Application of Reinforcement Learning to the Card Game Wizard

Jana Cathrin Backhus
Hokkaido University
Sapporo, Japan
jana@main.ist.hokudai.ac.jp

Hidetoshi Nonaka
Hokkaido University
Sapporo, Japan
nonaka@main.ist.hokudai.ac.jp

Takeshi Yoshikawa
Hokkaido University
Sapporo, Japan
yosikawa@main.ist.hokudai.ac.jp

Masanori Sugimoto
Hokkaido University
Sapporo, Japan
sugi@ ist.hokudai.ac.jp

***Abstract*—This article proposes an application using a reinforcement learning (RL) approach to the card game Wizard. The aim is to create a computer player that is able to learn a winning strategy for the game by himself. Wizard is a partially observable competitive multiplayer game that consists of two game phases, forecasting and trick playing. The biggest challenges in creating a strong player are dealing with multiple rounds which have a different grade of imperfection and the decision on the forecast at the beginning of every game round. We introduce an RL approach to the problem by adopting an existing RL algorithm to the playing phase of the game and by implementing an evaluator of the player's hand card using a Multi-Player-Perceptron to conduct the forecast. The results of our experiments show that the player is able to improve his playing strategy through learning. At the beginning the performance of the learning agent is very bad due to the bad forecasting behavior, but he is able to improve his performance over a few training episodes from 0% won games to approximately 25.68% won games in an experiment with 4 players. Therefore he plays equally strong as his opponents and even outperforms one of them.**

Keywords—multi-agent; reinforcement learning; competitive game; incomplete information

I. INTRODUCTION

Games have been platforms of interest since the first days of computer research. One of the aims of game research is creating strong computer players for the entertainment of human players. Another aim is to create human-like learning algorithms that can be generalized and applied to a broad number of applications. This makes games an important research topic in the field of artificial intelligence. Earlier objects of interest were fully observable games like chess or checkers. It was expected that creating strong computer players for these games would make it possible to create computers that can act like humans. But it was possible to create strong computer players without imitating human-like thinking, so more complicated games like Shogi (Japanese chess), Go or card games became the new object of interest.

Card games are considered to be more complicated, because they are only partially observable and most of them are played with multiple players, which provide new challenges to the researchers. The partial observability of the game state makes it necessary to handle unknown environment states and multiple players make it impossible to use effectual search algorithms, which are often used in applications for two-player games. So other methods have to be investigated.

Also, because of their attributes, card games are considered ideal test beds for real world situations. The partial observability of the state and the fact that the *learning agent* (LA) has to handle multiple opponents are similar to real world situations.

Research for card games has been conducted, but still existing approaches are not as successful as those for completely observable games. There have been applications of learning algorithms to diverse card games, e.g. Bridge[1], Hearts[2], Spades[3] and Skat[4]. Up until now, there are not many approaches that are able to play at an expert level. The Bridge card game player developed in [1] is one of the few computer players that are able to play at an expert level. Applications for Hearts, Spades and Skat followed, but due to the games' nature, it turned out to be more difficult to create strong computer players and so research is still ongoing.

In this paper, we introduce a reinforcement learning algorithm to the card game Wizard. Our approach is divided into two parts, one for the forecast and the other for the playing phase of the game. We implemented a Multi-Layer-Perceptron for the forecast phase that is evaluating the learning agent's hand cards and gives back an appropriate forecasted number of winnable tricks. The Multi-Layer-Perceptron is trained online by our approach. For the playing phase, we adopted an existing RL approach and fit it to our problem. The results of our conducted experiments show that the player is able to improve his playing strategy. He is able to improve his performance from 0% won games to 25.68% against 3 opponents in a 4 player game, which makes him equally strong as his opponents and he is even able to outperform one of them.

This paper proceeds as follows: first of all we describe the introduced method in section II, and then present our conducted experiments and results in section III. Finally, we draw a conclusion in section IV.

II. METHOD

In the following the implemented reinforcement learning algorithm is explained after a general introduction to the game rules of Wizard.

978-1-4799-0892-9/13/$31.00 ©2013 IEEE

A. Game Rules

Wizard is a competitive card game that can be played with 3 to 6 players. The card deck consists of 60 cards with 52 cards in the 4 basic card colors and 8 special cards. The special cards consist of four wizards, the strongest card of the game, and four jesters, the weakest card of the game. One game is conducted in a number of rounds, where the number is depending on the number of players. The dealt cards per player are increasing from round to round, starting with one card per player in the first round and finishing with all cards being dealt to the players in the last round. Except for the last round, a trump color is determined from the cards that were not dealt to the players by opening the top card of the remaining card deck.

