	✓		✓	✓	(21)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TG.2020.3036471, IEEE Transactions on Games

TABLE V
Payoff functions in MDPs. Checkmarks indicate functions used in each domain. Column labeled 'Eq.' shows references of equations that explain corresponding payoff functions.

Symbol	Description	$\mathcal{M}$	$\mathcal{M}_{\text{Fold}}$	$\mathcal{M}_{\text{Win}}$	$\mathcal{M}_{\text{Tenpai}}$	Eq.
$U_{\text{Fold}}(q, t, u_0)$	Terminal reward when i of u_0 chooses a_{FOLD} in $s = (q, \text{null}, \texttt{Fold}, t, u_0)$.	✓				
$U_{\text{OtherWin}}(u_0)$	Terminal reward when chance player chooses a_{OTHERWIN} after i of u_0 chooses a_{NOTFOLD}.	✓	✓			(27)
$U_{\text{TakeWin}}(q, h, u_0)$	Terminal reward when i of u_0 chooses a_{TAKEWIN} in $s = (q, h, \texttt{Take}, t, u_0)$.	✓		✓		(24)
$U_{\text{DrawWin}}(q, h, u_0)$	Terminal reward when i of u_0 chooses a_{TAKEWIN} in $s = (q, h, \texttt{Discard}, t, u_0)$.	✓		✓		(23)
$U_{\text{Lose}}(h, u_0)$	Terminal reward when chance player chooses a_{LOSE} after i of u_0 discards a tile of kind h.	✓	✓			(25)
$U_{\text{Washout}}(q, u_0)$	Terminal reward when number of tiles that i discards from his/her u_0 becomes T.	✓	✓	✓	✓	(26)