Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

User Tailored Jass AI

Semester Thesis

Balta Nisa
baltani@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Oliver Richter, Benjamin Estermann
Prof. Dr. Roger Wattenhofer

January 16, 2022

Acknowledgements

I want to thank my supervisors Oliver Richter and Benjamin Estermann for their guidance through this project and always offering me help when getting stuck.

I also want to thank all the Jass players that helped me testing the implementations by playing against the bots and enduring all the bugs that were present at the time being.

Abstract

Ratings in competitive games are very important. On the one hand it gives an estimation about ones own skill relative to other players and on the other hand it is an opportunity to offer a skill based matchmaking.

In this thesis, we will discuss the selection of a suitable rating system and evaluate the effect of such in connection with the bots and human players. To ensure a good game experience on the WebApp we implemented a login system and a user page where the users can access the statistics based on their game plays. Furthermore we extended the WebApp with a matchmaking system where the users can choose between three different bot strengths. The system will then assign bots in the chosen bot strength to the player.

Contents

Chapter 1

Introduction

A main aspect of playing competitive games is to compare ones skill with other players. To do so, we need a rating system to evaluate the skill of the players. A widely known example of such a rating systems is the Elo rating system [1] , originally developed for chess. To maintain a game interesting for every player one needs to play against people in the same skill range. Always loosing a game can discourage a player. However, even winning all the time could decrease the experience.
Schieber Jass is one of the most played Jass variations in Switzerland and it combines luck, skill and social competences in one game. The web application (WebApp) we worked on offers the possibility to play Jass against an AI or with other online players. In this thesis, we extended the WebApp with a rating system based on Microsoft Trueskill [2]. This way the bot assignments for the games will be more personalized. Additionally, the users will be able to get an estimation of their own skill relative to other players and bots on the WebApp.

For a larger variation in bot strengths we had to modify our bots. We address the different possibilities of such changes and also the effects achieved by those. Furthermore, we evaluate the game experience of our users regarding the matchmaking and the game play of the modified bots. This is evaluated within a user study, where the users tested the different bot strengths.

The main goal of this project is to improve the user experience of the WebApp regarding the matchmaking and also giving our users the possibility to choose the difficulty they want to play. For these means we implemented a login interface for our users and choose a suitable rating system. Furthermore, we explored possible changes of the bots to get different bot strengths. We evaluated the effect caused by these changes on one hand by their effect on the bot rating and on the other hand by a user study.

Chapter 2
Background

2.1 Jass

Jass is a four player card game which is played in teams of two where the teammates are sitting on the opposite side of the table. It is also considered a traditional Swiss game where Schieber Jass is one of the most played variations. This is also the variation used in the WebApp.
The game is played with a deck of 36 cards. At the beginning of each round every player gets 9 cards. In the trump phase the starting player can either decide on a trump or pass the decision to their teammate. If the decision is passed to the teammate it does not change the starting player. The trump can be any of the suits, bottom-up or top-down.
In the playing phase the starting player has to play the first card and the turns are anticlockwise. Having played 4 cards we have a trick where the player with the strongest card gets to collect the trick and starts the new game. A round is made out of 9 tricks. The game continues until one of the teams reaches the preset target score.
For the evaluation we used the Fair Game Mode [3] where 8 rounds are played and out of those 4 are repeated rounds, where the starting hands of a previous round are repeated for the opposing players. This mode eliminates the luck factor occurring in Jass.

2.2 Bots

Whenever we use the term bot we are referring to the mixed agent which has a policy trained by Proximal Policy Optimization [4] for the trump phase and a Determinized Monte Carlo Tree Search [5] (DMCTS) with fixed 400 determinizations and 40 iterations. These values will be discussed in Chapter 5.1. We will also explore the influence of changing these values.

CHAPTER 3

Login

As basis for the rating system we had to implement a login system to be able to separate and store all the relevant information of the players and bots. The previous solution for the user login was to use only a player name and therefore it was possible, that multiple users could access the same account, which was not secured by a password. To have a proper distinction between the users a login system was necessary.