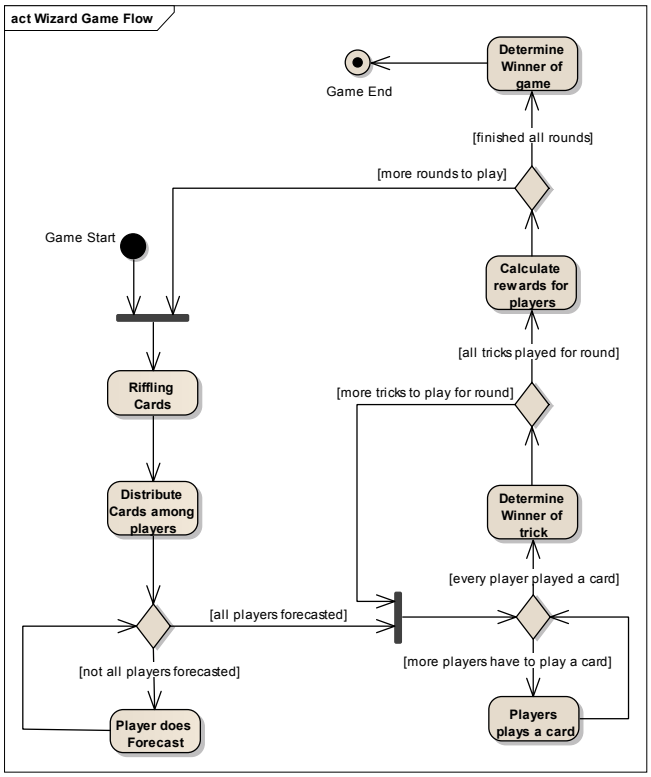

Fig. 1. Game flow of Wizard

The flow of the game is presented in Fig. 1. At the beginning of each round after receiving the own hand cards, every player has to forecast how many tricks he is going to make in the round. At every time step of a round, every player plays one card and the player of the strongest card wins all the played cards, where these cards are called a trick. Only players which fulfill their forecast by winning the correct amount of tricks are getting a positive game reward which consists of 20 basis points and 10 more points for every trick in the forecast. So in case of a forecast of 0 tricks, the player will get 20 points, for 1 forecasted trick 30 points and so on. All the other players, who did not achieve their forecasted number of tricks, get a negative game reward that is calculated from the difference between the forecast and the number of won tricks. At the end of a game, all game round rewards are summed up and the player with the highest points is determined as the winner. Considering the game flow as described above, generally the game can be divided into two phases, forecast and play. The play phase is related to the forecast phase, because in the play phase the forecast has to be fulfilled by winning the right amount of tricks to get a positive reward. Therefore an appropriate playing strategy is needed.

B. Learning Method

In strategy games like Wizard and many other card games, it is important to develop a strategy that is adapting to the actual game situation. This is necessary because of the partial observability of the state and the fact that the players have no influence over which cards are distributed to them. Therefore *reinforcement learning* (RL) is likely to be a suitable method to make the computer learn a strategy to evaluate the actual game state by itself. Fujita et al. introduced an RL algorithm [5] that describes the problem as a partially observable Markov decision process, which evaluates the actual game state and chooses the action that has the best utility for that game state. Function approximation is necessary, since due to the partial observability of card games the state space is too large to be computed in the whole. The introduced algorithm uses particle filtering for sampling possible current and next time step's game states.

The state samples are then used to calculate the utility of an action at a time step *t*. Fig. 2 presents a pseudo code for the calculation of this utility. The probability of the current sample state and the next sample state are calculated to consider the likelihood of these states under the available information. The immediate reward equals the reward that the player would get from the environment for taking an action when the game state transits from the current sample state to the next sample state. The state evaluator of the next sample is a *Multi-Layer-Perceptron* (MLP) that is trained by the reinforcement learning algorithm and returns a state value for the next time step state sample.

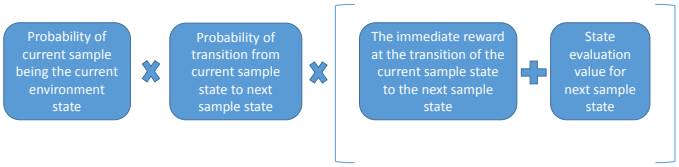

Fig. 2. Pseudo code for utility calculation of an agent action

At every turn of the learning player, the utility for every possible action is calculated using a fixed number of sample states and the action with the highest utility is chosen.

