Evaluating Gin Rummy Hands Using Opponent Modeling and Myopic Meld Distance

Phoebe Goldman, *1 **Corey R. Knutson,** *2 **Ryan Mahtab,** *3
Jack Maloney, *4 **Joseph B. Mueller,** 5,6 **Richard G. Freedman** 6
1New York University College of Arts and Science
2University of Minnesota Duluth
3Carnegie Mellon University
4University of Minnesota Twin Cities
5University of Minnesota
6SIFT, LLC

phoebe.goldman@nyu.edu, knuts983@d.umn.edu, rmahtab@andrew.cmu.edu, malon559@umn.edu, muel0053@umn.edu, {rfreedman,jmueller}@sift.net

Abstract

Gin Rummy is a popular two-player card game involving choices to draw and discard cards to form sets of matching cards. Unlike other popular games such as Chess, Poker, and Go, there is little formal artificial intelligence research about how to make good decisions when playing Gin Rummy. In this paper, we develop an agent that plays Gin Rummy through a combination of known and expected card values, modeling the opponent to predict their cards of interest, and a conservative approach to assessing when to end the hand. In addition to discussing our observations about Gin Rummy that inspired our agent's design and how the agent works, we evaluate the relative importance of various features employed by our agent by competing agents which implement various subsets of those features.

1 Introduction

Gin Rummy is a two-player card game that, despite having simple rules, introduces the opportunity for complex strategic decision-making based on both hidden information and randomness. Unlike other incomplete information games such as Poker, there has been little scholarly study about playing Gin Rummy. Although Poker has a betting element that is absent in Gin Rummy, it contains other interesting challenges not found in Poker such as a *sequence of actions per turn* and *gambling whether to terminate the round* in confidence of having a better score than the opponent. The absence of existing strategies in the literature, the simplicity of the game rules, and the potential to implement complex reasoning makes Gin Rummy an intriguing problem space for artificial intelligence research.

Using the limited information available to a player, our approach uses approximate statistical analysis of potential hand outcomes and analysis of the opponent's strategy given observations of their actions each turn. In addition, we applied domain knowledge from our experience playing Gin Rummy to introduce a conservative, but surprisingly simple and effective, strategy that decides to end the round only when the player has the best possible score (because the objective is to minimize a non-negative score, this is 0). These elements synergize to prepare for the case that the opponent ends the game first because there is a penalty for ending the game while having a greater score than the other player.

In this paper, we present our Gin Rummy-playing agent and its components in further detail. After we present how to play Gin Rummy with our insights in Section 2, Section 3 reviews related research about gameplaying agents. Following this background, Section 4 introduces each component of our agent in more depth. To study how our agent performs, we perform experiments involving play against variations of itself in Section 5. Based on the results, we provide discussions and propose future work in Section 6.

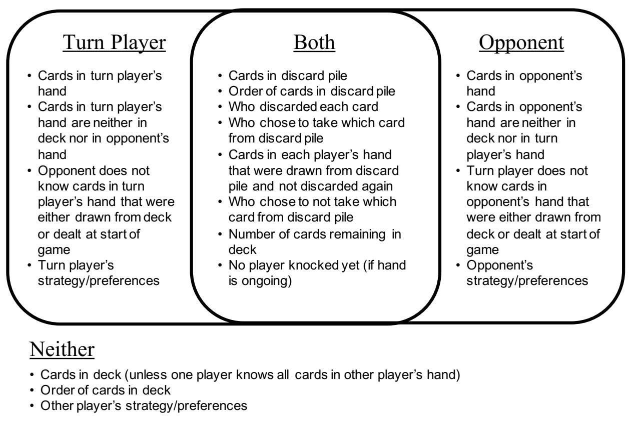
*These authors contributed equally to this work.

2 Game Rules

Gin Rummy is a two-player card game played in a series of rounds. At the beginning of each round, each player is dealt a ten-card hand from a standard 52-card deck. One additional card is placed face up as the initial discard pile. The players alternate turns making two or three decisions in sequence:

1. whether to draw the top card from the discard pile (face-up) or from the deck (face-down),
2. which card to discard from their hand, and
3. whether to *knock* and end the round (if applicable).

