JassTheRipper – A High-Human Artificial Intelligence for the Swiss Card Game Jass

Determinized Monte Carlo Tree Search for 4-Player Schieber

Master Thesis

Joel Niklaus

University of Fribourg

December 2019

Joel Niklaus: *JassTheRipper - A High-Human Artificial Intelligence for the Swiss Card Game Jass, Determinized Monte Carlo Tree Search for 4-Player Schieber*, Master of Science (MSc), © December 2019

SUPERVISION:
Prof. Rolf Ingold *Thesis Director*
Michele Alberti *Supervisor*
Prof. Markus Stolze *External Advisor*
Prof. Thomas Koller *External Advisor*

INSTITITION:
University of Fribourg
Faculty of Science and Medicine
Department of Informatics
Document, Image and Voice Analysis Group (DIVA)

LOCATION:
Bern

DATE:
December 2019

When something is important enough,
you do it even if the odds are not in your favor.

— Elon Musk

This is one of the most important lessons of the scientific method:
if you cannot fail, you cannot learn.

— Eric Ries (The Lean Startup)

ABSTRACT

Despite the recent successful application of Artificial Intelligence (AI) to games, the performance of cooperative agents in imperfect information games is still far from surpassing humans. Cooperating with teammates whose play-styles are not previously known poses additional challenges to current state-of-the-art algorithms. In the Swiss card game Jass, coordination within the two opposing teams is crucial for winning. Since verbal communication is forbidden, the only way to transmit information within the team is through a player's play-style. This makes the game a particularly suitable candidate subject to continue the research on AI in cooperation games with hidden information. To the best of our knowledge, performances of AI agents in the game of Jass do not outperform top players yet. In this work, we analyse the effectiveness and shortcomings of several state-of-the-art algorithms (Monte Carlo Tree Search (MCTS) variants and Deep Neural Networks (DNNs)) at playing the Jass game. Our key contributions are four-fold. First, we provide an overview of the current state-of-the-art of AI methods for card games in general. Second, we discuss their application to the use-case of the Swiss card game Jass. Third, we provide a performance overview for state-of-the-art algorithms, thus, setting a strong foundation for further research on the subject. Fourth, we implement and open-source[1] a platform where different agents (both humans and AI) can play Jass in an automated fashion, effectively reducing the overhead for other researchers who want to perform further experiments. Finally, preliminary results against humans players suggest that our most robust bots (Determinized Monte Carlo Tree Search (DMCTS) for Card Play and DNN for Trump Selection) can play the game at the level of active amateur human players with over 10 years of experience on average.

1 https://gitlab.enterpriselab.ch/jass/info/

PUBLICATIONS

Parts of this thesis are directly taken from the following publications:

[1] Joel Niklaus, Michele Alberti, Rolf Ingold, Markus Stolze, and Thomas Koller. "Challenging Human Supremacy: Evaluating Monte Carlo Tree Search and Deep Learning for the Trick Taking Card Game Jass". In: *Under Review: 19th International Conference on Autonomous Agents and Multi-Agent Systems*. Auckland, New Zealand, 2020.

[2] Joel Niklaus, Michele Alberti, Rolf Ingold, Markus Stolze, and Thomas Koller. "Challenging Human Supremacy: Evaluating Monte Carlo Tree Search and Deep Learning for the Trick Taking Card Game Jass". In: *Association for the Advancement of Artificial Intelligence Workshop on Reinforcement Learning for Games*. New York, United States, Feb. 2020.

[3] Joel Niklaus, Michele Alberti, Rolf Ingold, Markus Stolze, and Thomas Koller. "Comparing Learning and Search Algorithms in the Swiss Card Game Jass". In: *Applied Machine Learning Days*. Lausanne, Switzerland, Jan. 2020.

[4] Joel Niklaus, Michele Alberti, Vinay Pondenkandath, Rolf Ingold, and Marcus Liwicki. "Survey of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass". In: *2019 6th Swiss Conference on Data Science (SDS)*. June 2019, pp. 25–30. DOI: 10.1109/SDS.2019.00-12.

I could either watch it happen or be a part of it.

— Elon Musk

ACKNOWLEDGMENTS

I am greatly indebted to my supervisor Michele Alberti from the University of Fribourg for his exceptional support and guidance all along the way. I highly appreciate his high quality and almost instant advice on Telegram. He provided great help in how to communicate efficiently, in strategic decisions and of course in technical questions. The opportunity to ask him about almost anything and immediately receive a high quality answer and support was invaluable.

Next, I want to thank my thesis director Prof. Rolf Ingold from the University of Fribourg for his valuable feedback and for providing me with the computational resources needed for running the bot and the Jass game server and the infrastructure for conducting my experiments.

Further, I would like to thank my external advisors Prof. Markus Stolze from the University of Applied Sciences in Rapperswil for his valuable inputs, his connections and for participating in the tournament and Prof. Thomas Koller from the University of Applied Sciences in Lucerne especially for his great help and passion with conducting the bot competitions. I enjoyed very much the close and inspiring collaboration with him.

I would also like to sincerely acknowledge the great help of my friends:
First, thanks to Jeanne Kunz for her help with calculating the number of states and information sets in Jass. Second, thanks to Dominik Briner, who designed and implemented the first version of the rule based trump selection, for his advice and ideas with possible improvements for the bot. Third, thanks to Manuel Leuenberger for his great help with Python concurrency issues concerning the gym environment. Finally, thanks to Melissa Chang and Lukas Zbinden for their proofreading and for their valuable comments and feedback.

I also want to express my sincere thanks to all the experiment participants who played against the bot for their curiosity and their survey responses.

I want to sincerely thank the anonymous reviewers for taking the time to read our papers and for their valuable feedback.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

The Author

Joel Niklaus

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

API	Application Programming Interface
REST	Representational State Transfer
GUI	Graphical User Interface
MSE	Mean Squared Error
STD	Standard Deviation
AI	Artificial Intelligence
RL	Reinforcement Learning
SL	Supervised Learning
CFR	Counterfactual Regret Minimization
RB	Rule-Based
NE	Nash Equilibrium
API	Application Programming Interface
AI	Artificial Intelligence
RL	Reinforcement Learning
PIG	Perfect Information Game
IIG	Imperfect Information Game
MC	Monte Carlo
MCTS	Monte Carlo Tree Search
ISMCTS	Information Set Monte Carlo Tree Search
DMCTS	Determinized Monte Carlo Tree Search
P-DMCTS	Probability Determinized Monte Carlo Tree Search
P-ISMCTS	Probability Information Set Monte Carlo Tree Search
T-DMCTS	Time-based Determinized Monte Carlo Tree Search
I-DMCTS	Iteration-based Determinized Monte Carlo Tree Search
UCT	Upper Confidence Bound for Trees
NFSP	Neural Fictitious Self-Play

FOM First Order Methods

EGT Excessive Gap Technique

CFR Counterfactual Regret Minimization

MCCFR Monte Carlo Counterfactual Regret Minimization

OOS Online Outcome Sampling

PG Policy Gradient

PPO Proximal Policy Optimization

TRPO Trust Region Policy Optimization

TDL Temporal Difference Learning

EA Evolutionary Algorithm

RNN Recurrent Neural Network

ANN Artificial Neural Network

DNN Deep Neural Network

SDS Swiss Conference on Data Science

AAAI Association for the Advancement of Artificial Intelligence

AAMAS International Conference on Autonomous Agents and Multiagent Systems

Part I

BACKGROUND

In this part we introduce theoretical foundations and then give background information about the game Jass.

1 INTRODUCTION

In recent years, numerous breakthroughs have taken place in the field of research for AI in games. In particular, in the perfect information games division — where all players are familiar with the entire game state at all times — computers prevail over skilled human players on various occasions, such as Chess [13], the Atari games [39] or Go [59].

When it comes to imperfect information games — where players do not know some of the information, such as in card games — there is a thin line separating AI from people who still have the upper hand over state-of-the-art bots. Hidden information is also present in many real-world scenarios, such as business, negotiations, physics, surgery, and others. Many of these situations can be formalized as games that, in turn, can be solved using the methods developed in the card games testbed.

Recent work has shown that the distance between humans and AI is becoming smaller in constrained situations. This is particularly evident when considering developments on Texas hold'em no-limit poker [7, 9] and the StarCraft II computer game [68].

The multiplayer computer game Dota 2 includes hidden information and team play. Although OpenAI Five won against world champions in a 5 vs. 5 game, collaboration remains as an open research area in AI[1].

To instill AI systems with collaboration, card games may be a very suitable testbed since they a) include hidden information, b) frequently have a collaborative aspect, and c) are computationally easy to simulate because they have a finite set of actions.

There is a large variety of card games, where many use different cards and rules, which poses different challenges to the players. To tackle these different issues, several methods have been proposed. Unfortunately, these methods are often either very complex, or introduce only minor modifications to address a particular issue for a particular game. Despite producing good empirical results, this practice leads to a more complex landscape of literature which is at times hard to navigate, especially for new practitioners in the field. To combat this unwanted side effect, overviews of the current recent trends and methods are very helpful. In this work, we aim to provide such an overview of AI methods applied to card games (also published at Swiss Conference on Data Science (SDS) 2019 [46]). In the appendix there is a short description of the games we mention in this work.

1 https://openai.com/blog/how-to-train-your-openai-five/

1.1 MOTIVATION

Jass is a trick-taking card game featuring hidden information. In the 4-player Schieber variant, good coordination within the team is crucial for achieving victories in top tournaments.

DeepMind introduced the Hanabi Challenge, opening a new frontier in AI research using the fully cooperative card game Hanabi [1]. In Hanabi, the players need to lay down cards in order having only the knowledge of the other players' cards. Therefore, the players need to work together to be able to win the game. Jass combines both a cooperative and a competitive aspect as it includes two competing teams of two cooperating players.

Since there are approximately 1.16*e*28 states in Schieber after the cards have been dealt (see Section 3.3) and additionally it is not known in what state the player is because of the hidden cards, the game is computationally complex.

Jass is a very popular Swiss card game and is closely associated with Swiss culture. It is also similar to other games like the American Spades, the British Bridge, and the German Skat. Thus research in Jass is valuable for many other domains.

However to the best of our knowledge, a formal approach towards Jass has not been addressed in a scientific manner yet. The Swiss Intercantonal Lottery and some Jass applications have deployed some AI agents, but these programs are not yet able to beat top human players.

1.2 CONTRIBUTIONS

- We provide a literature review of the current state-of-the-art in AIs for card games.
- We discuss the potential merits and demerits of the different methods outlined in the paper towards Jass
- We perform an analysis of the most promising state-of-the-art methods for AI in card games (DMCTS, Information Set Monte Carlo Tree Search (ISMCTS), DNN and Rule-Based (RB)).
- We lay the groundwork for further research on AI in Jass.
- We release public open-source software infrastructure (see Section 11.2) and an Application Programming Interface (API), so anyone can quickly connect their bots and test them both against other bots as well as against human via a GUI.

2 THEORETICAL FOUNDATIONS

In this chapter we introduce terms necessary to understand AI for card games.

2.1 GAME TYPES

Games can be classified in many dimensions. In this section we outline the ones most important for classifying card games.

2.1.1 *Perfect Information Games vs. Imperfect Information Games*

In Perfect Information Games (PIGs), like Chess and Go, the players always know the current game state. In Imperfect Information Games (IIGs) on the other hand, there is information hidden from the players - for instance, the cards of other players in Poker or Bridge are examples of hidden information.

Note that perfect information is not the same as complete information. In a game of incomplete information, a player may not know certain rules known by another player. However, games of incomplete information can be reduced to IIGs by introducing moves by nature. For example, the random cards dealt at the beginning of a Poker game by the dealer can be represented as a chance node (move by nature) in the decision tree.

2.1.2 *Cooperative vs. Non-Cooperative*

In cooperative games players can form alliances or agreements to team up against other players or teams. In non-cooperative games this is generally not possible.

2.1.3 *Sequential vs. Simultaneous*

A game is sequential if only one player moves at a time and simultaneous if multiple players can make a move at the same time. Most card games are sequential.

2.1.3.1 *Representation*

Simultaneous games are mostly formalized as normal-form games. These games are represented by payoff matrices which show the

players with their strategies on the axes and the payoffs in the corresponding cells.

2.1.4 *Extensive-form Games*

Sequential games are normally formalized as extensive-form games. These games are played on a decision tree, where a node represents a decision point for a player and an edge describes a possible action leading from one state to another. For each node in this tree it is possible to define an information set.

2.1.4.1 *Information Sets*

An information set includes all the states a player could be in, given the information the player has observed so far. In PIGs, these information sets always only comprise exactly one state, because all information is known. In an IIG like Poker, this information set contains all the card combinations the opponents could have, given the information the player has, i.e. the cards on the table and the cards in the hand.