The authors also make use of feature extraction to reduce the state space and to reduce computing time. We applied this RL approach to the play-phase of Wizard. To adapt it to our problem appropriate features had to be constructed considering the characteristics of the game.

There are two types of constructed features for the playing phase of the application. First, features for the approximation of the state value function are needed. Second, opponent agent action predictors are used to predict likely opponent actions that help to make the next state sampling more accurate. Each

action predictor is approximated with one MLP for an opponent. Here, feature extraction is used to help to reduce the input as well as the output state space of the function.

The features for the state value function are constructed as follows. They contain information about the hand cards of every of *M* players.

- (7**i*+1): Number of wizards in agent *i*'s (*i*=0,…,*M*-1) hand
- (7**i*+2): Number of jesters in agent *i*'s hand
- (7**i*+3): Number of trump color cards in agent *i*'s hand stronger then strongest card in current trick
- (7**i*+4): Number of trump color cards in agent *i*'s hand weaker then strongest card in current trick
- (7**i*+5): Number of other color 1 cards in agent *i*'s hand
- (7**i*+6): Number of other color 2 cards in agent *i*'s hand
- (7**i*+7): Number of other color 3 cards in agent *i*'s hand
- (7**M*+1): Number of cards still needed to be won by LA
- (7**M*+2): Trump color
- (7**M*+3) to (7**M*+3+*M*): Bit sequence for the playing order (with 1 if it is LA's turn, otherwise 0)

The features are aiming at an appropriate presentation of the actual game state by giving important information about the hand cards of every player and reducing information volume where it is not needed. So for these features, wizard, jester and trump color cards are considered important, while the other color cards are considered to be not as important and therefore get less dimensions of the extracted feature state space.

In a similar way, the features for the opponent agent's action predictors were extracted as follows for the *M*-1 opponents (*M*=number of players).

- (1): Number of blue cards in opponent agent's hand that are weaker than the strongest card in the current trick
- (2): Number of blue cards in opponent agent's hand that are stronger than the strongest card in the current trick
- (3): Number of green cards in opponent agent's hand that are weaker than the strongest card in the current trick
- (4): Number of green cards in opponent agent's hand that are stronger than the strongest card in the current trick
- (5): Number of red cards in opponent agent's hand that are weaker than the strongest card in the current trick
- (6): Number of red cards in opponent agent's hand that are stronger than the strongest card in the current trick
- (7): Number of yellow cards in opponent agent's hand that are weaker than the strongest card in the current trick
- (8): Number of yellow cards in opponent agent's hand that are stronger than the strongest card in the current trick
- (9): Number of wizards in the opponent agent's hand
- (10): One bit that is 1 if wizard is stronger than strongest card in trick, otherwise 0 (only 0 if strongest card is a wizard)
- (11): Number of jester in the opponent agent's hand
- (12): Number of tricks that opponent still needs in this round
- (13) to (13+*M*): Bit sequence for the playing order

These features are constructed for an opponent player's action predictor and need to calculate the action selection probability of a possible opponent action. So, the features are extracted to describe the opponent's hand as good as possible dividing every color card type in stronger and weaker cards.

In the following, we introduce a method for the forecasting phase of our card game. The forecasting of game results in card games is less researched than playing tricks and therefore is said to be the weakness of card games when such a phase is included [4]. Many applications use only rule-based approaches as for example in the Bridge application in [1]. For the forecast in Wizard, we implemented a mixed solution using a rule-based approach in the former half of the game and an MLP as an evaluator for the LA's hand cards in the latter half of the game. The MLP was trained by reinforcement learning using the real number of won tricks at the end of every game as the training's output results.

Here, again we made use of feature extraction to merge the size of the input space into one, because in every round the hand card size is different, and to define the hand cards by its characteristics. Important cards like wizards, jesters and trump colors are given more dimensions of the feature state space, because there are the strongest cards (wizards, trump cards) and therefore likely to win a trick or strategically important cards (jesters). The rest of the cards are summarized into a smaller part of the feature state space. The trump color is given one dimension of the feature space, because there are also game rounds where there is no trump color, and then the other color cards become more important. There is a differentiation between being first in the playing turn or not, because being first makes it easier to win a card since the first player in a trick is deciding the leading color of the trick with his played card.

The detailed features for the forecast are as follows.