Each card has a point value, with aces worth one point, pip cards worth their number, and face cards worth ten points. Cards are worth no points if they are grouped into *melds* of at least three cards, either all matching rank (*sets*) or forming a straight within the same suit (*runs*). The sum of the cards' point values in a player's hand is called their *deadwood*. If a player has at most ten deadwood points at the end of a


Turn Player

- Cards in turn player's hand
- Cards in turn player's hand are neither in deck nor in opponent's hand
- Opponent does not know cards in turn player's hand that were either drawn from deck or dealt at start of game
- Turn player's strategy/preferences

Both

- Cards in discard pile
- Order of cards in discard pile
- Who discarded each card
- Who chose to take which card from discard pile
- Cards in each player's hand that were drawn from discard pile and not discarded again
- Who chose to not take which card from discard pile
- Number of cards remaining in deck
- No player knocked yet (if hand is ongoing)

Opponent

- Cards in opponent's hand
- Cards in opponent's hand are neither in deck nor in turn player's hand
- Turn player does not know cards in opponent's hand that were either drawn from deck or dealt at start of game
- Opponent's strategy/preferences

Neither

- Cards in deck (unless one player knows all cards in other player's hand)
- Order of cards in deck
- Other player's strategy/preferences

Figure 1: Observable and known information to each player during a game of Gin Rummy. "Turn player" and "Opponent" are relative to whoever is acting on the current turn.

their turn, they may knock to reveal their hand and declare their melds. Their opponent takes one final turn, reveals their hand and declares their melds; the opponent is then given the opportunity to *lay off* unmelded cards into the knocking player's melds. After laying off, the player with fewer deadwood points wins the round and scores points equal to the difference between the two players' deadwood points. The knocking player loses ties. Under the American rules, a player who knocks with zero deadwood points goes *gin* and scores an additional 25 points; if the winner was not the player who knocked, then they *undercut* their opponent and score an additional 25 points. At the end of the round, a player wins if they have accrued at least 100 points throughout the game—otherwise, they play another round.

2.1 Observable Information

The information available to each player is *asymmetric* because the hand of cards reveals their contents exclusively to the hand's owner. Figure 1 displays what information is available to each player during a game of Gin Rummy. In general, a player can observe the ten cards in their personal hand as well as all the cards (and their order) in the discard pile. Neither player knows the order of the cards that remain in the deck, and the degree to which a player knows which cards are in the deck depends on how much they know (and remember) about the cards in their opponent's hand.

On a player's turn, the new information that they observe is limited. Drawing the top card of the discard pile does not reveal any new information to that turn's player, but it is their final opportunity to add the card to their hand. On the other hand, drawing the top card from deck reveals another card that was not yet seen; this tells the player that it was not a card that is hidden in the opponent's hand. The player then knows that such a card is no longer in the deck as well.

During the opponent's turn, a player observes many new pieces of information: whether the opponent chose to draw or ignore the top face-up card in the discard pile, which card the opponent chose to discard, and whether the opponent chose to knock. In this case, the opponent's choice to do something is as informative as the opponent's choice to *not* do something because either action implies information about their unique knowledge/observations.

3 Related Works

Gin Rummy is far from the first game to be a target of study within artificial intelligence (AI). At a panel about the history of AI playing games, celebrating the fiftieth anniversary of the first computer chess tournament, many speakers noted that games have been a popular domain since the earliest days of the field (Greenwald et al. 2020). They cited various scientists' reasons for this interest, ranging from focus on the core challenges as a gateway to real-world problems, to better understanding how people think when solving challenges. We focus on several games and their related AI techniques that share commonalities with our approach.

Algorithms inspired by game theory defined early approaches to *adversarial search* that are still used today. These techniques propagate each player's actions across board states to create trees of possible game histories—depth-first search computes the path from the root that optimizes each turn player's final score (or maximizes the difference in scores when the player with the greatest score wins). Such techniques include Minimax (Russell and Norvig 2009) and Monte-Carlo Tree Search (Chaslot et al. 2006).

For games whose state spaces are too large for feasible evaluation from the leaves to the root, depth cut-offs limit the propagation to several turns' foresight rather than entire game histories. To account for myopic decision making where the final payoffs are not involved in the adversarial search computation, *evaluation functions* estimate the value of intermediate game states. These functions serve as a heuristic to guide the search when the exact payoffs are too far into the game's future to be known. Domain knowledge can engineer reasonable evaluation functions for well-studied games like Chess (Shannon 1950). Games with greater uncertainty due to randomness, such as Backgammon (Tesauro 1995), or enormous state spaces that cannot possibly be as well-studied, such as Go (Silver et al. 2016), employ machine learning to automate the development of evaluation functions because people have not yet formalized the best features to assess intermediate game states.

Unlike the above games where the intermediate game states are fully observable to all players (pieces on a board), Gin Rummy has intermediate states with much more hidden information between the cards in the deck and opponent's hand. These features are also prevalent in the card game Poker, which the AI community has studied formally for about two decades. Similar to the early Poker-playing agent Loki-2 (Billings et al. 1999), our agent relies on probabilistic approximations to address the high uncertainty when evaluating intermediate states. Loki-2's *Hand Evaluator* component considers how its hand's value could compare to possible combinations of cards in some opponent's hand.

The sets of possible cards available to the opponent are also updated via Loki-2's *Opponent Modeling* component that translates their revealed cards and actions into cards that influence such decisions. Other opponent modeling techniques that have been used in Poker include learning each opponent's parameters within a strategy space (Hoehn et al. 2005) and using methods like particle filters to approximate the opponent's state (Bard and Bowling 2007). Because Gin Rummy has a greater focus on matching cards than outranking an opponent (or bluffing) to win, we adjust our uses of opponent modeling and hand evaluation accordingly.