2.1.5 *Constant-sum vs. Non-constant-sum Games*

In a constant-sum game, there is a limited amount of resources available. So, the total benefit of all players always adds up to a constant. If one player gains some benefit through a move, at least one other player loses some benefit. In a non-constant-sum on the other hand, the amount of available resources may change during the game. Zero-sum games are a special case of constant-sum games where the constant is equal to zero. Well-known examples of zero-sum games are Chess or Go. A constant-sum game can be reduced to a zero-sum game by subtracting a constant from each player's benefit.

2.1.6 *Finite vs. Infinite*

A finite game ends after a finite number of moves. An infinite game may go on forever.

2.1.7 *Coordination Games*

Unlike many strategic situations, collaboration is central in a coordination game, not conflict. In a coordination game, the highest payoffs can only be achieved through team work. Choosing the side of the road to drive on is a simple example of a coordination game. It does not matter which side of the road you agree on, but to avoid crashes, an agreement is essential. In card games, like Bridge or Jass, where

there are two teams playing against each other, the interactions within the team can be seen as a coordination game.

2.2 AI PERFORMANCE EVALUATION

When developing an AI, it is important to accurately measure its strength in comparison to other AIs and humans. The ultimate goal is to achieve optimal play. When a player is playing optimally, s/he does not make any mistakes but plays the best possible move in every situation. When an optimal strategy in a game is known, this game is considered solved. Checkers is an example of a solved game [53].

2.2.1 *Nash Equilibrium*

A Nash Equilibrium (NE) describes a combination of strategies in non-cooperative games. When two or more players are playing their respective part of a NE, any unilateral deviation from the strategy leads to a negative relative outcome for the deviating player [42]. So when programming players for games, the goal is to get as close as possible to a NE. When one is playing a NE strategy, the worst outcome that can happen is coming to a draw. This means that a NE player wins against any player not playing a NE strategy. In games involving chance (the cards dealt at the beginning in the case of Poker), the player may not win every single game. Thus, many games may have to be played to evaluate the strategies. A NE strategy is particularly beneficial against strong players. Therefore, it does not make any mistakes the opponent could possibly exploit. On the other hand, a NE strategy might not win over a sub-optimal player by a large margin because it does not actively try to exploit the opponent but rather tries not to commit any mistakes at all. There exists a NE for every finite game [41, 42].

2.2.2 *Exploitability*

Exploitability is a measure for this deviation from a NE [20]. The higher the exploitability, the greater the distance to a NE, and therefore, the weaker the player. A NE has an exploitability of zero. A NE strategy constitutes optimal play, since there is no possible better strategy. However, there are different NE strategies which differ in their effectiveness of exploiting non-NE strategies [14]. If it is not possible to calculate such a strategy (for example, because the state space is too large), we want to estimate a strategy which minimizes the deviation from a NE.

2.2.3 *Rating Systems*

Player strength in general can be measured with a rating system which compares the player to other players using a score. In Chess and many other zero-sum games the Elo rating system is used to determine the strength of a player.

The Glicko rating system improves on the Elo system. Strength in card games can also be measured with the Elo or Glicko system. However, as there is randomness involved in many card games, the number of games against the same opponent has to be much higher than in PIGs to achieve satisfactory statistical quality for a ranked encounter. Poker AIs have often been measured using exploitability.

In Go there is additionally a ranking system ranging from double-digit kyu (30-20k) to professional dan (1-9p) level.

2.2.4 *Comparison to Humans*

When designing AIs it is always interesting to evaluate how well they perform in comparison to humans. Here we distinguish four categories: sub-human, par-human, high-human and super-human AI which respectively mean worse than, similar to, better than most and better than all humans. The current best AI agents in Jass achieve par-human standards. In Bridge, current computer programs achieve expert level, which constitutes high-human proficiency. In many PIGs like Go or Chess, current AIs achieve super-human level.

3

THE CARD GAME JASS

In this chapter, we briefly introduce the Jass card game together with necessary terminology, discuss its complexity from a mathematical point of view and highlight the most prominent similar games.

3.1 JASS IN A NUTSHELL

Jass is a traditional Swiss card game that is trick-taking and often played at social events. It involves hidden information, both a cooperative (cooperation within the team) and a non-cooperative aspect (two opposing teams), is sequential, finite, and constant-sum (since in each game there are always 157 points).

Figure 3.1: This figure depicts a trick in the Schieber variant. The order of play is counter-clockwise (Harry started with trump diamond 8). Ron had to follow suit with his Diamond 6 and Hermine did not have any Diamonds left, since she played Spades 6. Ginny's card (Diamond Jack) wins this trick because it is the highest trump card. The picture is taken from our Jass server.

The Schieber variant — our testbed — is one of the most widely played variants of Jass in Switzerland. It is played with two opposing teams of two players each. Each round consists of a trump selection phase and a consecutive card play phase. Since choosing a trump is a significant advantage, tournaments are played by a fixed number of rounds (divisible by 4) so that each player can choose trump an equal number of times. Trump selection implies that the selecting player can determine one of the four suits as trump or alternatively no trump with the card precedence top down or bottom up respectively. The player can also decide to pass on the decision to his teammate. This is called "schieben" and gave the name to the game. An example Schieber trick is shown in Figure 3.1.

3.2 TERMINOLOGY

4 played cards are called a *trick*, 9 tricks are a *round* (all 36 cards played) and a *game* lasts for 1000 points, or in tournaments for a number of rounds. When a player *passes* in trump selection, the partner can nominate the trump.

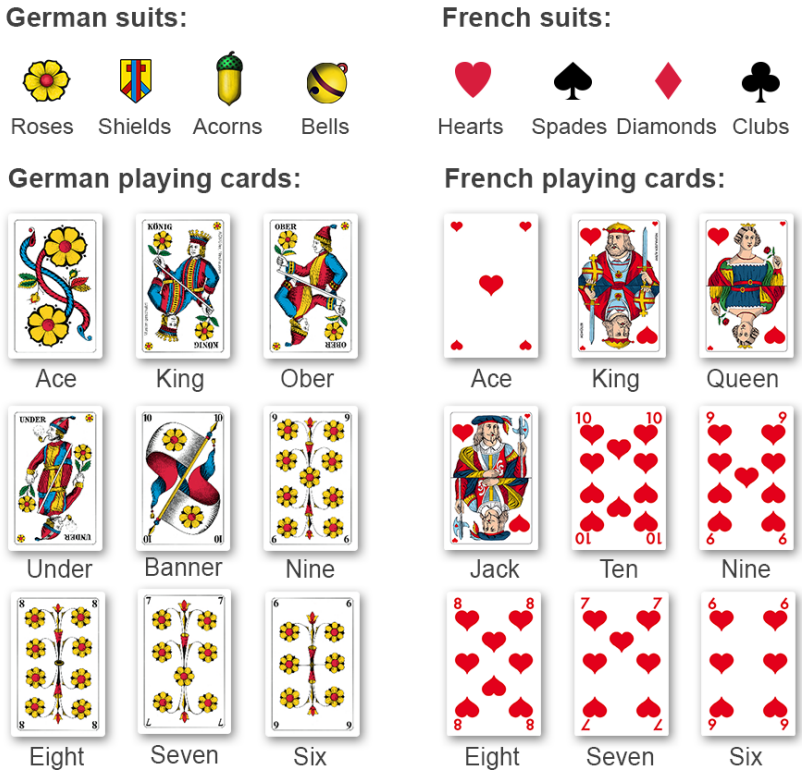

Figure 3.2: This figure depicts the cards used in Jass. In the eastern part of Switzerland the German set is used more often and in the western part the French set is predominant. Image taken from Swisslos.

The Swiss Intercantonal Lottery provides a guide for general Jass rules[1] and for the variant Schieber specifically[2]. They also provide hints and conventions about Jass[3] and the Jass Club Kilchberg also provides hints for how to play the Schieber variant well[4] Figure 3.2 displays the cards used in Jass.

3.3 COMPLEXITY

In Schieber, the number of possible paths through the game tree is $36! = 3.72e41$ since there are 36 cards in the game because every card is only played once, and the order matters.

At the beginning of the game the cards are being dealt randomly to the players. There are

$$\binom{36}{9}\binom{27}{9}\binom{18}{9}\binom{9}{9} = 2.15e19$$

possibilities to distribute the 36 cards to 4 stacks. After the cards have been dealt, each player knows their cards, so the possible distributions of the other cards are $2.28e11$.

To estimate the number of possibilities that a round can be played, we gathered empirical evidence from 1.8 million played rounds (see Section 11.1 for the data) to determine the number of valid plays permitted by the rules for each of the 36 cards played. We found $5.1e16$ possible playouts, so the number of states that an algorithm has to deal with after receiving the cards is in the order of $5.1e16 \cdot 2.28e11 = 1.16e28$.

3.4 SIMILAR GAMES

Jass is very similar to other trick-taking card games like the French Belote, the German Skat and the British Contract Bridge. In Skat, Bridge, and the Sidi Barani variant of Jass, the trump is determined at the start of the game with a process called bidding. Many trick-taking card games have a variant involving four players divided into two teams, like in Schieber.

In contrast to Bridge, Skat, or Hearts, in Jass, it is allowed to play trump at any point in the game, even when the player can still follow suit. This brings more possibilities and thus makes card play harder.

1 www.swisslos.ch/en/jass/informations/jass-rules/principles-of-jass.html
2 www.swisslos.ch/en/jass/informations/jass-rules/schieber-jass.html
3 https://www.swisslos.ch/en/jass/informations/tips-from-the-jass-expert/schieber-jass.html
4 https://jassclubkilchberg.jimdo.com/app/download/8968421285/Tipps+Schieber1309.pdf?t=1551874362

4

RELATED WORK

In this chapter we review the relevant related work. In their book, Yannakakis et al. [75] gave a general overview of AI development in games, while Rubin et al. [51] provided a more specific review on the methods used in computer Poker. In his thesis, Burch [12] reviewed the state-of-the-art in Counterfactual Regret Minimization (CFR), a family of methods very heavily used in computer Poker. Finally, Browne et al. [11] surveyed the different variants of MCTS, a family of methods used for AIs in many games of both perfect and imperfect information. We are not aware of any work that specifically addresses the domain of card games.

Related work about the methods used in card games is presented in the Sections 5, 6, 7 and 8.

Part II

STATE OF THE ART AND BEYOND

In this part we provide a detailed literature overview of the state-of-the-art AIs in card games.

5 RULE-BASED SYSTEMS

In this chapter we give the gist of RB methods and list applications to card games.

Rule-based systems leverage human knowledge to build an AI player [75]. Many simple AIs for card games are rule-based and then used as baseline players. This mostly entails a number of if-then-else statements which can be viewed as a man-made decision tree.

Ward and Cowling [70] created a rule-based AI for Magic: The Gathering which was used as a baseline player. Robilliard, Fonlupt, and Teytaud [49] developed a rule-based AI for 7 Wonders which was used as a baseline player. Watanabe and Lanzi [71] implemented three rule-based players. The greedy player behaves like a beginner player. The other two follow more advanced strategies taken from strategy books and are behaving like expert players. Osawa [48] presented several par-human rule-based strategies for Hanabi. His results indicated that feedback-based strategies achieve higher scores than purely rational ones. Bergh et al. [2] developed a strong par-human rule-based AI for Hanabi. Whitehouse et al. [73] evaluated the rule-based Spades player developed by AI Factory. Based on player reviews they found it to decide weakly in certain situations but to be a strong par-human player overall.

6 REINFORCEMENT LEARNING METHODS

In this chapter we provide a short explanation of Reinforcement Learning (RL) and list applications to card games.

RL is a machine learning method which is frequently used to play games. It consists of an agent performing actions in a given environment. Based on its actions, the agent receives positive rewards which reinforce desirable behaviour and negative rewards which discourage unwanted behaviour. Using a value function, the agent tries to find out which action is the most desirable in a given state.

Rong, Qin, and An [50] use two DNNs with self-play RL to learn a policy which beat the top rule based player in the Bridge bidding phase. The estimation neural network infers the cards of the partner and the policy neural network selects bids based on the public information and the output of the estimation neural network. Tian et al. [67] use two DNNs for learning non-competitive bridge bidding. The belief network models the other agent's private information and the policy network forms a distribution over actions based on the belief network. Foerster et al. [23] establish a new state-of-the-art in two-player Hanabi using a training algorithm they call Bayesian Action Decoder. They measure superior performance than Recurrent Neural Networks (RNNs) and networks trained with Policy Gradient (PG). [35] improve on this and other previous methods by adding search achieving state-of-the-art performance in 2, 3, 4 and 5-player Hanabi.