3.1 Implementation

For the implementation of the new login system we made use of Passport.js [6] which is an authentication middleware for Node.js. Passport.js provides a large set of strategies out of which we used the local login and the Google authentication. If there is any demand for additional strategies it can be easily expanded. After implementing the login system we also added a user page to the WebApp giving the users an opportunity to check their personal informations like the amount of played games and their own ranking.

3.2 Login Data

All login and user related data is saved to a SQL database with help of Sequelize [7].which is a promise-based Node.js ORM (Object-Relational Mapping Library). Queries can be written in JavaScript when using Sequelize.
For the current login system we offer a local and a Google authentication. For all user related data we use three different SQL tables. One for each authentication method and one for a global user identification. This combined table allows us to identify every user directly by their user ID instead of differentiating the login method and then searching in the corresponding table. In the global table we are also saving all the rating related parameters which we will further discuss in Chapter 4.1.

Chapter 4

Rating Systems

4.1 Elo Rating

The Elo rating system [1] was first developed to calculate the relative skill levels of chess players. The main idea of this method is to assign a number to each player which gives an estimation on the outcome of the game. The winner takes points from the losing player and player's with the same rating have draw possibility of 1.
Elo rating assumes that the skill of each player follows a normal distribution. Since our performance regarding a particular task isn't going to be always constant the collection of all performances will eventually be normally distributed. When it comes to the comparison of players, the Elo points assigned to the players are giving an estimation about the outcome of a game between these players. If the difference in the points of two players is equal to zero a draw is expected. The player with a higher rating has a higher probability of winning.

4.1.1 Updating

The new rating of Player A is calculated as follows:

$$R_{A,n+1} = R_{A,n} + K \cdot (S_A - E_A) \tag{4.1}$$

$\mathrm{E_A}$ is the expected score for Player A playing against player B. $\mathrm{R_A}$ is the Elo rating of player A. $\mathrm{S_A}$ stands for what player A actually scored and K is a weighted factor to steer the speed in which the rating evolves. This factor is based on the players skill and is often higher for the first 20 games in order to adjust the rating quickly, since the system is not yet certain about the current belief.

4.1.2 Example Chess

Let's assume we have two players one of them has a rating of 1500 points and the other 1700 points. Player 2 has a higher probability to win considering the higher rating. For the rating updates, we can think of it as betting points considering the winning chance one has. Now the winner will take some points off of the loser. We are going to use the value K=25. The resulting new ratings will be 1519 and 1681. It gets obvious that the Elo rating doesn't take the individual uncertainties into account and updates the values with a fixed factor.

4.2 Microsoft Trueskill

The Trueskill [2] rating system was developed at Microsoft Research and used for Xbox Live games. Trueskill also uses a normal distribution and characterizes a player's skill by the two values μ (mu), the mean value and σ (sigma), the variance. We can see it as a generalization of the Elo rating system. Elo rating only uses a single value and does not take the uncertainty into account therefore it also needs more games to evaluate a players skill. It is also limited to two players whereas Trueskill was designed to support multiplayer games. Therefore, Trueskill was better suited for our purpose. Furthermore, the K factor that was fixed for the Elo rating system is implemented in a dynamic way for the Trueskill rating system. Using Elo rating the change in the rating of the players is the same even if the system is more certain in the skill of one player, Trueskill updates the ratings according to the uncertainty σ it has regarding the skill belief. We will see an example afterwards.

4.2.1 Updating

The default starting values for Xbox Live games is set to $\mu = 25$ and $\sigma = 25/3$. The estimation of the player's skill is calculated with R = $\mu - 3 \cdot \sigma$. This calculation assumes, that the players skill is to 99% higher than what is calculated. The rating value is later also used for the leader board. With the given start values every player starts with a rating R=0 and the range for the rating is set to the interval [0,50]. For the WebApp we will use the same default values as stated here.