The latter approaches are similar to an application of probabilistic plan recognition to hypothesize opponent strategies in the videogame StarCraft (Kabanza et al. 2013). In general, plan, activity, and intent recognition research provide opportunities to model the other agents based on their actions. These models can identify the opponents' waypoints of interest in first-person shooter videogames (Tastan, Chang, and Sukthankar 2012) as well as infer sets of goals over which the observing agent may compute responsive plans (Freedman and Zilberstein 2017). Although we do not use recognition algorithms, our agent similarly uses the opponent model to affect its reasoning over actions to take.

4 Methods

Our Gin Rummy-playing agent consists of three main components, each used for at least one decision-making task: Myopic Meld Distance estimation to identify potentially valuable cards for the agent's hand, Opponent Modeling to predict potentially valuable cards for the opponent's hand, and a conservative knocking strategy. We describe how each component works below. For the majority of the in-game decisions, our agent uses these components to compute a hand evaluation function r_t based on the current game state. S.

Definition 1 *A game state S is a tuple*

$$(H_i, H_o, D, t, D_i, U)$$

where H_i is the set of cards in the agent's hand, H_o the set of known cards in the opponent's hand, D the set of cards in the discard pile, $t \in D$ the face-up card on top of the discard pile, $D_i \subset (D \cup H_o)$ the set of cards the agent has discarded, and U the set of unrevealed cards either remaining in the face-down pile or hidden in the opponent's hand.

Definition 2 *A* hand evaluation function *r_t maps (H, S) tuples to positive real numbers, where H is a hand to evaluate in game state S. $r_t(H_1, S) < r_t(H_2, S)$ denotes that H_1 is more apt to win than H_2 from game-state S.*

We note that, for any game state S, $H_i \cup H_o \cup D \cup U = \{$A♠, 2♠, . . . , K♦$\}$ is the set of all 52 cards in the game.

4.1 Agent Assumptions

We assume that our opponent will never discard cards which they have chosen to draw face up; that is, we assume that all known cards in H_o are permanently inaccessible to our agent, and thus that we can never achieve any melds containing those cards. A survey of games by human and computer players could potentially validate or reject this assumption, were such data available, and future research might seek to demonstrate using tests whether the removal of this assumption decreases or improves an agent's win rate.

We also assume that all cards $c \in U$ are equally likely to be drawn; for random variable Γ as the top card of the deck,

$$P(\Gamma = c) = \frac{1}{|U|} \tag{1}$$

A more sophisticated agent might alter its expectation of drawing particular cards based on inferences about the contents of its opponent's hand based on their behavior; for example, noticing that an opponent has picked up both 2♦ and 2♣ from the face-up pile could reduce the projected probability of drawing either 2♥ or 2♠, as the opponent is likely to already have one or both of those cards.

When counting the deadwood in a hand, there might be multiple sets of melds that yield the same deadwood points for the hand. However, of all the meld sets that derive some hand's minimum deadwood points, we choose one arbitrarily for computation assuming no loss of generality. We discuss the implications of this assumption in Section 6.

Whenever an opponent discards a card, we assume that they have no intention of forming any melds which would be improved by that card. Likewise, if the opponent decides to draw the top card from the deck instead of t, we assume that the opponent has no intention of forming a meld which would be improved by t. These assumptions allow us to quickly prune the potential melds that our opponent could make, leading to more reasonable predictions when the assumption is accurate. We decided this improvement is worth the trade-off of inaccurate predictions when our opponent passes or discards cards that would fit in a meld they are pursuing—we believe such moments are rare. A more conservative agent would need only to prune those melds which are inaccessible as a result of excluding such cards, leaving as possibilities those which are improved by those cards but do not depend on them. Future research might seek to determine whether such an agent performs better or worse than one which makes this assumption.

4.2 Choosing What to Draw and Discard

Our agent considers whether to draw the face-up or face-down card at the same time that it chooses which card to discard using an evaluation function r_t, which computes a positive real number representing the expected deadwood value of a given ten-card hand in a particular game state. r_t includes the weighted sum of two functions r_m and r_o, which Sections 4.3 and 4.4 respectively describe in detail. $d_{min}(h)$ is the minimum amount of deadwood in hand h after removing its valid melds, and w_m and w_o are tunable weights.