6.1 TEMPORAL DIFFERENCE LEARNING

Temporal Difference Learning (TDL) updates the value function continuously after every iteration, as opposed to earlier strategies which waited until the episode's end [65].

Sturtevant and White [63] developed a sub-human AI for Hearts using Stochastic Linear Regression and TDL which outperforms players based on minimax search.

6.2 POLICY GRADIENT

PG is an algorithm which directly learns a policy function mapping a state to an action [64, 65]. Proximal Policy Optimization (PPO) [56] is an extension to the PG algorithm improving its stability and reducing the convergence time and improvement by simplification over Trust Region Policy Optimization (TRPO) [55].

Charlesworth [16] applied PPO to Big 2, reaching par-human level.

6.3 COUNTERFACTUAL REGRET MINIMIZATION

CFR [76] is a self-playing method that works very well for IIGs and has been used by the most successful poker AIs [7, 40]. "Counterfactual" denotes looking back and thinking "had I only known then...". "Regret" says how much better one would have done, if one had chosen a different action. And "minimization" is used to minimize the total regret over all actions, so that the future regret is as small as possible. Note that CFR only requires memory linear to the number of information sets and not to the number of states [51]. Additionally, CFR has been able to exploit non-NE strategies computed by Upper Confidence Bound for Trees (UCT) agents in simultaneous games [57].

6.3.1 *Counterfactual Regret Minimization+*

CFR+ is a re-engineered version of CFR, which drastically reduces convergence time. It always iterates over the entire tree and only allows non-negative regrets [4]. Bowling et al. [4] used CFR+ to essentially solve heads-up limit Texas hold'em Poker in 2015.

Moravčík et al. [40] developed a general algorithm for imperfect information settings, called DeepStack. With statistical significance, it defeated professional poker players in a study over 44000 hands.

6.3.2 *Deep Counterfactual Regret Minimization*

Deep CFR [5] combines CFR with deep Artificial Neural Networks (ANNs). Brown et al. [7] leverage deep CFR to decisively beat four top human poker players in 2017 with their program called Libratus. First, they build a blueprint strategy using deep CFR [5] which is applied in the general case. When an action is not in this strategy, or as soon as the remaining game tree is sufficiently small, they apply nested subgame solving to devise a better solution for this particular part of the game [6]. Using this approach they solved a major problem in subgame solving in IIGs, namely that one subgame may depend on other unreached ones.

Depth-Limited Solving [10] makes nested subgame solving already possible in the early game by specifying a maximum depth. Using this approach, comparable performance to state-of-the-art CFR approaches could be achieved using significantly less computing power.

6.3.3 *Discounted Counterfactual Regret Minimization*

Discounted CFR [8] matches or outperforms the previous state-of-the-art variant CFR+ depending on the application by discounting prior iterations.

6.4 NEURAL FICTITIOUS SELF PLAY

In Neural Fictitious Self-Play (NFSP), two players start with random strategies encoded in an ANN. They play against each other knowing the other player's strategy improving the own strategy. With an increasing number of iterations, the strategies typically approach a NE. Since NFSP [26] has a slower convergence rate than CFR it is not widely used.

Heinrich and Silver [26] applied NFSP to Texas hold'em Poker and reported similar performance to the state-of-the-art super-human programs. In Leduc Poker, a simplification of the former, they approached a NE. Kawamura, Mizukami, and Tsuruoka [29] calculated approximate NE strategies with NFSP in multiplayer IIGs.

6.5 FIRST ORDER METHODS

First Order Methods (FOM) like Excessive Gap Technique (EGT) are, like CFR, methods which approximate NE strategies in IIGs. They have a better theoretical convergence rate than CFR because of lower computational and memory costs. Note that, like CFR, EGT is only able to approach a NE in two-player games [31].

Kroer, Farina, and Sandholm [31] applied a variant of EGT to Poker reporting faster convergence than some CFR variants. They argue that, given more hyper parameter tuning, the performance of CFR+ can be reached.

7 MONTE CARLO METHODS

In this chapter we introduce Monte Carlo (MC) based methods and list applications to card games.

MC methods use randomness to solve problems that might be deterministic in principle.

7.1 MONTE CARLO SIMULATION

MC Simulation uses a large number of random experiments to numerically solve large problems involving many random variables.

Mizukami and Tsuruoka [38] developed a par-human AI for Japanese Mahjong using MC Simulation. Kupferschmid and Helmert [32] applied MC Simulation to Skat to obtain the game-theoretical value of a Skat hand. Note that they converted the game to a PIG by making all the cards known. Yan et al. [74] report a 70% win rate using MC Simulation in a Klondike version, which has all cards revealed to the player. Note that this converts the game to a PIG.

7.2 FLAT MONTE CARLO

Flat MC uses MC Simulation, with the actions in a given state being uniformly sampled [11].

L. Ginsberg [33] achieves world champion level play in Bridge using Flat MC in 2001.

7.3 MONTE CARLO TREE SEARCH

MCTS consists of four stages: Selection, Expansion, Simulation and Backpropagation [11]. Selection: Starting from the root node, an expandable child node is selected. A node is expandable if it is non-terminal (i.e. it does have children) and has unvisited children.
Expansion: The tree is expanded by adding one or more child nodes to the previously selected node.
Simulation: From these new children nodes a simulation is run to acquire a reward at a terminal node.
Backpropagation: The simulation's result is used to update the information in the affected nodes (nodes in the selection path). A tree policy is used for selecting and expanding a node and the simulation is run according to the default policy.

Browne et al. [11] give a detailed overview of the MCTS family. In this section we outline the variants used on card games.

7.3.1 *Upper Confidence Bound for Trees*

UCT is the most common MCTS method, using upper confidence bounds as a tree policy, which is a formula that tries to balance the exploration/exploitation problem [30]. When the search explores too much, the optimal moves are not played frequently enough and therefore it may find a sub-optimal move. When the search exploits too much, it may not find a path which promises much greater payoffs and it therefore also may find a sub-optimal move. Minimax is a basic algorithm used for two-player zero-sum games, operating on the game tree. When the entire tree is visited, minimax is optimal [75]. UCT converges to minimax given enough time and memory [30].

Sievers and Helmert [58] applied UCT to Doppelkopf reaching par-human performance. Schäfer [54] used UCT to build an AI for Skat, which is still sub-human but comparable to the MC Simulation based player proposed by Kupferschmid and Helmert [32]. Swiechowski, Tajmajer, and Janusz [66] combined an MCTS player with supervised learning on the logs of sample games, achieving par-human performance. Santos, Santos, and Melo [52] outperformed basic MCTS based AIs by combining it with domain-specific knowledge. Heinrich and Silver [25] combined UCT with self-play and apply it to Poker. They reported convergence to a NE in a small Poker game and argue that, given enough training, convergence can also be reached in large limit Texas Hold'em Poker.

7.3.2 *Determinization*

Determinization is a technique which allows solving an IIG with methods used for PIGs. Determinization samples many states from the information set and plays the game to a terminal state based on these states of perfect information.

Bjarnason, Fern, and Tadepalli [3] studied Klondike using UCT, hindsight optimization and sparse sampling. Hindsight optimization uses determinization and hindsight knowledge to improve the strategy. They developed a policy which wins at least 35% of games, which is a lower bound for an optimal Klondike policy. Sturtevant [62] applied UCT with determinization to the multiplayer games Spades and Hearts. He reported similar performance to the state-of-the-art at that time in Spades and slightly better performance in Hearts. Cowling, Ward, and Powley [19] applied MCTS with determinization approaches to the card game Magic: The Gathering achieving high-human performance and outperforming an expert-level rule-based player. Robilliard, Fonlupt, and Teytaud [49] applied UCT with determinization to 7 Wonders outperforming rule-based AIs. The experiments against human players were promising but not statistically significant. Solinas, Rebstock, and Buro [61] used UCT and supervised learning to infer the cards of the

other players, improving over the state-of-the-art in Skat card-play. Edelkamp [22] combined distilled expert rules, winning probabilities aggregations and a fast tree exploration into an AI for the Misère variant of Skat significantly outperforming human experts.

7.3.3 *Information Set Monte Carlo Tree Search*

ISMCTS tackles the problem of strategy fusion which includes the false assumption that different moves can be taken from different states in the information set [18]. However, because the player does not know of the different states in the information set, it cannot decide differently, based on different states. ISMCTS operates directly on a tree of information sets.

Whitehouse, Powley, and Cowling [72] used MCTS with determinization and information sets on Dou Di Zhu. They did not report any significant differences in performance between the two proposed algorithms. Watanabe and Lanzi [71] presented a high-human AI using ISMCTS for the Italian card game Scopone which consistently beat strong rule-based players. Walton-Rivers et al. [69] applied ISMCTS to Hanabi, but they measured inferior performance to rule-based players. Whitehouse et al. [73] found an MCTS player to be stronger than rule-based players in the card game Spades. They integrated ISMCTS with knowledge-based methods to create more engaging play. Cowling et al. [17] performed a statistical analysis over 27592 played games on a mobile platform to evaluate the player's difficulty for humans. Devlin et al. [21] combined insights from game play data with ISMCTS to emulate human play.

7.4 MONTE CARLO COUNTERFACTUAL REGRET MINIMIZATION

Monte Carlo Counterfactual Regret Minimization (MCCFR) drastically reduces the convergence time of CFR by using MC Sampling [34]. MCCFR samples blocks of paths from the root to a terminal node and then computes the immediate counterfactual regrets over these blocks.

Lanctot et al. [34] showed this faster convergence rate in experiments on Goofspiel and One-Card-Poker. J. V. Ponsen, Jong, and Lanctot [27] evidences that MCCFR approaches a NE in Poker.

7.4.1 *Online Outcome Sampling*

Online Outcome Sampling (OOS) is an online variant to MCCFR which can decrease its exploitability with increasing search time [36].

Lisý, Lanctot, and Bowling [36] demonstrated that OOS can exploit ISMCTS in Poker knowing the opponent's strategy and given enough computation time.

7.5 COMBINING RL WITH MCTS

AlphaZero [60] combines MCTS with self-play RL. This architecture can be applied to imperfect information games as well.

Jiang et al. [28] apply an Alpha-Zero-like architecture to Dou Di Zhu dominating existing Dou Di Zhu programs and coming close to expert human level.

8 EVOLUTIONARY ALGORITHMS

In this chapter we explain the main idea of Evolutionary Algorithms (EAs) and list applications to card games.

EAs are inspired by evolutionary theory. Strong individuals — strategies in the case of game AIs — can survive and reproduce, whereas weaker ones eventually become extinct [75]. According to a fitness function, the strong members of the population (strategies in the case of a card game) are allowed to mate and produce children (crossover). Additional to crossover, some mutation is applied to every child, ensuring certain changes to the population. Hence, the population gradually evolves better and better members.

Mahlmann, Togelius, and Yannakakis [37] compared three EA agents with different fitness functions in Dominion. They argued that their method can be used for automatic game design and game balancing. Noble [47] applied a EA evolving ANNs to Poker in 2002 improving over the state-of-the-art at the time. García-Sánchez et al. [24] used an EA to build decks that were better than human-crafted ones in the collectible card game HearthStone when they were played by their AI.

9 SUITABLE METHODS FOR AI IN CARD GAMES

In this chapter we compare the methods presented in the previous chapters and give recommendations for which methods to use in card games and specifically in trick-taking ones like Jass.

MCTS and CFR are the two families of algorithms that have most successfully been applied to card games.

To the best of our knowledge, CFR has almost exclusively been applied to Poker so far, although the authors claim that it can be applied to any IIG [5]. CFR provides theoretical guarantees for approaching a NE in two player IIGs [76]. On the other hand, as we discussed in section 2.2.1, pure NE strategies may not be able to specifically exploit weak opponents. Additionally, CFR needs a lot of time to converge, compared to MCTS [27].

MCTS has been applied to a plethora of complex card games including Bridge, Skat, Doppelkopf or Spades, as we have illustrated in the previous sections. It finds good strategies fast but only converges to a NE in PIGs and not necessarily in IIGs [27]. As opposed to CFR, MCTS does not find the moves with lowest exploitability, but the ones with highest chance of winning [15]. MCTS eventually converges to minimax, but total convergence is infeasible for large problems [11].

In our overview, we found many use cases for both the ISMCTS and the DMCTS variant.

So, if the goal is to find a good strategy relatively fast, MCTS should be chosen, whereas CFR should be selected, if the goal is to be minimally exploitable [27]. To put it simply, CFR is great at not losing, but not very good at destroying an opponent and MCTS is great at finding good strategies fast, but not very good at resisting against very strong opponents.