Example

	Old Mu	Old Sigma	New Mu	New Sigma
1	25.0	8.33	32.04	6.17
2	27.95	1.63	27.66	1.70

Table 4.1: 1v1 example

Table 4.1 shows the update of a 1v1 game. Here we took the μ and σ values of one of our server bots for player 2 and the default starting values for player 1. We are going to explore the outcome where player 1 won the game. Figure 4.1 shows the distribution of both players before we update the ranks. We can clearly see that the uncertainty for the skill of player 2 is significantly smaller as the one of player 1. This is due to the amount of games evaluated by the system. More updates will generally lead to a lower σ value and the server bot already played many games.

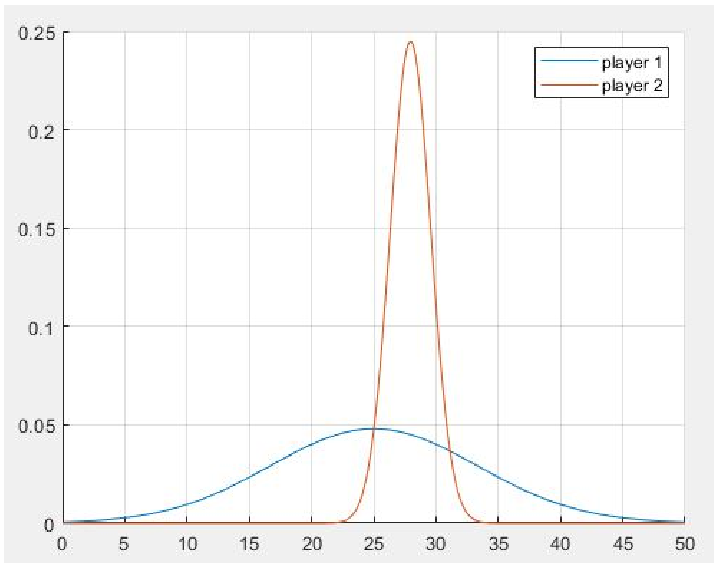

Figure 4.1: Distribution before update.

In Figure 4.2 we can observe the changes of the distributions after updating the ratings. Again the uncertainty of the players decreased. When we analyse the changes for the μ and σ of both players we see that the player 2 has slight changes, since the system is more certain about their skill. With a chance of winning of 17% player 1 was expected to lose the game. This is also the reason why the σ value of player 2 increased, because player 2 was initially estimated stronger than player 1.

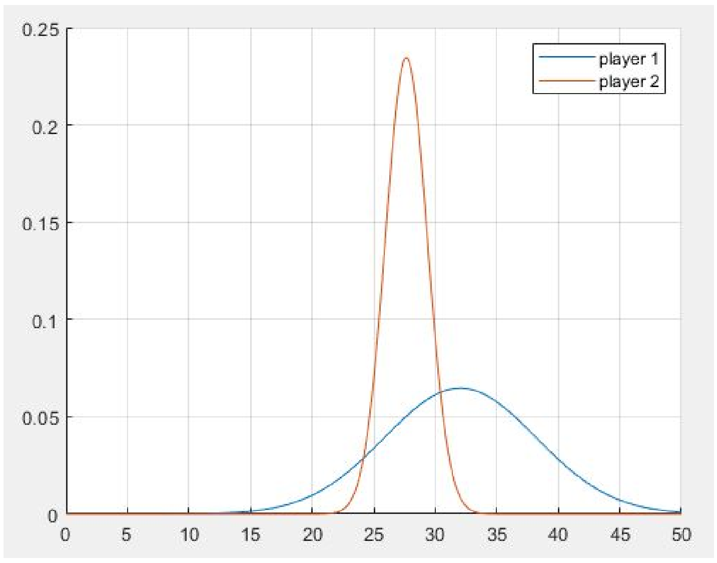

Figure 4.2: Distribution after update.