- (1): Number of wizards
- (2): Number of jesters
- (3): Number of trump color cards bigger than 10
- (4): Number of trump color cards bigger than 6
- (5): Number of trump color cards equal or smaller than 6
- (6): Number of other color cards bigger than 10
- (7): Number of other color cards bigger than 6
- (8): Number of other color cards equal or smaller than 6
- (9): Trump color
- (10): Playing turn at first time step (1 if first, else 0)

Since we are applying two approaches to the forecast, we had to decide which forecast approach is used in any of the game rounds. We tested both approaches for every game round and decided on the more successful one. For example, in a 4 player game, rule-based approaches turned out to be more successful than an MLP approach for the first 8 rounds. For the latter half, the MLP approach was more successful.

III. EXPERIMENTS

A. Preparations

Several experiments were conducted for the implemented learning approach using different parameter settings. Next to experiments containing the whole game, other experiments were done where the game was simplified to just one game round. This was done to evaluate how successful the learning approach is in every game round. Because of different hand card sizes and different amounts of information about the opponents in every round, it is very likely that the effect of the learning approach is different. We conducted experiments which consist of only one game round to confirm these effects. Since every game round is a closed game in itself, it is easy to conduct experiments in such a simplification.

Before the experiments were carried out, some general decisions were made. The number of players was set to 4. The experimental environment consisted only of one LA, who faced three opponent players that were rule-based. Then the experiments were divided into learning and evaluation phases. Evaluation phases were carried out before every learning phase and at the end of the last learning phase. In every experiment, 20 test runs were executed with 10 learning phases and 11 evaluation phases per run. At the end of the experiment, the test run's results were averaged for the experiments result. A learning phase had 500 training games and an evaluation phase had 400 evaluation games. For every evaluation, the same set of card distributions is used, since different card distribution can bear some advantage for some of the players and it would be difficult to compare the performance changes otherwise. The set of card distributions was created randomly for 100 games. Since the playing order has some influence on the performance of the players, every player is sitting once in the first to fourth position at the beginning of a game for every card distribution in the set.

B. Results

In the following we present the results of our experiments.

First of all the experimental results for the whole game are explained. Afterwards, the results for the simplified games that consist of only one game round are discussed.

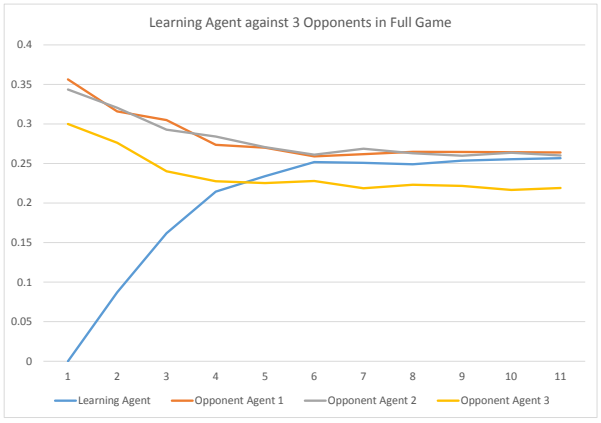

Fig. 3. Test result for Wizard game

The first experiment was conducted for the whole game as described in the game rules in II.A. The results in Fig. 3 show the percentage of won games per player for every of the 11 evaluation rounds. If every player would win equally many games, the percentage of won games per player would be 0.25 for an experiment with 4 players. So everything above the percentage of 0.25 means that the player was able to win more than to loose. The LA's results were very bad at the beginning of the first evaluation episode, which is mainly due to the very bad forecasting behavior. It also shows how essential a good forecast is for winning in this game. After some training episodes, the LA could improve his performance and was able to get a stable result averaged slightly over 0.25 in the latter half of the training. Although the agent is third place, his performance was just slightly worse than the performance of the two opponents who are better than the LA. These results show us that the LA was able to learn a strategy to play the game and improve his performance so he is almost equal with his opponents. But it also means that there is still space to improve, so that the agent is able to win over his opponents.

In the second experiment, the computer players played a simplified version of the game with only one game round. In the experiment, there were 4 players so that in total there are 15 game rounds to be played. The simplified experiment was once executed for every of the 15 game rounds separately. The results of the LA for the 15 experiments are presented summarized in Fig. 4.