DNNs have been used successfully in many AIs for card games in the literature. AlphaZero-like architectures have been proposed where a value function learned by a DNN enhances the MCTS rollouts. Often, DNNs have also been used to provide an estimation of the hidden cards, since knowledge thereof brings the AI closer to a perfect information game making good play easier. Since DNNs have successfully been applied to Bridge [50], we estimate them to be helpful in the game of Jass too.

Part III

EMPIRICAL EVALUATION

In this part we present the experiments we conducted and analyze the results. It has partially been coauthored with Prof. Thomas Koller for a paper currently under review at the International Conference on Autonomous Agents and Multiagent Systems (AAMAS) [43]. The DNN, Iteration-based Determinized Monte Carlo Tree Search (I-DMCTS) and ISMCTS bots have been developed by Prof. Thomas Koller from the University of Applied Sciences in Lucerne. Additionally, he conducted the value estimation experiments and provided the Jass server infrastructure. Big thanks for the inspiring cooperation.

10 IMPLEMENTATION

In this chapter, we document the implementations of our bots.

10.1 TIME-BASED DETERMINIZED MCTS

The implementation of the Time-based Determinized Monte Carlo Tree Search (T-DMCTS) is publicly available on Github[1]. It uses a ranked RB trump selection. If no trump surpasses a given threshold, it passes.

It uses a time budget as a termination criterion for the search, so it can easily be compared to other bots with the same resources. We use *robust child* as final selection policy, choosing the most visited node after the search. It does not use any heuristic function in the tree policy, does not prune branches, does not bound the scores and uses the standard exploration constant $\sqrt{2}$, since different configurations did not improve the performance in our hyper parameter search.

It uses a determinization factor of 15 and runs for 10s per move. This factor multiplied with the number of tricks remaining in the round results in the number of determinizations created. Example: In the third trick, six tricks are remaining (the current trick included). This means that in this trick, it generates $6 \cdot 15 = 90$ determinizations. For each one we create a thread which after the search returns a selected move together with a score for this move. Then we bundle together all of the determinizations for the same moves and average the scores. Multiplying the average score of a move with the number of times it has been selected gives us a final score. This final score is then used for the final selection. This comes with the advantage that not only the number of times a specific move is selected is considered but also the estimated score it is associated with.

With a technique called soft-pruning it would be possible to assign higher probabilities to cards which are likely to be good based on domain knowledge. The MCTS would then take into account the supposedly good cards more which could improve the playing strength if a low number of iterations is used. With a high number of iterations it might limit the algorithm or be not of much use.

10.2 ITERATION-BASED DETERMINIZED MCTS AND INFORMATION SET MCTS

The I-DMCTS uses a configurable number of determinization and MCTS iterations, independent of the time budget, to enable testing of differ-

1 https://github.com/JoelNiklaus/JassTheRipper

ent configurations. Results for different numbers of determinization and iterations are given in Section 13.2. The ISMCTS implementation uses the same framework as the I-DMCTS.

10.3 DEEP NEURAL NETWORK

We implemented two DNNs on the dataset described in Section 11.1. The card play network computes a value of a given state and chooses a card to be played. The trump selection network is used in the beginning in the trump selection phase and chooses one of seven trumps.

10.3.1 *Card Play Network*

We trained a DNN to perform 3 different tasks using Supervised Learning (SL). We a) predict the action a, i.e. the card played by a player as a policy function $p(a|s_o)$, b) the value function $v(s_o)$ and c) the card distribution probability $p_{\text{card}}(\text{player} = i|c)$ that the card c was in the hand of player i at the beginning of the game. The parameter θ describes the weights of the DNN, where as s_o describes the current state of the game observable by the player.

A single convolutional neural network was used with 3 different heads and loss functions. The loss function for the policy head is the cross entropy loss between the predicted action probability p and the actual action a. The loss of the value prediction is the mean squared error between the predicted value v and the actual result z of the game and finally the loss of the card prediction is the sum of the cross entropy losses between the predicted and actual card distributions for each player. $l_{\text{policy}} = \mathbf{a}^T \log \mathbf{p}^a$, where $\mathbf{a}^T$ is the one-hot encoded vector of the action a. The loss of the value prediction is the error between the predicted value v and the actual result z of the game $l_{\text{value}} = (v - z)^2$ and finally the loss of the card prediction is the sum of the cross entropy losses between the predicted card distribution $p_{\text{card}}(\text{player} = 1|c)$ and the actual card distribution $d_i(\text{player} = i|c$ for each player i.

The DNN consists of 6 convolutional layers with 256 channels and kernel sizes of $2x3$ for the first 3 layers, and $1x2$, $1x2$ and $1x2$ for the following layers, whereby each layer reduces the size of the input as valid padding is used. The 6 layers result in a $1x256$ vector that connects to each loss function by a fully connected layer. The total loss is calculated as the weighted sum of the 3 losses, whereas the weights are chosen so that they scale the magnitude of each loss in the same range i.e., each loss will contributes the same.

The input to the DNN is a $4x9$ matrix of 43 channels containing all the information available to the player. This consists of the cards that have been played so far and in which trick and by which player, the

cards in the hand of the current player and the valid cards to play, as well as global information about the game, i.e. who declared trump, how many cards have been played so far and how many points has each team achieved.

This consists of the cards that have been played so far and by which player (4 channels), in which trick (9) and in which order in the trick (4) they have been played, the cards of the current trick (3), the cards in the hand of the player (1), the valid cards to play (1) followed by layers with constant values indicated the selected trump, the player who choose trump and if it was declared forehand (= not passed), the dealer, the current player, the number of tricks and cards played and the points achieved in the game so far.

Training was done for 200 epochs using an Adam optimizer. R2 regularization was enabled on all weights, but no dropout or batch normalization was used. Training achieved an accuracy of 0.776 for the policy, and 0.771 for the card distribution and Mean Squared Error (MSE) of 0.016 for the value function.

Convolutional networks outperformed similar fully connected networks using the same input by 4% in card accuracy, 2% in the card distribution and 10% in the MSE for the value function.

We also trained a second neural network to predict the policy function for the action a $p(a|s)$ and the value $v(s)$ from the complete state s that includes the cards of all the players. This network achieves a much lower MSE on the value function as 8.29×10^{-3} as can be expected. Also, the policy receives a slightly higher accuracy of 0.793, even as the players on which decisions the DNN was trained, did not have the information about the other player's cards available and must, therefore, deduced some of it.

We use the card-distribution prediction in variants of the search algorithms as Probability Determinized Monte Carlo Tree Search (P-DMCTS) and Probability Information Set Monte Carlo Tree Search (P-ISMCTS) to draw cards according to the predicted distribution during determinization. The value function is used as a card play algorithm by evaluating all valid cards and selecting the one with the highest value.

10.3.2 *Trump Selection Network*

Trump selection was trained by a different DNN that uses only the cards in the hand of a player as input, as well if the player is the first or second player of the team to be asked to declare trump. The network consists of two fully connected layers with 592 channels followed by a fully connected layer with 7 channels, which corresponds to the 7 possible actions (4 colors, top-down, bottom-up and passing). The accuracy of the network reached 0.8193 on the validation set. Deeper networks did not perform better.

11 GENERAL EXPERIMENTAL SETUP

In this chapter, we first describe the data sets used for training the DNN and for evaluations, second the technical infrastructure used for carrying out the experiments and finally the tournament setup.

11.1 DATA SETS

Data for training and evaluation was taken from an online platform[1], where users can play the game and either register or play anonymously.

The collected data is from a period of 6 months, starting in October 2017 and consists of about 1.8M played rounds. Each round consists of playing 36 cards, and only completed rounds were taken. The data is split into training, validation, and test sets with a ratio of 0.6:0.2:0.2 by random selection. As plays from the same round are correlated, we further split the files into records for single card plays and shuffle them randomly. From this data set, we filtered out plays by all players that performed less than average, i.e., did not get an average score of 78.5 points. This also eliminates the unregistered players who performed with 78.43 points on average. The resulting data set contains about 14M card plays in the training set and about 4.8M card plays in the test and validation sets.

11.2 TECHNICAL INFRASTRUCTURE

We provide a gym environment for the Schieber variant[2] so RL methods can easily be tested. We publish repositories for a Jass server (deployment[3] and sources[4]) that can run games and tournaments and display the results, as well as a Python development kit to implement algorithms.

Any bot implementing a Representational State Transfer (REST) API[5] can be connected to the Jass server. We also provide a Graphical User Interface (GUI)[6] allowing humans to play on the Jass server[7].

1 https://www.swisslos.ch
2 https://github.com/JoelNiklaus/gym-jass
3 https://jass-server.abiz.ch
4 https://gitlab.enterpriselab.ch/jass/info/
5 https://jasschamp.ch/wp-content/uploads/2019/09/JassInterface.pdf
6 https://github.com/JoelNiklaus/jass-server
7 https://jassteppich.abiz.ch

11.3 TOURNAMENT SETUP

Friendly Jass matches are played until an agreed number of points is reached. Tournaments, however, are usually played for a number of rounds, and the number of points over all rounds are accumulated.

In many card games like Jass, Bridge and Skat, cards are dealt at random in the beginning, and it is much easier to get more points with a good hand than with a bad one. This randomness makes it hard to compare the absolute strength of players.

We address this issue in our experiments by dealing the cards dealt to the North/South pair in the first game to the East/West pair in the second round, which we call a *double round*. We compare the performance of two bots against each other by playing 10 times 100 rounds (= 50 double rounds) and report the mean and the Standard Deviation (STD) of the accumulated score over the 100 rounds. However, for human playing, this feature is disabled, as they would remember the cards and so the cards are dealt entirely at random.

For more reliability, we disabled additional points awarded to card combinations like Melds (Weisen) and Marriages (Stöck) as well as the Matchbonus (100 points if a team wins all tricks in a round).

12 VALUE ESTIMATION COMPARISON

In this chapter we compare different methods to estimate the value of a current game state. We assume that estimating the value better leads to a better overall card play performance.

SETUP

After DMCTS samples a determinization, the algorithm finds itself in a perfect information game situation, so that all the cards are known. To evaluate algorithms in this setting, we omit the sampling and give them the perfect information of the card distribution. The experiment is performed on the validation set (described in Section 11.1).

As a baseline comparison, we plot the value estimation from the DNN, which, however, only uses information about the cards in the hand of the current player. The DNN Max. Policy is used to compute a heavy rollout, resulting in a score of the game at the end of the round. The average of a different number of random rollouts (without any tree search!) is also listed. Finally, different numbers of MCTS iterations with random rollout are shown. To calculate the estimated value, we multiply the probability of an action with the value of that action and sum this for all actions. All the methods have access to the hidden information except DNN Value.

RESULTS

Figure 12.1 shows the results of the different investigated methods. While the improvement from 25 random rollouts to 10 is evident, the improvement of 1000 to 25 random rollouts is only marginal. The DNN value function seems to be comparable to the average of 10 random rollouts. The policy function does not give significantly better results than using random rollout with 1000 iterations or 100 MCTS iterations. The accuracy of the MCTS based value estimation improves clearly with the number of iterations.