4.2.2 Matchmaking

The Matchmaking with the Trueskill System is made by using the chance of drawing. This is calculated by taking the combined Gaussian of the players and taking the probability of a draw. This will be the function value at 0. The chance of drawing will always be in the interval [0,1] where 0 indicates the worst match and 1 the best match. For team matches Trueskill takes the sum of the team players as the skill of the team.
We implemented three different bot strengths; easy, normal and hard mode. Since we used a Trueskill package for JavaScript which contained the chance of winning we had to adjust it slightly. The chance of winning equals 0.5 when the drawing probability is equal to 1. This way we could limit the allowed range to a certain threshold. For the normal mode it matches players with a chance of winning in the range [0.45, 0.55]. Algorithm 4.2.2 shows a pseudo code for the implementation of our bot assignment. For the easy mode we search for bots where our chance of winning is higher than 0.7 and for the hard mode opponents where it is below 0.3.
The implemented matchmaking system is only used for filling the bots and not for additional human users. The matching algorithm starts whenever the user presses "Start the Game". The reason for not implementing the matchmaking for our online users is that the number of players on the WebApp are not enough to get a reasonable queuing to be able to assign the best matches. The users would have to wait an unreasonable amount of time to find a match and also there might be no perfect match for this particular player which would further extend the searching time.

Algorithm 1 Matchmaking for normal bot strength

```
for bot in botlist do
    $CoW \leftarrow ChanceOfWinning(player, bot)$
    if $abs(CoW - 0.5) \geq 0.05$ then return $bot$
    else if $abs(CoW - 0.5) \geq abs(previousCoW - 0.5)$ then
        $previousCoW \leftarrow CoW$
    end if
end for
```

CHAPTER 5

Evaluation

5.1 Rating of Bots

The standard bot used on the server is the mixed agent with the values determinizations of 400 and iterations of 40. To analyse the performance of the server bots in the rating system we let the bots play against each other on the cluster. The results are then written into the database where each bot variation has 8 "instances" that behave like normal users with different names and ids. The user for the update is picked randomly for each game.
The first run consisted of 500 games with bots using four different combinations for the determinizations and iterations values. As you can see in table 5.1 there is no significant difference in the average rating of the different bot variations. Theoretically the bot with the highest determinizations and iterations values should show a higher average rating. This is likely due to the minimal difference in their performance. Since Trueskill doesn't take into account how many points each player has made but only what the outcome of the game was the difference is too small to see the effect after an average 62.5 evaluated games per user.

Determinizations	Iterations	Random factor	Average rating
400	40	-	20.3311
300	30	-	20.3103
200	20	-	20.3600
100	10	-	20.0596

Table 5.1: Bot games on cluster without random factor

To further increase the difference in strength we tried to influence the action decision of the bots with a random factor. The random factor is the probability with which the given bot will choose a random action out of the legal actions instead of the calculated best move. This change should be reflected in the ratings of the bots. Table 5.2 shows the outcome of the 1000 games we evaluated. Here the first bot is clearly the strongest among the four variations. This is also the standard bot used before.

Determinizations	Iterations	Random factor	Average Rating
400	40	0	21.9901
300	30	0.1	19.6657
200	20	0.2	20.1829
100	10	0.3	19.9052

Table 5.2: Bot games on cluster with random factor

Table 5.2 shows the results of the last run with bot variations in the random factor but no changes in the determinizations and iterations values.

Determinizations	Iterations	Random factor	Average Rating
400	40	0	21.8545
400	40	0.1	19.6447
400	40	0.2	20.0947
400	40	0.3	19.6100

Table 5.3: Bot games on cluster only random factor

When comparing the two runs with a random factor we can see that we have an overall decrease in the rating whenever the bot has a randomness implemented in their action. The reason why we don't see the impact of the factor clearly in the rating is most likely due to the pairings for the cluster games. There was no matchmaking system used and the bots also had to pair up with higher or lower ranked bots. When playing Schieber Jass the outcome not only depends on ones own performance but also on the skills of your team partner. Therefore the ratings of the three other bots is coming out noisy.