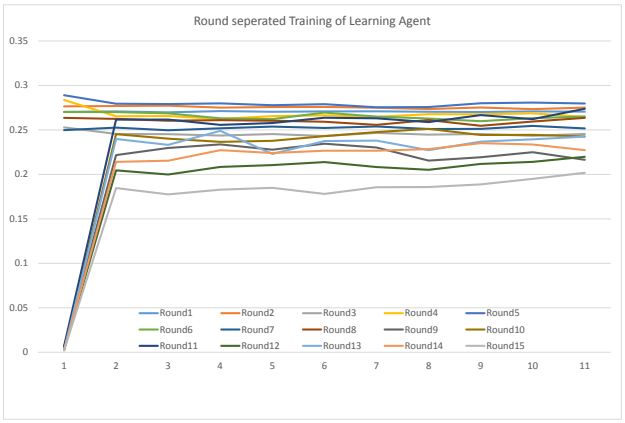

Fig. 4. Test result for simplified one round Wizard game

As in the experiment 1, the results present the percentage of won games of the LA. So every result above 0.25 can be considered as above average meaning that there is at least one opponent player who has been playing worse than the LA. Comparing the results for the different rounds, we can see that the results of the LA are differing a lot. Due to the different hand card sizes in the game rounds, the game has a different grade of information imperfection [6]. In the beginning rounds there is much less information available than in the later rounds, so it is very likely that we need to take different strategy approaches. As a first step, we already divided the forecast into two parts using a rule-based and MLP approach.

In Fig. 4, the results can be easily distinguished since MLP forecast had to be trained first and therefore started at a very

low performance in the first evaluation period. Rule-based approach performed better from the start, because no training is needed. All in all game rounds including the rule-based forecast approach performed better than the MLP-based. But they are not enough to win the game as a whole, because the latter rounds are more important to the game, since the possible reward is bigger than in the first half. Therefore, the latter half of the game can be said to be more important for winning the game. And for this part of the game, MLP-approach performed better than rule-based approach, but not yet good enough to win without fail. When we compared the game rounds within the two approaches, we realized that the performance of both approaches is not consistent in between game rounds. For example, the MLP-approach performed best for game round 11, and also good for game round 10 and 13. But the performance of the game round 12 is worse than for game round 11 and 13. We would have expected a performance result which lies between game round 11 and 13. Further studies are needed to find the reason for this and ways to improve our approach. Anyway, the results for this experiment showed us that it is important to see not only the game as a whole, but also every game round as a separate game in itself to determine how good the approach is working for the respective game round. That is why improving the performance of one round means also to improve the performance of the whole game.

IV. CONCLUSION

In this paper, we introduced an RL approach to the card game Wizard. The game consists of two game phases, forecasting and playing, which were both approached by a reinforcement learning approach separately. We conducted some experiments to evaluate the implemented method by letting an LA compete with rule-based agents. There was a visible improvement of the LA's playing strategy during the training. In the end the LA was able to perform almost equally well as his opponents. But although the LA was able to perform well in the game as a whole, a further experiment conducted for any of the 15 respective game rounds shows that the performance is differing a lot in any of the respective rounds.

To improve our approach, we need to consider the problems of every game round separately and look into approaches that can improve the performance in the whole. A good approach may be the separation of the game into phases which hold game rounds with similar information imperfection and tune the used method for these phases separately. In order to realize such an approach, we have to find an effective method for the division of the whole game and investigate the consistency of game performance with consideration of the differences among game rounds.

REFERENCES

[1] M. L. Ginsberg, "GIB: Imperfect Information in a Computationally Challenging Game," Journal of Artificial Intelligence Research, vol. 14, AI Access Foundation and Morgan Kaufmann Publishers, 2001, pp. 303-358.

[2] N. R. Sturtevant and A. M. White, "Feature Construction for Reinforcement Learning in Hearts," Lecture Notes in Computer Science, vol. 4630, Springer Verlag Berlin Heidelberg, 2007, pp. 122-134.

[3] N. R. Sturtevant and M. Bowling, "Robust Game Play Against Unknown Opponents," Autonomous Agent and Multiagent Systems (AAMAS), Association for Computing Marchinery, 2006.

[4] M. Buro, J.R. Long, T. Furtak and N. Sturtevant, "Improving State Evaluation, Inference, and Search in Trick-Based Card Games," Twenty-First International Joint Conference on Artificial Intelligence, 2009, pp. 1407-1413.

[5] H. Fujita and S. Ishii, "Model-Based Reinforcement Learning for Partially Observable Games with Sampling-Based State Estimation," Neural Computation, vol. 19, MIT, 2007, pp.3051-3087.

[6] M. Sakuta, "Research of imperfect information games," Operations Research, vol. 52, issue 1, 2007, pp. 27-34 (in Japanese).