ANALYSIS

Random rollouts do not seem to improve much after a number of iterations have been reached, while MCTS continually improves the accuracy with more iterations. We expect that this improved accuracy translates to stronger overall card play. Overall, already 100 MCTS iterations outperform both 1000 random rollouts and the DNN Max

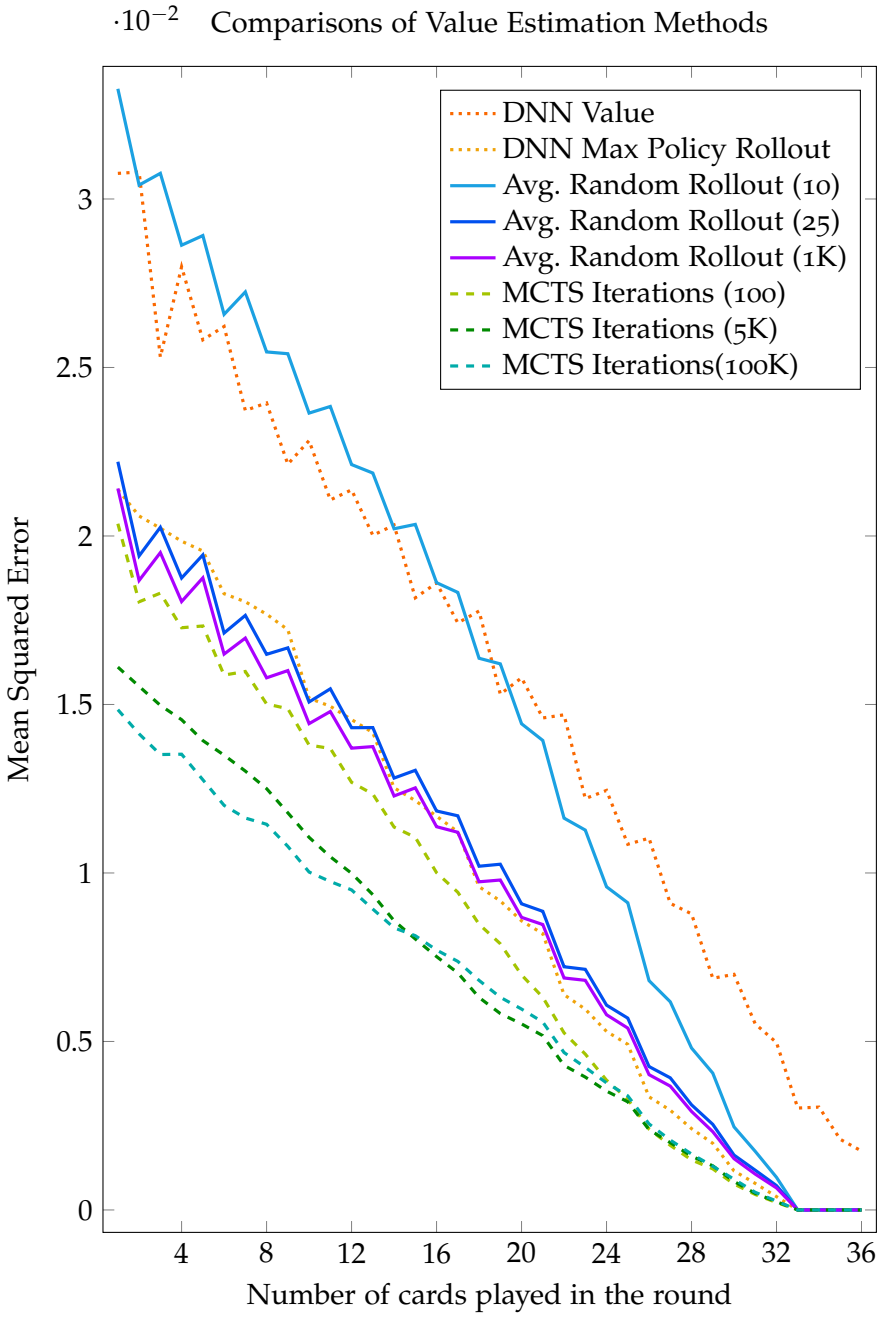

Figure 12.1: We calculated the MSE between the estimated value from the algorithm and the actual outcome at the end of the round. Since the validation set contains 4.8M card plays in total it contains 133K card plays per game stage (number of cards played) on average. Each data point therefore represents the mean of the MSEs of these 133K card plays.

Policy rollout in the perfect information game setting. The difference between the investigated methods is particularly evident in the first few tricks (0 to 24 cards played). Our analyses show that this phase is also the most crucial time in a round. The further the round progressed,

the easier it becomes to play optimally and thus, the smaller the difference between different bots.

13 EXPERIMENTS BETWEEN BOTS

In this chapter we describe the experiments we conducted between different bots.

In Section 13.2 we describe experiments with DMCTS hyper parameters. Since Jass consists of two distinct phases (trump selection and card play) we can also separately evaluate our bots in these two phases, explained in detail in Section 13.3 and 13.4 respectively. In all of these experiments between bots we used double rounds (see Section 11.3) to reduce randomness.

Note that already a small difference in points between two teams can lead to a victory in a Jass game, like in a ski race a difference of 0.5s can bring an athlete from outside the top 20 to winning the gold medal. Therefore, even if there is a considerable amount of variance a slight improvement in the mean performance can be important.

13.1 EXPERIMENT SETUP

In this section we describe the general experimental setup for the experiments between bots. Each experiment consists of playing a game of 100 rounds (= 50 double rounds) 10 times. We report the mean percentage of total points and the STD of the 10 different games. The p-value has been calculated with an unpaired t-test.

13.2 HYPER PARAMETERS FOR DETERMINIZED MONTE CARLO TREE SEARCH

We conducted several experiments to find the best hyper parameters for DMCTS. The factors we investigated were the number of determinizations, the number of iterations and the exploration factor.

13.2.1 *Determinizations and Iterations*

Given a specified number of iterations to be performed, or a specific time constraint, we investigate if it is better to have a larger number of determinizations, thus exploring many different card configurations, or if it is better to devote more resources to the MCTS giving a more accurate result for the cards.

Setup

In this experiment we pitted two I-DMCTS players with different allocations of determinizations and iterations of a total budget against each other.

Results

The first four results in Figure 13.1 (blue) show an overview of the performance of DMCTS with different allocations of a fixed budget against DNN Max Policy. The budget is 800K iterations (e.g., 20K determinizations with 40 iterations each, or 500 determinizations with 1600 iterations each).

Analysis

We find that an increase in the number of determinizations is beneficial up to a certain point ($p = 0.015$ for 1Kx800 and 500x1600). However, further increasing the number of determinizations to over 1K shows no improvement. We find the sweet spot with this particular budget to be at 1K determinizations with 800 iterations each.

13.2.2 *Exploration Constant*

In pure MCTS experiments we have found that an exploration factor of 0.2 gives the best results. In this experiment we investigate the optimal exploration factor for DMCTS.

Setup

In this experiment we pitted two I-DMCTS players with different exploration factors against each other.

Results

The last four results in Figure 13.1 (red) show an overview of the performance of different exploration parameters each with a fixed budget of 1000 determinizations and 1000 iterations each.

Analysis

In DMCTS, much larger exploration values than for pure MCTS result in better performance. We find the standard value of around $\sqrt{2}$ to show the highest mean value ($p = 0.13$ for 0.5 and 1.5) This corresponds with the findings of Browne et al. [11], stating that for perfect information games, very low exploration constants are optimal, but for imperfect information games, the value lies higher.

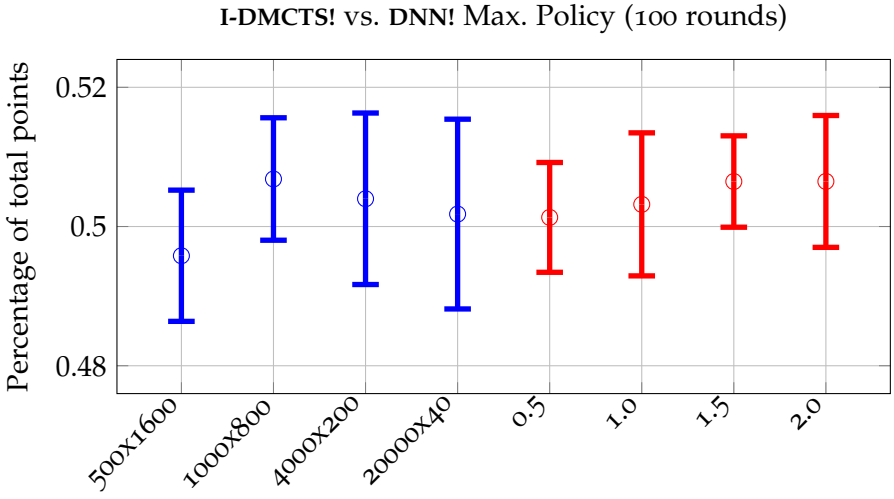

Figure 13.1: Different configurations of I-DMCTS playing against DNN Max. Policy. The first four results (blue) show different ratios between the number of determinizations d and MCTS iterations i ($d \times i$), while the last 4 results (red) show different exploration parameters c each executed with $d = 1000$ and $i = 1000$.

13.2.3 *Scalability*

In Section 12 we saw that a bigger number of MCTS iterations can increase the accuracy of estimating the value at the end of the game. In this section we present an experiment that checks if more iterations and more determinizations really are beneficial to the overall card play strength (measured in percentage of total points).

Setup

In this experiment we pitted two I-DMCTS players with different configurations of determinizations and iterations against each other.

Results

Table 13.1 shows different combinations of iterations and determinizations of DMCTS against DNN which allows us to interpret the scalability properties of DMCTS.

Analysis

With 25 determinizations there is a strong improvement from 25 to 100 iterations ($p < 0.01$). However, our data does not clearly support an improvement from 100 to 10K iterations ($p = 0.29$). Yet, for 100 determinizations, there is an increase from 100 to 10K iterations ($p = 0.023$). Increasing both the determinizations and the iterations clearly has a positive effect on the overall card play strength ($p < 0.0001$ for 25x25 to 1Kx1K). When the number of determinizations or iterations

Table 13.1: Percentage of total points of I-DMCTS playing against DNN with different number of determinizations and iterations and exploration constant 1.5.

Iter.	Determinizations			
	25	100	1K	10K
25	48.07 ± 0.85	48.76 ± 1.58	49.45 ± 1.00	50.02 ± 1.34
100	49.53 ± 1.10	49.92 ± 0.66	49.89 ± 1.21	49.55 ± 0.97
1K	50.11 ± 1.27	50.30 ± 0.93	50.65 ± 0.66	50.75 ± 1.34
10K	49.98 ± 0.71	50.79 ± 0.89	50.38 ± 1.02	very costly

are high though (1Kx1K to 10Kx1K and to 1Kx10K), our data does not support clear claims. It would be very interesting to see how the card play strength changes from 1Kx1K to 10Kx10K and further to 1Mx1M. However, running experiments in these dimensions are very costly (10Kx10K would take 600h (25 days) on a 8 core machine running 16 threads).

Since increasing the number of determinizations and iterations comes with high computational costs it is not a feasible options for running such a highly scaled AI in production. In many algorithms it is possible to trade time (computation) for space (memory). So far though, we have not found a suitable way to do this for DMCTS. However, learning a good policy and representing it in memory – what the DNN does – is a good alternative to the problem of the computational resources at runtime.

13.3 TRUMP SELECTION PHASE

In this section we analyse different trump selection methods.

Setup

To evaluate the trump selection methods, we let four different trump selection methods play against each other while using the same card play algorithm, DNN, for all of them. DNN card play is fast, robust, and deterministic, putting no additional variance into the experiment.

The four trump selection methods we tested were the following: First, the *Random* chooses the trump completely randomly. The *Simple RB* method tries to estimate the number of certain tricks that can be won. The *Ranked RB* implements a ranking algorithm and is used in T-DMCTS. The *MCTS* method considers the trump selection as just another move in the tree to be searched. Finally, the *DNN* performs trump selection as described in Section 10.3.

Table 13.2: Percentage of total points of different trump selection algorithms playing against DNN.

Bot	Result (%)
Random	34.19±2.02
Simple Rule	47.69±0.82
Ranked Based Rule	49.26±1.11
MCTS	48.23±1.98

Results

The results are shown in the Table 13.2 for playing 100 rounds 10 times as described in Section 11.3. DNN achieves the best results, while the more elaborate RB algorithm based on the ranking is only slightly worse ($p = 0.063$ for DNN and Ranked RB).

Analysis

Trump selection proves quite essential, as even a simple algorithm is much better than random selection, so a good bot must combine good trump selection and card play. In Schieber, passing can be very valuable in trump selection, since more information is available afterward. The player who selects trump after the first one passed knows for example that the first player does not have very good cards to choose a trump. So, with passing, the players have another shot at a good trump. We analyzed the choices of the MCTS based trump selection method and noticed that it rarely passes. This may be a reason for it to perform worse than the DNN method ($p = 0.012$).

13.4 CARD PLAY PHASE

In this section we analyse different card play methods.