5.2 Users Study

The main goal of the user study was on the one hand to find out how the WebApp users experienced the matchmaking and on the other hand if the randomness of the weaker bots was noticeable. For the user study every participant created an account and played three quick games to get a more stable rating. After those three ranking games they had to play one game for every bot strength. The players where asked to fill out a short form after each of these games, see Appendix A for details.

Figure5.1 shows how the participants rated the game play of the assigned bots. We can see, that the bots in the easy mode were rated more random then the ones assigned in the hard or normal mode. In figure 5.3 we can see how the bots where assigned to a game regarding the chosen bot strengths. We used the bot parameters in table 5.2 where bot 0 corresponds to the first row, bot 1 to the second row and so on. The user study shows that in the hard mode the bots had a rather natural game play this is most likely due to the use of bot 0 which had no random factor. Bot 0 was exclusively assigned in the hard mode. The participants also observed the randomness of the bots, which is reflected in figure 5.1. There we can observe that bots in the easy mode were considered more random than in the other two modes. Finally, when looking at figure 5.2 we can see that the participants were satisfied with the overall bot assignments. The matchmaking for the normal and easy modes both appear more random, this could be due to the randomness of the assigned bots. Random actions can also make the bot appear less human like and influence the overall experience of the game.

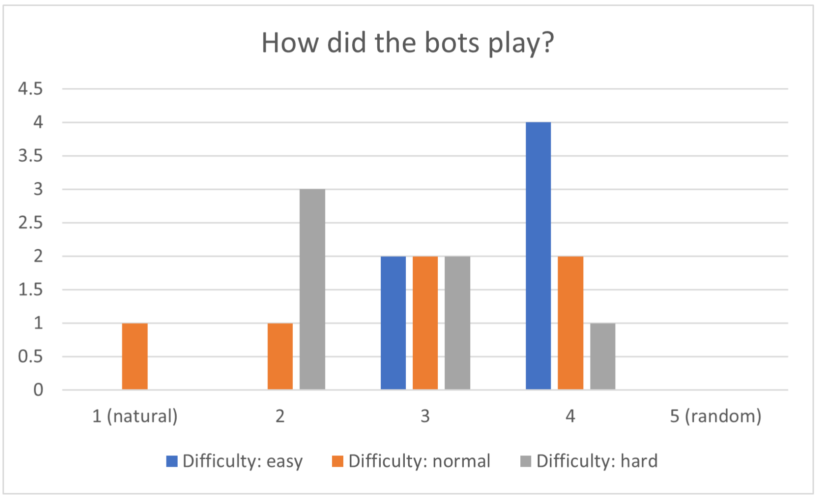

Figure 5.1: How did the bots play?

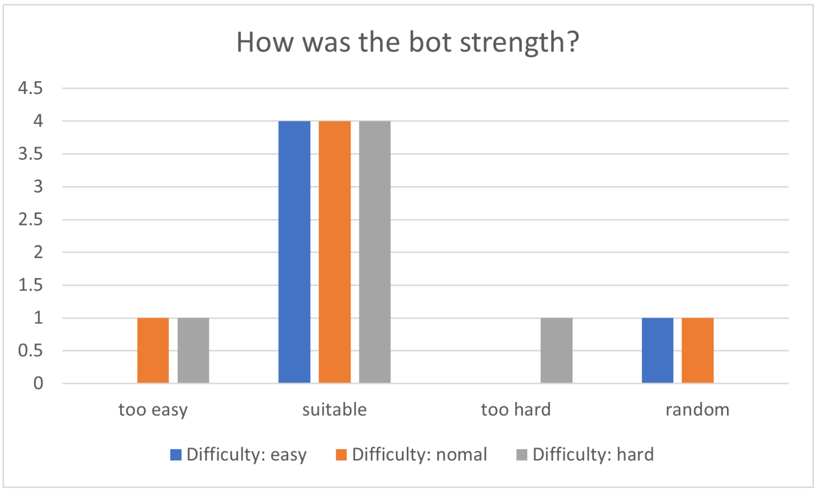

Figure 5.2: How was the bot strength?