$$r_t(h) = d_{min}(h) - w_m r_m(h) - w_o r_o(h) \tag{2}$$

We first explain how the agent determines the value of drawing the face-up card in the discard pile. Adding the face-up card to the agent's hand yields the set $H_i \cup \{t\}$ of eleven cards. Because the taken face-up card cannot be discarded on the same turn, the agent only considers the $|H_i| = 10$ choices for which card to discard—our agent ignores the case of drawing and discarding the deck's top face-down card during this stage of the computation. For each of the ten possible ten-card hands resulting from discarding a card, our agent runs the hand evaluation function in Equation 2 above to score them. This takes into account the deadwood of the hand, the potential for a hand to create new melds, and the potential for a hand to disrupt the opponent's play or be laid off onto opponents melds. The agent then takes the minimum of these scores and assigns it as the value for the face-up card t. This is because the agent will choose to discard the card that minimizes the hand evaluation function $r_t(H_i)$ after taking t. Mathematically, $R(c, H_i)$ is the value of H_i after drawing c:

$$R(c, H_i) = \min_{x \in H_i} \{r_t((H_i \cup \{c\}) \setminus x)\}. \tag{3}$$

The calculation of the value for the face-down card is similar, except the drawn card may be discarded:

$$R'(c, H_i) = \min_{x \in (H_i \cup \{c\})} \{r_t((H_i \cup \{c\}) \setminus x)\}. \tag{4}$$

We then take the expected value of this modified function $R'(c, H_i)$ over all the cards in U:

$$R_f = \sum_{c \in U} P(\Gamma = c) R'(c, H_i) = \frac{1}{|U|} \sum_{c \in U} R'(c, H_i). \tag{5}$$

The simplification is true because of our assumption in Equation 1 that $\mathrm{P}(\Gamma = c)$ is uniformly distributed.

The agent finally compares $R_t = R(t, H_i)$ and R_f, choosing the actions that lead to the lower score. If the face-up card is chosen, the agent draws it, then discards the precomputed card associated with the best hand containing the face-up card. If the face-down card is chosen, the agent decides which card to discard by evaluating the eleven possible result hands as in equation 4. As deciding whether to draw the face-up or face-down card has already computed the best discard associated with each possible draw in the face-down pile, our agent's performance might be improved by caching those best discards. Instead, we rerun that computation for the drawn card; as yet, such performance optimizations have not been a serious concern.

4.3 Myopic Meld Distance

Many hands have the same naïve deadwood value, but given a particular game state, some hands are worth more than others. Consider the following two hands.

$$\text{2♦ 6♦ 7♦ 9♦ J♠ J♥ K♥ 3♣ 8♣ 9♣} \tag{6}$$

$$\text{8♦ Q♦ 6♠ K♠ 2♥ 9♥ 7♥ 5♣ 7♣ J♣} \tag{7}$$

Both of these hands have 74 deadwood. However, hand (6) is better because it has many sets of cards that are *almost-melds* while hand (7) has almost no almost-melds. We propose a myopic algorithm to choose our draw and discard cards based on sets in our own hand that are at most one card away from becoming a meld.

Definition 3 *Let H_i be the set of cards that are in the agent's hand. Then an* almost-meld *is a set $\Lambda \cup \{a\}$, where $\Lambda \subset H_i$ and $a \in U$. We call Λ and a the* root *and* affix *of the almost-meld, respectively.*

With this definition, assuming $D = H_o = \emptyset$, we can see that hand (6) has the following ten almost-melds:

$$\begin{gathered}\{\text{6♦}, \text{7♦}, \text{8♦}\}, \{\text{6♦}, \text{7♦}, \text{8♦}, \text{9♦}\}, \{\text{5♦}, \text{6♦}, \text{7♦}\}, \\ \{\text{J♠}, \text{J♥}, \text{J♣}\}, \{\text{J♠}, \text{J♥}, \text{J♦}\}, \{\text{8♣}, \text{9♣}, \text{10♣}\}, \\ \{\text{7♣}, \text{8♣}, \text{9♣}\}, \{\text{9♣}, \text{9♦}, \text{9♠}\}, \{\text{9♣}, \text{9♦}, \text{9♥}\}, \\ \{\text{J♥}, \text{Q♥}, \text{K♥}\}\end{gathered} \tag{8}$$

and hand (7) has only two almost-melds:

$$\{\text{5♣}, \text{6♣}, \text{7♣}\}, \{\text{7♥}, \text{8♥}, \text{9♥}\}. \tag{9}$$

Let us inspect one of the almost-melds from hand (6): $\{\text{J♠}, \text{J♥}, \text{J♣}\}$. We can partition this almost-meld into the root $\Lambda = \{\text{J♠}, \text{J♥}\} \subset H_i$ and the affix $a = \text{J♣} \in U$.

Using this fact, we can now evaluate each hand to better represent its potential to improve. Our evaluation function begins with the deadwood of a hand, and then it reduces this value for hands with better potential. We construct a map where the keys are the cards in the agent's hand $c \in H_i$, and the values $V(c, H_i)$ are the set of possible affixes for the set of almost-melds in H_i that contain c.