Setup

To evaluate the different card play algorithms, we let them play against each other with the settings described in Section 11.3. The DMCTS and ISMCTS are the bots as described in Section 7.3. The RB bot[1] is a baseline bot and builds on the Jass Challenge environment released by the Software Engineering company Zühlke[2]. It won the Zühlke Jass Challenge Competition in 2017. The T-DMCTS RB rollouts uses RB

1 https://github.com/Murthy10/pyschieber/tree/master/pyschieber/player/challenge_player
2 https://github.com/webplatformz/challenge

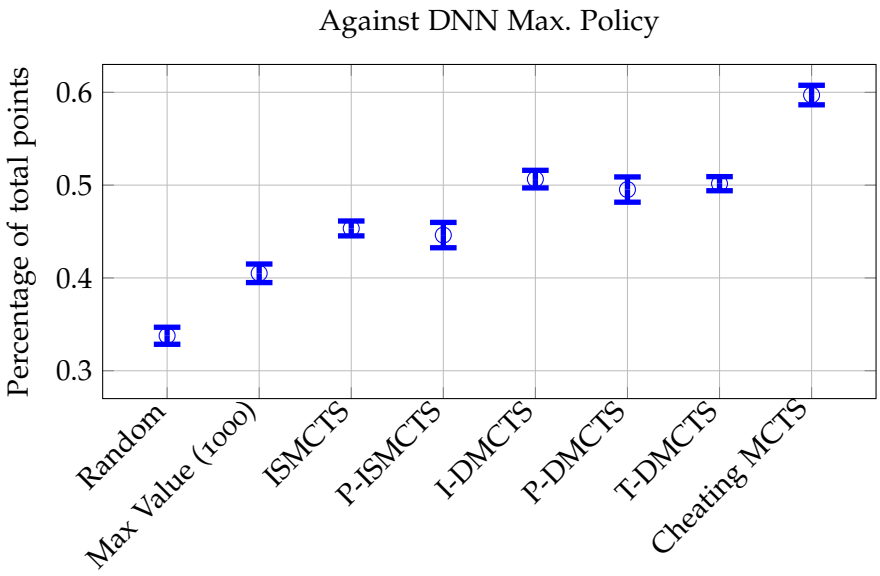

Figure 13.2: Each experiment consists of playing a game of 100 rounds (= 50 double rounds) 10 times, and the average received percentage of total points is shown. The error bar is the STD of the 10 different games.

rollouts instead of random rollouts. The random bot selects a random card while using DNN for trump selection, the Max Value bot evaluates the value network for each valid card out of 1000 card distributions and plays the card with the highest value. The P-DMCTS and P-ISMCTS bots use the probability distributions of the cards from the DNN and draw cards according to this distribution instead of random cards. The cheating MCTS has access to the hidden information (the cards of the other players) and is added as an upper bound.

Results

Figure 13.2 displays the results of the different bots against the DNN method, while Figure 13.3 compares the strength of different bots against the T-DMCTS method. The bots are configured with their best settings; comparisons between different settings are explored more in the following sections.

Analysis

As expected, knowledge of the unknown cards is precious, which can be seen in the big jump in strength by the cheating MCTS player. However, surprisingly, having access to the probability distributions of the cards does not improve the card play strength compared to just sampling random cards ($p = 0.046$ for P-DMCTS and I-DMCTS and $p = 0.17$ for P-ISMCTS and ISMCTS). Rather, the variance increases,

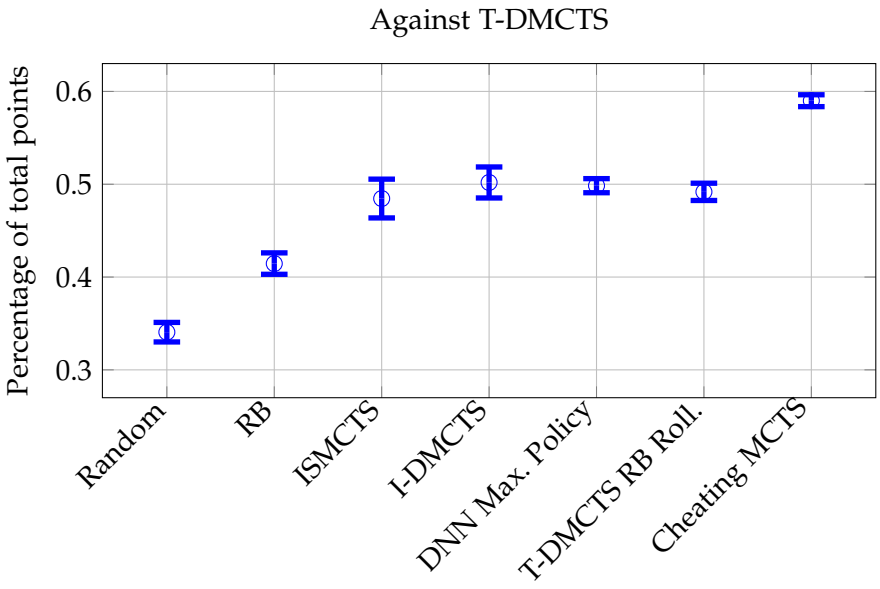

Figure 13.3: Each experiment consists of playing a game of 100 rounds (= 50 double rounds) 10 times, and the average received percentage of total points is shown. The error bar is the STD of the 10 different games.

suggesting that there are both occasions where the DNN guessed the distribution of the cards correctly and others where it did not.

14 EXPERIMENTS AGAINST HUMAN PLAYERS

In this chapter we describe the experiments we conducted against human players.

The goal of these experiments is to a) assess how the bots actually fare against human players and b) to get feedback on how it feels for a human to play in a cooperative fashion with a bot. To this end, we designed two sets of experiments: two human vs. two bots (Section 14.1) and one human and a bot vs. two bots (Section 14.2). In both scenarios the opposing team is composed of two bots of the same type. In all our experiments against human players we used single rounds (as opposed to the double rounds used in the bots-only experiments) since it would be too easy for human players to remember the cards from the previous round and exploit this advantage. At the end of each session, the human player has been asked to provide feedback in which they need to evaluate both their own performance (self-evaluation) and the bots performance (external-evaluation). Given the difficulty – and expensiveness, both in terms of time and money – of collecting large amount of results and the ever-present problem of assessing the true strength of a human player, our results cannot be used to draw a final conclusion on whether humans still have the upper hand in this game against AI.

14.1 HUMAN TEAM VS BOT TEAM

Having a complete human team play against a bot team is arguably the best way to really judge if the bots already have the upper hand against humans. In this section we present our experiment with human teams against teams of T-DMCTS players.

Setup

Conducting experiments involving two humans against two bots poses a number of challenges such as the logistics of having the people being available at the same time and the extra measures that need to be taken to ensure that humans are not cheating (e.g. ensuring players do not communicate with each other, thus gaining an unfair advantage).

In our experiments we had 6 distinct human teams play 136 rounds against a team of T-DMCTS players. It is important to mention that the human teams have played together before and consist of advanced players having 15 years of experience in Jass on average.

Results

The human team scored 49.5% ± 14.2% of the points against the team of T-DMCTS players.

Analysis

Since we cannot use double rounds against humans and the humans may differ in strength, the variance is high in comparison to experiments between bots. However, it seems to be compatible with our data that the T-DMCTS player is able to compete with the human teams.

14.2 HUMAN AND BOT VS BOT TEAM

In a setup of a human with a bot partner vs. 2 bots, it is challenging to assess the impact of the robot-partner on the game outcome. This is because being an excellent partner to a human might be even more complicated than just being an excellent partner to a sibling bot (which will have the same "view" on the game as the bot itself). A good "human-partner-bot" should be playing according to common human practices. This includes "taking into consideration the human partner's way of playing" and establishing legal ways of signaling between the team-players (e.g., discarding policies). These problems are very similar to the ones described in the Hanabi Challenge [1] and are an open area of research in AI.

Setup

We randomly assigned one of the following three bots to be the partner of the human: DNN Max. Policy, I-DMCTS with DNN trump selection and T-DMCTS with RB trump selection. The humans did not know if their partner was human or a bot. In the end, we asked each player if they thought their partner was human or a bot.

Results

Table 14.1 gives an overview of the strength of the three methods against single humans in a total of 960 played rounds. 20% of the players thought their partner was human.

Analysis

In our preliminary results, the DMCTS bots seem to perform slightly better against the humans than the DNN ($p = 0.17$ for DNN and I-DMCTS and $p = 0.0007$ for DNN and T-DMCTS). Interestingly, when its time

Table 14.1: Percentage of total points of different bots against single humans (Higher numbers in result column are better). #Distinct lists the number of distinct human players.

Bot	#Rounds	#Distinct	Result (%)
DNN	68	13	49.79±8.81
I-DMCTS	60	14	52.09±9.85
T-DMCTS	832	47	55.53±13.74

budget is larger, the DMCTS player get more aggressive (smear aces, play trump 9 on an empty trick) and pull trumps more often at the beginning.

14.3 HUMAN PLAYERS FEEDBACK

We received a survey response from 40 of the 74 participants in the experiment, where the humans had a bot partner and played against a bot team. The overall self-assessed level of Schieber competence of the human participants is quite high; 70% rate their ability as "strong" (4) or "very strong" (5) out of 5.

From the participants qualitative feedback we can conclude that humans strongly base their decision whether their partners are a bot (or not) on the time they take to play a card. This was evaluated at different moments in the game: while in the beginning it is human-like to take a long time to think about your move, this is no longer the case when there are only few remaining cards (as the moves are often constrained or very clear). Moreover, human players seem to be sensitive to the way trumps are played i.e., bots are at times deemed to be very aggressive players and occasionally blamed for not pulling all the trumps (Austrumpfen). The humans were positively surprised with the bots using the strategy of playing high value cards when the trick belongs to the partner (Schmieren). In bottom up, a human might play a sequence of 6 to 9 in ascending order. The bots, however, played them in any order, somewhat confusing the humans. Advanced coordination techniques like leading a trick with a suit where the player is strong or signalling a strong suit are not implemented explicitly and are according to the humans also not found implicitly by the bots.

15 ANALYSIS

In this chapter we summarize the most important findings of the experiment chapters.

15.1 VALUE ESTIMATION

For value estimation we investigated the average of different numbers of random rollouts, the DNN Max. Policy and a different numbers of MCTS iterations. The accuracy of using the average of random rollouts increased up until around 25 rollouts but plateaued afterwards. This is also comparable to the accuracy of the DNN Max Policy. Already 100 MCTS iterations outperform both the aforementioned methods. Additionally, 100K iterations still outperformed 5K iterations and it is likely that further increasing the number of iterations will increase the accuracy.

15.2 MCTS HYPERPARAMETERS

We experimented with the number of iterations i and determinizations d and also with different exploration factors c. We found that increasing the number of determinizations and iterations simultaneously leads to overall higher playing strength in our DMCTS bot ($p < 0.0001$ for 25x25 to 1Kx1K). Our data shows that the sweet spot for a budget of 800000 ($d \times i$) is at 1000×800. When using 1000 determinizations and 1000 iterations we found the best exploration factor to be 1.5. Although the variance of these experiments is comparably high, small differences should not be disregarded, since in a tournament a very small difference in points can be responsible for a victory.

15.3 TRUMP SELECTION

We compared different trump selection methods by evaluating the chosen trump games with the deterministic and fast DNN Max. Policy. We found the DNN trump to beat all other investigated trump selection methods. However, the ranked rule based method came a very close second ($49.26\% \pm 1.1\%$ of the points against DNN).

15.4 CARD PLAY

We compared different bots in the card play phase against both the DNN Max. Policy and the T-DMCTS. Both bots clearly outperformed

the random baseline and both also won against the ISMCTS bots. The T-DMCTS also clearly beat the rule-based baseline method. We also observed that an MCTS bot having access to the hidden information clearly beats both the non-cheating DNN Max Policy and the T-DMCTS bot. Replacing the random rollout with a rule-based rollout did not improve the performance of DMCTS. Similarly, using the DNN to estimate the hidden cards and sampling from this distribution did not improve the performance of neither ISMCTS nor DMCTS. Given enough computational resources, the DMCTS is able to beat the DNN Max. Policy by a small margin.

The "Bitter Lesson" from Rich Sutton[1] states that domain knowledge is not a viable approach to AI in the long term but loses against search and learning based methods. Our results confirm this, with the DMCTS and DNN outperforming RB methods.

15.5 HUMAN EXPERIMENTS

We conducted preliminary experiments against humans testing how the bots fare in the real world.

The 6 human teams scored $49.5\% \pm 14.2\%$ of the points against T-DMCTS in 136 played rounds. The human teams had 15 years of experience on average and have played together before. So we conclude that DMCTS is able to compete with experienced amateur human teams. The large STD can be explained through the randomness of the dealt cards.

The experimental setup for only one human is simpler since we do not need to ensure that the human participants do not cheat. This allowed us to collect a bigger dataset here. We found the 47 humans matched up with DMCTS partner to score $44.47\% \pm 13.74\%$ of the points against the DMCTS team in 832 rounds. The mean is 5% smaller than the mean of the human teams. Our data supports the conclusion that the human teams played much stronger than the single humans matched up with DMCTS against a DMCTS team ($p < 0.0001$). A possible explanation might be that the humans expected their DMCTS partner to conform to some conventions, which it presumably did not. This might have led to confusions on the side of the humans, making them play bad moves. Additionally, the human teams might be coordinating together well. Another explanation might be that the humans in the teams experiments are simply better players.

In addition to the played games we also asked the humans to fill in a survey. Their summarized qualitative feedback states that the humans were positively surprised by how well the bots played high value cards when the trick belonged to their partner (Schmieren). They also found the overall playing style of the bots to be very aggressive (trump play

1 http://www.incompleteideas.net/IncIdeas/BitterLesson.html

and smearing aces). Finally they noted that the bots did not adhere to common signalling practices followed by advanced human players.