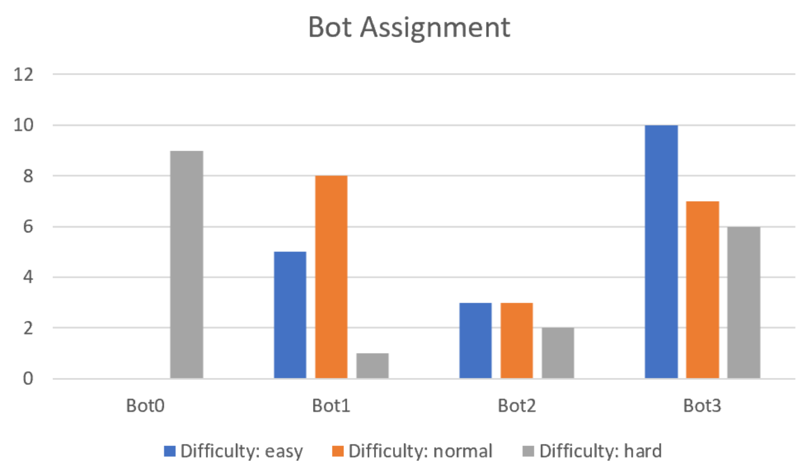

Figure 5.3: Bot assignments partner and enemy team combined.

Chapter 6

Conclusion

In this project we have worked on a rating system to enable a better matchmaking on the Server. The user study showed that the bot assignment worked fine and the participants were mostly satisfied with the matchmaking. The biggest issue seemed to be the random factor. Since the lower ranked bots with a randomness in their action decision seemed to be less human like and the whole matchmaking appeared to be random. The hard mode performed best regarding the bot assignment. This mode was assigning mostly bot 0 to the players. Some participants reported, that the random plays of the bots were rather confusing and also influenced their own game performance. We decided to always assign bot 0 as partner for our human users. The enemy bots will still be assigned according the chosen bot strength. This way we can guarantee a improved game experience, instead of users being dissatisfied by the performance of their own partner.
For further extensions we would consider to make different changes in the bot parameters to be able to create different strengths and achieve bad players with less randomness. Furthermore we would add a reset possibility for the password. This caused some trouble at the beginning of the user study. It would also be interesting to find other ways of impacting the rating of the bots instead of a random variable. This would also solve the issue with how they appear to the human players.

Bibliography

[1] A. E. Elo, *The Rating of Chess Players.* Arco Pub., 1978.

[2] Trueskill ranking system. [Online]. Available: https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/

[3] Developing a jass ai server. [Online]. Available: https://pub.tik.ee.ethz.ch/students/2020-HS/BA-2020-47.pdf#page=18&zoom=100,0,0

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, Jul. 2017.

[5] 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*, Oct. 2011.

[6] Passport.js : Simple, unobtrusive authentication for node.js. [Online]. Available: https://www.passportjs.org/

[7] Sequelize: A promise-based node.js orm tool. [Online]. Available: https://sequelize.org/

Appendix A

User Study

To analyse the rating systems regarding our online users, the participants where asked to fill out a form after playing a game with selected bot strengths.

1. How was the bot strength in this round?
2. On a scale from 1 (human/natural) to 5 (random), how did the bots play?

Table A.1 shows the answers of all participants

	Bot strength	Bot game play	Difficulty
1	suitable	3	Hard
2	suitable	3	Normal
3	suitable	4	Easy
4	too easy	4	Normal
5	suitable	1	Normal
6	random	4	Easy
7	too strong	2	Hard
8	random	4	Normal
9	too easy	3	Hard
10	suitable	2	Normal
11	suitable	3	Normal
12	suitable	4	Easy
13	suitable	4	Hard
14	too easy	4	Easy
15	suitable	2	Hard
16	suitable	2	Hard
17	suitable	3	Easy
18	suitable	3	Easy

Table A.1: User study results