Continuing our example, J♥ maps to value

$$V(\text{J♥}, \text{2♦ 6♦ 7♦ 9♦ J♠ J♥ K♥ 3♣ 8♣ 9♣}) = \{\text{J♣}, \text{J♦}, \text{Q♥}\}$$

because the three almost-melds in (8) that contain J♥ are $\{\text{J♠}, \text{J♥}, \text{J♣}\}$, $\{\text{J♠}, \text{J♥}, \text{J♦}\}$, and $\{\text{J♥}, \text{Q♥}, \text{K♥}\}$. Thus the affixes of these almost-melds are J♣, J♦, and Q♥, respectively.

Let W_{H_i} be the set of cards already in a complete optimal meld. We can compute the expected deadwood reduction using this map via the following formula:

$$r_m = \sum_{c \in (H_i \setminus W_{H_i})} \sum_{a \in V(c, H_i)} \mathrm{P}(\Gamma = a)\, d(c) \tag{10}$$

where $\mathrm{P}(\Gamma = a)$ is the probability of drawing a next turn and $d(c)$ is the deadwood value of c. Using our assumption that $P(a)$ is uniformly distributed, we can rewrite Equation 10 as

$$r_m = \sum_{c \in (H_i \setminus W_{H_i})} \frac{|V(c, H_i)|}{|U|} d(c). \tag{11}$$

r_m naïvely computes the reduction in deadwood that we would expect to have after drawing one card. Drawing any card in $V(c, H_i)$ forms a new meld in the player's hand, which would reduce $d(c)$ to 0 because cards in a meld are no longer scored as part of the hand's deadwood. This ignores the fact that certain almost-melds for a single hand may not be mutually exclusive, introducing double-counting during expected deadwood reduction.

4.4 Opponent Modeling

Knowing what cards are in the opponent's hand can be key to winning a game of Gin Rummy. If we know the melds in the opponent's hand, then we can decide which of our cards to discard that would benefit our opponent the least. The agent should only discard cards from their hand that will not go into the opponent's melds for two reasons:

1. they are very likely to pick up the card, and
2. if the agent is confident that the card will go into an opponent's meld, then that card is effectively worth zero deadwood if they knock because the agent can lay off the card.

While it is impossible to know all the cards an opponent has in their hand, our agent can make an informed decision based on what it knows in the current state of the game.

First, we want to find the set of all possible cards an opponent could have, including the cards in the agent's hand since they must discard one card at the end of their turn:

$$C_p = H_i \cup H_o \cup U. \tag{12}$$

The agent does not want to discard a card that would potentially meld with the opponent's hand. Next, we calculate the set of all possible meld sets M_p with respect to C_p.

To reduce the search space, we take the subset of melds in M_p that contain *at least one card from the agent's hand*:

$$M_1 = \left\{ M \,\middle|\, M \in M_p \wedge (M \cap H_i \neq \emptyset) \right\}. \tag{13}$$

We also reduce the search space to only include melds that contain *at least one card that we know the opponent has*:

$$M_2 = \left\{ M \,\middle|\, M \in M_1 \wedge (M \cap H_o \neq \emptyset) \right\}. \tag{14}$$

The reductions in Equations 13 and 14 allow us to filter out melds that the agent cannot affect as well as the melds that the agent has no evidence to support.

We reduce the search space even further by removing melds that we assume our opponent will not try to attain, due to our assumption that an opponent will not try to make any melds that contain a card they either discarded or chose not to take from the discard pile. In particular, this additional pruning applies to three-card sets because all cards have the same rank; thus the opponent would *never forfeit a card of a desired set's rank*. For example, if the opponent discards J♥, then the agent should assume that the opponent is not interested in meld {J♠, J♦, J♣} because J♥ could join any pair of cards to form yet another set-type meld.

$$M_3 = \Big\{ M \,|\, M \in M_2 \wedge \\ (isSet(M) \rightarrow (ranks(M) \cap ranks(D_o \cup P_o) = \emptyset)) \Big\} \tag{15}$$

In the above equation, $isSet(M)$ evaluates to true if and only if the set of cards M is a set-type of meld, and $ranks(M)$ abstracts M to the set of ranks for each card in M. Equation 15 does not prune any melds of the run type because the set of melds M_p already removed all such melds related to the above assumption. Discarding J♥ means J♥ $\notin C_p$, and run-type melds in M_p without J♥ are distinct from run-type melds with the card.

With our final set of melds M_3, we compute the deadwood reduction our opponent would gain for each possible card that we could discard from H_i.

$$r_o = \sum_{c \in H_i} \sum_{M \in \{\mu | \mu \in M_3 \wedge c \in \mu\}} P(M \in M_o)\, d(c) \tag{16}$$

where M_o is a random variable for the set of melds in the opponent's hand and $d(c)$ calculates the deadwood value of c. The probability that a meld is in the opponent's hand is

$$P(M \in M_o) = \prod_{c \in M} \begin{cases} 1 & c \in H_o \\ \frac{1}{|U|} & c \notin H_o \end{cases}. \tag{17}$$

By design, r_o represents the expected score by which the opponent's hand would decrease with respect to the ten cards in the agent's hand H_i.