Part IV

CONCLUSION AND OUTLOOK

In this part we draw conclusions and provide directions for future work.

16 CONCLUSION

In this chapter we summarize the most important findings and list the take home messages.

We provide a literature review of the methods used in AI development for card games. We discuss the advantages and disadvantages of the three most promising families of algorithms (MCTS, CFR and DNN) for trick-taking card games in more detail and we present an analysis for how to apply these methods to the Swiss card game Jass. We provide a comparison of the most widely used methods in trick-taking card games at the example of the Schieber variant of the Swiss card game Jass. In the trump selection phase, empirical evaluation suggests that the DNN slightly outperforms the ranked RB method. In the card play phase, we found that the similarly strong DMCTS and DNN outperform the random baseline, a robust RB bot and also ISMCTS. While there is no clear winner, preliminary results against humans indicate that our best bots (DMCTS) perform on par with the average human players.

17

OUTLOOK

In this chapter we provide possible directions for consecutive research in the near future.

Future work could take the challenging task of recruiting strong human teams for a detailed evaluation of the bots against humans. For this experiment it would be helpful to use two tables simultaneously with the same cards combinations (on one table for the human team and on the other for the bots team). In this way the influence of good hands could be eliminated. Naturally, we would need to perform many of these games to also reduce the bias that a certain card combination might be more favorable for a specific team.

Since we did not see the bots implicitly applying the coordination aspect with humans, future work could try to instill the bots with effective coordination more explicitly.

BIBLIOGRAPHY

[1] Nolan Bard et al. *The Hanabi Challenge: A New Frontier for AI Research*. 2019. arXiv: 1902.00506 [cs.LG].

[2] Mark J. H. van den Bergh, Anne Hommelberg, Walter A. Kosters, and Flora M. Spieksma. "Aspects of the Cooperative Card Game Hanabi". In: *BNAIC 2016: Artificial Intelligence*. Cham: Springer International Publishing, 2017, pp. 93–105. ISBN: 978-3-319-67468-1.

[3] Ronald Bjarnason, Alan Fern, and Prasad Tadepalli. "Lower Bounding Klondike Solitaire with Monte-Carlo Planning". In: *Proceedings of the 19th International Conference on International Conference on Automated Planning and Scheduling*. ICAPS'09. Thessaloniki, Greece: AAAI Press, 2009, pp. 26–33. ISBN: 978-1-57735-406-2.

[4] M. Bowling, N. Burch, M. Johanson, and O. Tammelin. "Heads-up limit hold'em poker is solved". In: *Science* 347.6218 (Jan. 2015), pp. 145–149. DOI: 10.1126/science.1259433.

[5] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. "Deep Counterfactual Regret Minimization". In: *CoRR* abs/1811.00164 (2018). arXiv: 1811.00164.

[6] Noam Brown and Tuomas Sandholm. "Safe and Nested Subgame Solving for Imperfect-Information Games". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 689–699.

[7] Noam Brown and Tuomas Sandholm. "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals". In: *Science* (2017). ISSN: 0036-8075. DOI: 10.1126/science.aao1733. eprint: http://science.sciencemag.org/content/early/2017/12/15/science.aao1733.full.pdf.

[8] Noam Brown and Tuomas Sandholm. "Solving Imperfect-Information Games via Discounted Regret Minimization". In: *CoRR* abs/1809.04040 (2018). arXiv: 1809.04040.

[9] Noam Brown and Tuomas Sandholm. "Superhuman AI for multiplayer poker". In: *Science* 365.6456 (2019), pp. 885–890. ISSN: 0036-8075. DOI: 10.1126/science.aay2400. eprint: https://science.sciencemag.org/content/365/6456/885.full.pdf. URL: https://science.sciencemag.org/content/365/6456/885.

[10] Noam Brown, Tuomas Sandholm, and Brandon Amos. "Depth-Limited Solving for Imperfect-Information Games". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 7674–7685.

[11] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (Mar. 2012), pp. 1–43. ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2012.2186810.

[12] Neil Burch. "Time and Space: Why Imperfect Information Games are Hard". PhD thesis. University of Alberta, 2018. DOI: 10.7939/r36w96q7c.

[13] Murray Campbell, A.Joseph Hoane, and Feng-hsiung Hsu. "Deep Blue". In: *Artificial Intelligence* 134.1 (2002), pp. 57–83. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(01)00129-1.

[14] Jiri Cermak, Branislav Bosansky, and Viliam Lisý. "Practical Performance of Refinements of Nash Equilibria in Extensive-Form Zero-Sum Games". In: *Frontiers in Artificial Intelligence and Applications* 263 (Aug. 2014). DOI: 10.3233/978-1-61499-419-0-201.

[15] H. Chang, C. Hsueh, and T. Hsu. "Convergence and correctness analysis of Monte-Carlo tree search algorithms: A case study of 2 by 4 Chinese dark chess". In: *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. Aug. 2015, pp. 260–266. DOI: 10.1109/CIG.2015.7317963.

[16] Henry Charlesworth. "Application of Self-Play Reinforcement Learning to a Four-Player Game of Imperfect Information". In: *CoRR* abs/1808.10442 (2018). arXiv: 1808.10442.

[17] P. I. Cowling, S. Devlin, E. J. Powley, D. Whitehouse, and J. Rollason. "Player Preference and Style in a Leading Mobile Card Game". In: *IEEE Transactions on Computational Intelligence and AI in Games* 7.3 (Sept. 2015), pp. 233–242. ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2014.2357174.

[18] P. I. Cowling, E. J. Powley, and D. Whitehouse. "Information Set Monte Carlo Tree Search". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.2 (June 2012), pp. 120–143. ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2012.2200894.

[19] P. I. Cowling, C. D. Ward, and E. J. Powley. "Ensemble Determinization in Monte Carlo Tree Search for the Imperfect Information Card Game Magic: The Gathering". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.4 (Dec.

2012), pp. 241–257. ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2012.2204883.

[20] Trevor Davis, Neil Burch, and Michael Bowling. "Using Response Functions to Measure Strategy Strength". In: *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence (AAAI)*. 2014, pp. 630–636.

[21] Sam Devlin, Anastasija Anspoka, Nick Sephton, Peter Cowling, and Jeff Rollason. "Combining Gameplay Data with Monte Carlo Tree Search to Emulate Human Play". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (2016).

[22] Stefan Edelkamp. "Challenging Human Supremacy in Skat – Guided and Complete And-Or Belief-Space Tree Search for Solving the Nullspiel". In: *31. Workshop "Planen, Scheduling und Konfigurieren, Entwerfen"*. 2018.

[23] Jakob N. Foerster, H. Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. "Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning". In: *CoRR* abs/1811.01458 (2018). arXiv: 1811.01458. URL: http://arxiv.org/abs/1811.01458.

[24] Pablo García-Sánchez, Alberto Tonda, Antonio Mora, Giovanni Squillero, and Juan Merelo Guervós. "Automated Playtesting in Collectible Card Games using Evolutionary Algorithms: a Case Study in HearthStone". In: *Knowledge-Based Systems* 153 (Apr. 2018). DOI: 10.1016/j.knosys.2018.04.030.

[25] Johannes Heinrich and David Silver. "Smooth UCT Search in Computer Poker". In: *International Joint Conference on Artificial Intelligence* (2015).

[26] Johannes Heinrich and David Silver. "Deep Reinforcement Learning from Self-Play in Imperfect-Information Games". In: *CoRR* abs/1603.01121 (2016). arXiv: 1603.01121.

[27] Marc J. V. Ponsen, Steven Jong, and Marc Lanctot. "Computing Approximate Nash Equilibria and Robust Best-Responses Using Sampling." In: *J. Artificial Intelligence Res.* 42 (Sept. 2011), pp. 575–605. DOI: 10.1613/jair.3402.

[28] Qiqi Jiang, Kuangzheng Li, Boyao Du, Hao Chen, and Hai Fang. "DeltaDou: Expert-level Doudizhu AI through Self-play". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 1265–1271. DOI: 10.24963/ijcai.2019/176. URL: https://doi.org/10.24963/ijcai.2019/176.

[29] Keigo Kawamura, Naoki Mizukami, and Yoshimasa Tsuruoka. "Neural Fictitious Self-Play in Imperfect Information Games with Many Players". In: *Computer Games*. Cham: Springer International Publishing, 2018, pp. 61–74. ISBN: 978-3-319-75931-9.

[30] Levente Kocsis and Csaba Szepesvári. "Bandit Based Monte-Carlo Planning". In: *Machine Learning: ECML 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293. ISBN: 978-3-540-46056-5.

[31] Christian Kroer, Gabriele Farina, and Tuomas Sandholm. "Solving Large Sequential Games with the Excessive Gap Technique". In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 872–882.

[32] Sebastian Kupferschmid and Malte Helmert. "A Skat Player Based on Monte-Carlo Simulation". In: *Computers and Games*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 135–147.

[33] M L. Ginsberg. "GIB: Imperfect Information in a Computationally Challenging Game". In: *Journal of Artificial Intelligence Research* 14 (June 2001), pp. 303–358. DOI: 10.1613/jair.820.

[34] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. "Monte Carlo Sampling for Regret Minimization in Extensive Games". In: *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., 2009, pp. 1078–1086.

[35] Adam Lerer, Hengyuan Hu, Jakob Foerster, and Noam Brown. *Improving Policies via Search in Cooperative Partially Observable Games*. 2019. arXiv: 1912.02318 [cs.AI].

[36] Viliam Lisý, Marc Lanctot, and Michael Bowling. "Online Monte Carlo Counterfactual Regret Minimization for Search in Imperfect Information Games". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '15. Istanbul, Turkey: International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 27–36. ISBN: 978-1-4503-3413-6.

[37] T. Mahlmann, J. Togelius, and G. N. Yannakakis. "Evolving card sets towards balancing dominion". In: *2012 IEEE Congress on Evolutionary Computation*. June 2012, pp. 1–8. DOI: 10.1109/CEC.2012.6256441.

[38] N. Mizukami and Y. Tsuruoka. "Building a computer Mahjong player based on Monte Carlo simulation and opponent models". In: *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. Aug. 2015, pp. 275–283. DOI: 10.1109/CIG.2015.7317929.

[39] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836.

[40] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker". In: *Science* (2017). ISSN: 0036-8075. DOI: 10.1126/science.aam6960. eprint: http://science.sciencemag.org/content/early/2017/03/01/science.aam6960.full.pdf.

[41] John F. Nash. "Equilibrium points in n-person games". In: *Proceedings of the National Academy of Sciences* 36.1 (1950), pp. 48–49. ISSN: 0027-8424. DOI: 10.1073/pnas.36.1.48. eprint: https://www.pnas.org/content/36/1/48.full.pdf.

[42] John F. Nash. "Non-Cooperative Games". In: *Annals of Mathematics* 54.2 (1951), pp. 286–295. ISSN: 0003486X.

[43] Joel Niklaus, Michele Alberti, Rolf Ingold, Markus Stolze, and Thomas Koller. "Challenging Human Supremacy: Evaluating Monte Carlo Tree Search and Deep Learning for the Trick Taking Card Game Jass". In: *Under Review: 19th International Conference on Autonomous Agents and Multi-Agent Systems*. Auckland, New Zealand, 2020.

[44] Joel Niklaus, Michele Alberti, Rolf Ingold, Markus Stolze, and Thomas Koller. "Challenging Human Supremacy: Evaluating Monte Carlo Tree Search and Deep Learning for the Trick Taking Card Game Jass". In: *Association for the Advancement of Artificial Intelligence Workshop on Reinforcement Learning for Games*. New York, United States, Feb. 2020.

[45] Joel Niklaus, Michele Alberti, Vinay Pondenkandath, Rolf Ingold, and Marcus Liwicki. "Survey of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass". In: *2019 6th Swiss Conference on Data Science (SDS)*. June 2019, pp. 25–30. DOI: 10.1109/SDS.2019.00-12.

[46] Joel Niklaus, Michele Alberti, Vinaychandran Pondenkandath, Rolf Ingold, and Marcus Liwicki. "Overview of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass". In: *6th Swiss Conference on Data Science (SDS)*. Bern, Switzerland: IEEE, 2019, pp. 25–30. ISBN: 978-1-7281-3105-4. DOI: 110.1109/SDS.2019.00-12. arXiv: 1906.04439.

[47] Jason Noble. "Finding Robust Texas Hold'em Poker Strategies Using Pareto Coevolution and Deterministic Crowding". In: *International Conference On Machine Learning And Applications*. 2002.

[48] Hirotaka Osawa. "Solving Hanabi: Estimating Hands by Opponent's Actions in Cooperative Game with Incomplete Information". In: *AAAI Workshop: Computer Poker and Imperfect Information*. 2015.