4.5 Knocking Strategy

Under the American rules, the gin and undercut bonuses are both worth 25 points. To us, the penalty for knocking too early is severe and the reward is small; correctly predicting that it is safe to knock without gin will usually reward a handful of points, but predicting wrong and being undercut gives the opponent one quarter of the threshold to win. Similarly, the reward for going gin is large enough that, even when a player is confident that they would win the hand, it is often worth the risk to play additional turns for the chance to form melds with last few cards and achieve gin. Specifically, the risk is the possibility that their opponent will recover and win first during these additional turns.

For these reasons, we believe that the number of instances when it is correct to knock without gin forms a negligible edge case. As such, *our agent only knocks when it has gin*. We call this strategy *stubborn knocking* and compare it to a strategy that knocks as soon as the player has fewer than ten deadwood, which we term *quick knocking*. The latter greedily knocks as soon as possible while the former lazily knocks once it is not worth waiting any longer.

5 Experiments

To assess the performance of specific features of our agent, we devised an experimental setup of five agents. Each agent played in a tournament of 500 games against every agent, including itself. We chose n=500 as it yields a sufficient sample size for hypothesis testing. Each agent has a different combination of the following components: *Stubborn Knocking* (SK, Section 4.5), *Myopic Meld Distance* (MMD, Section 4.3), and *Opponent Modeling* (OM, Section 4.4). Eight different agents are possible by combining these three components, but due to time constraints, only five were tested. Future work could test all eight combinations. Table 1 compares each agent by their features.

5.1 SIGRA, The SIFT Gin Rummy Agent

The SIFT Gin Rummy Agent (`SIGRA`) incorporates all three components: Stubborn Knocking, Myopic Meld Distance, and Opponent Modeling. The agent calculates the MMD for H_i as well as the Opponent-Modeled deadwood reduction for H_o, and then weights them by 0.85 and 1.0 respectively. These weights were determined by an iterative test, but future work could refine these values. The current deadwood value of the agent's hand subtracts the sum of these weighted scores, and the hand with the lowest value is chosen. This combination of components results in a strategy that favors cards that can potentially become part of the opponent's meld, yet also works to reduce the agent's own overall deadwood by making melds if possible. If the agent cannot, then it chooses a lower-deadwood card. The agent has a stubborn knocking policy, which means it only knocks once it has gin in order to try and earn the extra 25 point knocking bonus, otherwise undercutting the opponent for an additional 25 point bonus.

Agent	SK	MMD	OM
SIGRA	✓	✓	✓
StubbornMmd	✓	✓	✗
MmdOpModel	✗	✓	✓
Stubborn	✓	✗	✗
Simple	✗	✗	✗

Table 1: Comparison of Agent Features

Component	SE	Z-Score	P-Value
SK	0.031	4.972	6.613e-7
SK + MMD	0.029	0.477	0.634
SK + MMD + OM	0.028	1.900	0.0575

Table 2: Significance of Incrementally Adding Components

5.2 StubbornMmd

`StubbornMmd` is `SIGRA` without the Opponent Modeling component. The resulting strategy favors cards that will make melds with the agent's hand, or else they choose a lower-deadwood card. Once again, the agent only knocks once they have gin in order to secure the 25-point knocking bonus or the 25-point undercut bonus.

5.3 MmdOpModel

`MmdOpModel` is `SIGRA` with a quick knocking policy instead of a stubborn knocking policy. This combination of components picks cards that could potentially go into an opponent's hand, while prioritizing cards in their own hand second. The agent knocks as soon as possible in order to get only the point difference between each player's hand.

5.4 Stubborn

`Stubborn` only incorporates the stubborn knocking policy. They only draw the face-up card if it immediately becomes part of a meld or creates a new meld. Otherwise, the agent draws the face-down card. To discard, the agent selects the highest-ranked unmelded card. These policies create a play style that neither pays attention to future melds nor to what the opponent is doing. The agent's stubborn knocking policy refrains from knocking until it has gin.

5.5 Simple

`Simple` is `Stubborn` with a quick knocking policy instead of a stubborn knocking policy. We anticipate this agent to be the least successful in the tournament. The 2020 Gin Rummy EAAI Undergraduate Research Challenge provides `Simple` as a baseline (Neller 2020).

5.6 Results

Table 3 displays the experimental results of the 15 possible matches between each of the five agents in the tournament. Take the bottom-left cell as an example. We interpret it as, "`SIGRA` beat `Simple` 74.6% of the time out of the 500 games they played against each other." Figure 2 visualizes this information as well. Directed edges $v_1 \rightarrow v_2$ indicate the *domination* relation such that agent v_1 won the majority of the games against agent v_2. The bottom-left cell from Table 3 is represented in Figure 2 as the edge originating from the node labeled '`SIGRA`' pointing towards the node labeled '`Simple`'. This illustrates that `SIGRA` dominated `Simple`.