[49] Denis Robilliard, Cyril Fonlupt, and Fabien Teytaud. "Monte-Carlo Tree Search for the Game of "7 Wonders"". In: *Computer Games*. Cham: Springer International Publishing, 2014, pp. 64–77. ISBN: 978-3-319-14923-3.

[50] Jiang Rong, Tao Qin, and Bo An. *Competitive Bridge Bidding with Deep Neural Networks*. 2019. arXiv: 1903.00900.

[51] Jonathan Rubin and Ian Watson. "Computer poker: A review". In: *Artificial Intelligence* 175.5 (2011). Special Review Issue, pp. 958–987. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2010.12.005.

[52] A. Santos, P. A. Santos, and F. S. Melo. "Monte Carlo tree search experiments in hearthstone". In: *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. Aug. 2017, pp. 272–279. DOI: 10.1109/CIG.2017.8080446.

[53] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. "Checkers Is Solved". In: *Science* 317.5844 (2007), pp. 1518–1522. ISSN: 0036-8075. DOI: 10.1126/science.1144079. eprint: http://science.sciencemag.org/content/317/5844/1518.full.pdf.

[54] Jan Schäfer. "The UCT Algorithm Applied to Games with Imperfect Information". MA thesis. Germany: Otto-Von-Guericke University Magdeburg, 2008.

[55] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. "Trust Region Policy Optimization". In: *CoRR* abs/1502.05477 (2015). arXiv: 1502.05477.

[56] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347.

[57] Mohammad Shafiei, Nathan Sturtevant, and Jonathan Schaeffer. "Comparing UCT versus CFR in simultaneous games". In: *Proceedings of the IJCAI-09 Workshop on General Game Playing (GIGA'09)* (Jan. 2009).

[58] Silvan Sievers and Malte Helmert. "A Doppelkopf Player Based on UCT". In: *KI 2015: Advances in Artificial Intelligence*. Springer International Publishing, 2015, pp. 151–165. ISBN: 978-3-319-24489-1.

[59] David Silver et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

[60] David Silver et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: *CoRR* abs/1712.01815 (2017). arXiv: 1712.01815.

[61] Christopher Solinas, Douglas Rebstock, and Michael Buro. "Improving Search with Supervised Learning in Trick-Based Card Games". In: *CoRR* abs/1903.09604 (2019). arXiv: 1903.09604.

[62] Nathan R. Sturtevant. "An Analysis of UCT in Multi-player Games". In: *Computers and Games*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 37–49. ISBN: 978-3-540-87608-3.

[63] Nathan R. Sturtevant and Adam M. White. "Feature Construction for Reinforcement Learning in Hearts". In: *Computers and Games*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 122–134. ISBN: 978-3-540-75538-8.

[64] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. NIPS'99. Denver, CO: MIT Press, 1999, pp. 1057–1063.

[65] Richard Sutton and Andrew G. Barto. "Reinforcement Learning: An Introduction". In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 9 (Feb. 1998), p. 1054. DOI: 10.1109/TNN.1998.712192.

[66] Maciej Swiechowski, Tomasz Tajmajer, and Andrzej Janusz. "Improving Hearthstone AI by Combining MCTS and Supervised Learning Algorithms". In: *2018 IEEE Conference on Computational Intelligence and Games (CIG)* (Aug. 2018). DOI: 10.1109/cig.2018.8490368.

[67] Zheng Tian, Shihao Zou, Tim Warr, Lisheng Wu, and Jun Wang. "Learning Multi-agent Implicit Communication Through Actions: A Case Study in Contract Bridge, a Collaborative Imperfect-Information Game". In: *CoRR* abs/1810.04444 (2018). arXiv: 1810.04444. URL: http://arxiv.org/abs/1810.04444.

[68] Oriol Vinyals et al. *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. [Online; accessed January 29, 2019]. 2019. URL: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

[69] Joseph Walton-Rivers, Piers R. Williams, Richard Bartle, Diego Perez-Liebana, and Simon M. Lucas. "Evaluating and modelling Hanabi-playing agents". In: *2017 IEEE Congress on Evolutionary Computation (CEC)* (June 2017). DOI: 10.1109/cec.2017.7969465.

[70] C. D. Ward and P. I. Cowling. "Monte Carlo search applied to card selection in Magic: The Gathering". In: *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE, Sept. 2009. DOI: 10.1109/cig.2009.5286501.

[71] Marcos Norio Watanabe and Pier Luca Lanzi. "Traditional Wisdom and Monte Carlo Tree Search Face-to-Face in the Card Game Scopone". In: *IEEE Transactions on Games* 10 (2018), pp. 317–332.

[72] D. Whitehouse, E. J. Powley, and P. I. Cowling. "Determinization and information set Monte Carlo Tree Search for the card game Dou Di Zhu". In: *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. Aug. 2011, pp. 87–94. DOI: 10.1109/CIG.2011.6031993.

[73] Daniel Whitehouse, Peter I. Cowling, Edward J. Powley, and Jeff Rollason. "Integrating Monte Carlo Tree Search with Knowledge-based Methods to Create Engaging Play in a Commercial Mobile Game". In: *Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AIIDE'13. Boston, MA, USA: AAAI Press, 2013, pp. 100–106. ISBN: 978-1-57735-607-3.

[74] Xiang Yan, Persi Diaconis, Paat Rusmevichientong, and Benjamin V. Roy. "Solitaire: Man Versus Machine". In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, 2005, pp. 1553–1560.

[75] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. http://gameaibook.org. Springer, 2018.

[76] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. "Regret Minimization in Games with Incomplete Information". In: *Advances in Neural Information Processing Systems 20*. Curran Associates, Inc., 2008, pp. 1729–1736.

Part V

APPENDIX

In this part we add additional content for the interested reader.

A

DESCRIPTION OF MENTIONED GAMES

In this chapter we give the gist of the less well-known games discussed in the paper (in order of appearance).

Magic: The Gathering is a trading and digital collectible card game played by two or more players. *7 Wonders* is a board game with strong elements of card games including hidden information for two to seven players. *Scopone* is a variant of the Italian card game Scopa. Scopa is normally played with two or four players but there also exist variants for three, five or six players. *Hanabi* is a French cooperative card game for two to five players. *Spades* is a four player trick-taking card game mainly played in North America. *Big 2* is a Chinese card game for two to four players mainly played in East and South East Asia. The goal is, to get rid of all of one's cards first. Tichu, a game popular in Switzerland, is a variant of Big 2. *Mahjong* is a traditional Chinese tile-based game for four (or seldom three) players similar to the Western game Rummy. *Skat* is a three player trick-taking card game mainly played in Germany. *Klondike* is a single-player variant of the French card game Patience and shipped with Windows since version 3. *Bridge* is a trick-taking card game for four players played world-wide in clubs, tournaments, online and socially at home. It has often been used as a test bed for AI research and is still an active area of research, since super-human performance has not been achieved yet. *Doppelkopf* is a trick-taking card game for four people, mainly played in Germany. There exist variations for three to seven players. Doppelkopf is similar to Skat, both originating from the game Schafkopf. *Hearthstone* is an online collectible card video game, developed by Blizzard Entertainment. It is played turn by turn by two opponents using a constructed deck of 30 cards. *Hearts* is a four player trick-taking card game, mainly played in North America. *Dou Di Zhu* is a Chinese card game for three players. *Goofspiel* is a simple bidding card game for two or more players. *One-Card-Poker* generalizes the minimal variant Kuhn-Poker. *Cribbage* is a traditional English card game for two to four players, played with the normal 52-card Poker set. *Dominion* is a modern deck-building card game similar to Magic: The Gathering.

B

REVIEWS

In this chapter we add the reviews we received from our peers. We would like to take this opportunity to thank them for their valuable feedback.

B.1 SWISS DATA SCIENCE CONFERENCE 2019

In this section we list the three reviews received for the paper accepted at the Swiss Data Science Conference in June 2019 [45].

B.1.1 *Review 1*

SCORE: 2 (accept)
——— Overall evaluation ———
AI development for games has seen steady improvements for several decades - Deep Blue, for instance, dates back more than 20 years, as do many sophisticated PC game engines including some card games. The word "decade" should definitely appear in plural in the abstract of this work. Apart from this detail, the scope of the text is of interest because it clearly covers two needs in a single paper: a niche with a game not covered by existing literature, and a survey about the field.

The introduction could mention that as opposed to card games, general game playing engines are quite successful for board games and other types of games.

The related work could be elaborated on, for instance shedding some light on who is doing this game research - one is a book and one is a thesis, hence there seem to be few conference/journal articles on this topic.

Section V on reinforcement learning would need some polishing. For instance, V-A talks about "updates the value function" without referring to what the value function is.

Sections IV to VII could be compressed to gain additional analytical explanations for the use case of Jass. Currently, this reads like "we think this is the case" whereas a more substantiated argumentation would strengthen the choice of MCTS/CFR.

The presentation structure could be improved to avoid single subsections (e.g. I-A main contribution). The term man-made should probably read human-made.

From the perspective of somebody who spent a lot of time in the past in game development including AIs, this is an enjoyable read

with little need for conference-level improvements and is therefore recommended for publication.

B.1.2 *Review 2*

SCORE: 1 (weak accept)
——— Overall evaluation ———
This paper gives a survey of the current state of the art of AI for card games, and applies it to the Swiss game of Jass (a game of imperfect information, thus solutions are relevant for several real use cases apart from games).

The survey part is very well done, informative and readable; some sort of table or figure would help just to break up the huge wall of text.

The application part to Jass is rather disappointing: it shows no experimental results, just a "quick" theoretical analysis. More "flesh to the bone" would have been nice.

- The paper is generally very well written. Areas to improve could be: add more references to the forst paragraph of Sec. I; add more (visual) structure (in terms of paragraphs) to the appendix; in Sec. VIII.B, in the phrase "To put it simply" doesn't it need to be "simple" instead of "simply"?

- Sec. II is quite brief and only focuses on overview articles; how about relevant related work that focuses on single approaches? Of course you treat them later in your own survey, so for the reader, this section really comes in a little strange. Consider integrating it into Sec. I altogether. Also consider using the term "Survey" in the title rather than "Overview" (would make the paper better findable).

- The conclusion is a mere summary and written in present tense rather past tense; it should be replaced by real conclusions.

B.1.3 *Review 3*

SCORE: -1 (weak reject)
——— Overall evaluation ———
The paper provides a sound overview about different approaches using AI to play (card) games.

However, I am a bit puzzled what to take out of the article. While I highly appreciate the comprehensive overview of techniques applied, the part on formalizing or model the Jass game seems to be short. A classification / characterization of the problem class in which Jass seems to be is given, but further modeling of the problem is missing. However, this would have been an initial step towards selecting a technique for providing a Jass playing system.

Regarding some games the references are a bit outdated. In particular planning and search based approaches, e.g. for Skat have improved

and now provide competitive performance, and are, probably also closer to the problem class of Jass than Poker is.

B.2 AAAI WORKSHOP ON REINFORCEMENT LEARNING FOR GAMES 2020

In this section we list the two reviews received for the paper accepted at the Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Reinforcement Learning for Games in February 2020 [44].

B.2.1 *Review 1*

SCORE: 2 (accept)
——— Overall evaluation ———
In my judgement this paper is a good fit for the workshop. Even though it does not show any algorithmic innovation it presents nice empirical evaluation of well known methods applied to this domain. Most of the infrastructure is open-sourced which is also valuable contribution to the community.

Minor notes:

- We compare MCTS and DNN with human players - > should this be: "We compare ISMCTS and DNN with human players"?
- The DNN Max. Policy is -> The DNN Max policy is
- What is 'heavy rollout'? Can you define it in the text? After reading section 7 I don't exactly know what DNN Max is even though it is important for the rest of the paper. Can you describe it in more detail?
- Figure 2 should be organized better, it is now difficult to separate different families of methods (Rollout/MCTS/DNN).
- Would it be possible to publish also the training data?

B.3 REVIEW 2

SCORE: 2 (accept)
——— Overall evaluation ———
This paper seems to be well in scope of the workshop and discusses the card game JASS. It does a good job in introducing the game (though I already know it, so not sure how clear it is for newcomers), and I kind of like to see some RL work happening in this area. I am not sure how much the paper really contributes to the SOTA as it mostly gives an overview of what has been done in the area and does some experimental comparisons of some of the techniques that have

been already investigated in trick-taking games. I guess the authors could place their work (the experiments) a little bit more into context related to other games where some of the presented techniques have been tested before. It is also nice the authors have open-sourced their implementation (though I wasn't able to verify).

DECLARATION

I declare on my honour that my graduation work is a work of personal art, composed without unauthorized external assistance.

Bern, December 2019

Joel Niklaus