Based on Figure 2, we observe that all four agents with at least one of our three strategic components dominated `Simple`. Furthermore, `SIGRA` dominated all other agents, but its win rate over `StubbornMmd` was only 0.51. This result seems to indicate that both agents are evenly matched such that the domination relation may evenly balance closer to 0.5 or potentially flip with more games.

A question that arises from these results is whether adding certain strategic components significantly changes the win rates of our agents. Using `Simple` as the baseline, we hypothesize that incrementally including the SK, MMD, and then the OM components each improves the win rate. For a controlled environment, we consider each agent's 500 matches against `Simple` that lacks all our components.

To test the inclusion of each component, we want to evaluate whether the agent with that component's performance against `Simple` is significantly better than the agent without that component's performance against `Simple`. To do so, we used a two-sided, two-proportion z-test of significance. We will treat both instances of 500 games as the two samples and the win rates from each tournament as the two proportions being compared. The assumptions of this hypothesis test are satisfied as each game has a randomly generated deck shuffling, each game starts with a randomly generated starting hand, and the individual games' shuffled decks are independent of one another. The sample size of $n = 500$ games also greatly improves the likelihood that our win rates will be normally distributed across repeated samples according to the Central Limit Theorem.

For the hypothesis test for the SK component, we evaluate whether `Stubborn`'s performance against `Simple` is significantly better than `Simple`'s performance against itself. Let $P_1 = 0.678$, $P_2 = 0.524$, $n_1 = 500$, and $n_2 = 500$. The hypotheses for this test are as follows: $H_0 : P_1 = P_2$ and $H_A : P_1 \neq P_2$. The results of this significance test can be found in the first row of Table 2. The p-value of this test is 6.613e−7, indicating quite a low probability that we would randomly observe such a difference in win rates if the null hypothesis was true. Similarly, we perform significance tests for the MMD and OM components, the results of which can again be found in Table 2.

Based on the results in Table 2, there is no evidence that the MMD component contributes to our agent's ability to play Gin Rummy effectively if it already uses the stubborn knocking strategy. However, when we combine MMD with OM and SK, there is evidence that our agent is able to play Gin Rummy better than with SK alone. There is also strong evidence that using SK instead of quick knocking improves an agent's ability to play Gin Rummy.

6 Discussion

In the card game Gin Rummy, there are three main decisions that an agent makes per turn: deciding which card to draw, which card to discard, and when to knock. Our proposed

	Simple	Stubborn	Mmd OpModel	Stubborn Mmd	SIGRA
Simple	0.524				
Stubborn	0.678	0.508			
MmdOpModel	0.674	0.000	0.512		
StubbornMmd	0.692	0.524	1.000	0.502	
SIGRA	0.746	0.578	0.990	0.510	0.520

Table 3: Triangular matrix of agent win rates. Win rate represents proportion of games out of 500 that the agent in the left column won against the agent in the top row.

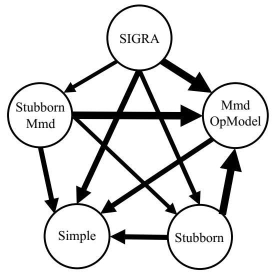


Figure 2: Graph of tournament results. Directed edges denote domination relations, and the edge thickens with respect to win rate.

agent utilizes three components; Myopic Meld Distance, Opponent Modeling, and Stubborn Knocking; to make decisions on these respective choices. The Myopic Meld Distance strategy attempts to score hands based on their potential to have melds in the short term. Opponent Modeling uses the known information in the game in attempts to predict which melds an opponent may be trying to create. The stubborn knocking policy only allows our agent to knock when their hand contains zero deadwood points. By simulating tournaments between agents, we found evidence that each component plays a role in an agent's performance.

Ranking each agent according to its performance is a daunting task, given the limited number of strategies considered in our tournament's design. `SIGRA`, which used all three components, had the greatest win rate against `Simple`. We also saw a significant increase in win rate after adding the SK component alone. This suggests that the stubborn knocking policy on its own is responsible for the agent's improvement, but it is more difficult to narrow down the component(s) that provided the largest benefit because we did not consider all agent variations based on the powerset of the components $\mathcal{P}$\{MMD, OM, SK\}—future work could evaluate such combinations. Based on the increasing win rate as more components are included (of which two appear significant), we believe there is synergy between all three components that creates a more robust strategy. Having fewer components misses complementary strategic benefits.

Although every agent dominated `Simple`, it was not dominated as badly as `MmdOpModel`. It appears the combination of MMD and OM create an inferior agent, but it might be the case that the use and weights of these components' evaluation functions depends on the chosen knocking strategy. For example, OM's preference for deadwood cards that the opponent wants means that they have options for laying off if the agent knocks as soon as possible—this backfires against our current intentions with SK. We hypothesize that an optimal agent would knock without gin when it both is confident that it has less deadwood than the opponent and believes that the opponent is close to achieving gin.

We will also consider dynamically adjusting the weights of our components based on the state of the game. Changing the weights of evaluation functions effectively changes our agent's strategy. Finding the optimal weights throughout a game would thus help the agent perform as well as it can. It might even be possible to personalize these weights per opponent, adapting the agent's strategy during each round of the game. Furthermore, the weights could be changed throughout the tournament, using safer strategies in the first few games and then resorting to more dangerous (but higher reward) strategies if the agent falls behind.

Under the US rules, a stubborn knocking strategy improves our agent's performance. However, the UK rules award fewer points for achieving gin (20) and the undercut bonus (10); it is worth studying if this affects our agent's performance with its reliance on the American rule's 25 points for both. In the future, we also intend to explore how effective our agent's strategies are against a wider variety of opponent strategies. Besides playing AI-driven agents, we wish to investigate how it plays against humans. This extends beyond performance evaluation and into human perception of our agent; would people feel like it is playing "naturally?"

We are further interested in measuring the win rate between two agents, the number of times that each agent knocked, and their scores per round. Does an agent that consistently knocks first, but only earns a handful of points each round outperform an agent that waits until they have gin? What about an agent that only tries to get the undercut bonus? In these two situations, we might see a disproportionate number of knocks and scores. The correlation between these variables could help design more effective strategies for future Gin Rummy-playing agents.

We also plan to consider the cases we omitted via our assumptions in Section 4.1. One assumption worth addressing is when there are multiple ways to optimally assign cards to melds in a given hand. We assumed these situations were rare with minuscule impact, but distinguishing between two different meld sets that have the same immediate deadwood value and different strategic value would be an interesting addition to the theory of Gin Rummy-playing AI.

Acknowledgements

The authors would like the thank the anonymous reviewers for their helpful feedback, Nathan Ringo for providing computational resources to run the experiments, Todd Neller for organizing the Gin Rummy EAAI Undergraduate Research Challenge, and everyone at SIFT for their support.

References

Bard, N.; and Bowling, M. 2007. Particle Filtering for Dynamic Agent Modelling in Simplified Poker. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence*, 515–521. Vancouver, British Columbia, Canada: AAAI Press.

Billings, D.; Peña, L.; Schaeffer, J.; and Szafron, D. 1999. Using Probabilistic Knowledge and Simulation to Play Poker. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference*, 697–703. Orlando, Florida, USA: AAAI Press.

Chaslot, G.; Saito, J.-T.; Bouzy, B.; Uiterwijk, J. W.; and van den Herik, H. J. 2006. Monte-Carlo Strategies for Computer Go. In Schobbens, P.-Y.; Vanhoof, W.; and Schwanen, G., eds., *Proceedings of the Eighteenth Belgium-Netherlands Conference on Artificial Intelligence*, 83–91. University of Namur.

Freedman, R. G.; and Zilberstein, S. 2017. Integration of Planning with Recognition for Responsive Interaction Using Classical Planners. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 4581–4588. AAAI Press.

Greenwald, A.; Campbell, M.; Bowling, M.; Kitano, H.; Kasparov, G.; and Silver, D. 2020. AI History Panel: Advancing AI by Playing Games. Recording available at https://vimeo.com/389556398 by AAAI Livestreaming. Last accessed on 2020-9-05.

Hoehn, B.; Southey, F.; Holte, R. C.; and Bulitko, V. 2005. Effective Short-Term Opponent Exploitation in Simplified Poker. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 783–788. Pittsburgh, Pennsylvania, USA: AAAI Press.

Kabanza, F.; Filion, J.; Benaskeur, A. R.; and Irandoust, H. 2013. Controlling the Hypothesis Space in Probabilistic Plan Recognition. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2306–2312. Beijing, China: AAAI Press.

Neller, T. 2020. Gin Rummy EAAI Undergraduate Research Challenge. http://cs.gettysburg.edu/~tneller/games/ginrummy/eaai/. Last accessed on 2020-9-07.

Russell, S.; and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach*. USA: Prentice Hall Press, 3rd edition. ISBN 0136042597.

Shannon, C. E. 1950. Programming a Computer for Playing Chess. *Philosophical Magazine* 41(314): 256–275.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529: 484–489.

Tastan, B.; Chang, Y.; and Sukthankar, G. 2012. Learning to Intercept Opponents in First Person Shooter Games. In *IEEE Conference on Computational Intelligence and Games*, 100–107. Granada, Spain: IEEE.

Tesauro, G. 1995. Temporal Difference Learning and TD-Gammon. *Communications of the ACM* 38(3): 58–68.