university of groningen / faculty of science and engineering

A pure coordination game for the multi-agent card game The Mind

Bachelor's Project Thesis

Thomas Veldboer, s3686450, t.i.veldboer@student.rug.nl,
Supervisors: Dr H. De Weerd

Abstract: The game "The Mind" is a game in which players have to cooperate to play cards in ascending order without communicating. This study focuses on different strategies while playing The Mind and which strategy performs best. Four different strategies will be evaluated in an agent model made in Netlogo. Out of these four strategies the most patient strategy, in regards of time, performed the best. This has to do with the fact that agents who used this strategy were more patient and waited with playing the card until the right time to do so. In general these results are hard to apply in real life, since a lot more variables are at play in real life, for example the non-verbal communication that humans use.

1 Introduction

For many people card games are played to have fun with other people and relax. Card games are often experienced as easy, but when it comes to predicting the outcome, there are many points of uncertainty. There are a lot multi-player card games with unknown variables for the players participating in the game. Take for example the cards in the hands of the other players or cards that have yet to be dealt. This can be especially hard for programmers to write down in code, which possibly leads to another interpretation of the game or the game not working as intended.

The card game that will be researched in this study is called "The Mind" (Warsch, 2018). The Mind is a game played by two or more players. Every player receives the same amount of cards, each containing a number from 1 to 100. There are no turns in the game and every player is free to play their cards when they think the card should be played. The goal is to play the cards in ascending order. The cards that are played, need to be placed on a central pile. However, there is a catch. Players cannot communicate in any way with each other, verbally or non-verbally. The players can also not see each other's cards, but only their own. The game is played in levels. In the first level every player receives one card, the second level two cards and so on. If the players manage to complete level one and every player plays their card in the right order, they continue with level two. It does not matter who plays their card first, as long as the cards are played in ascending order. If the players do not complete the round, they fail the game and the game ends. The main goal of the game is to cooperate and get to the highest level as possible, which requires great collaboration.
Normally, the game is played with cards which represent lives and cards that represent shurikens. These items do make the game more interesting, but are beyond the scope of this paper.

The game is an example of a pure coordination game with possible Pareto efficient solutions. A pure coordination game is a coordination game in which both agents obtain the same outcome, either win or lose (Goranko, Kuusisto, and Rönnholm, 2020). Pareto efficient solutions are solutions in a situation in which an individual cannot change their choice without negatively influencing some agent (Scalzo, 2010). Take for example table 1.1. It can be seen that if both agents choose choice 1 they get an equal and positive pay off, the same holds for choice 2. However, if each agent picks a different choice, the agents receive a zero pay off.

Unlike typical coordination games, The Mind is not played is discrete steps, but in continuous time. The choices are relatively easy: to play the card or

	Choice 1	Choice 2
Choice 1	1,1	0,0
Choice 2	0,0	1,1

Table 1.1: An example of coordination game with two Pareto efficient outcomes.

not to play the card. The timing however is the hard part; when is the right time to play a card.
For coordination games in continuous time, the type of game and the number of players influence the process of the game (Leng, Friesen, Kalayci, and Man, 2018). Take for example a study done on the prisoner's dilemma in continuous time (Friedman and Oprea, 2012). The study by Friedman and Opera was done with human participants, which participated in a continuous time game, just like The Mind. The study found that in a continuous time experiment, for the prisoner's dilemma problem, mutual cooperation rose to a high level (a median of 81 percent to 93 percent). The study did also find that if the participants had a 60 second time period, they reached a median rate of 100 percent cooperation quickly.

To find how cooperation will work in The Mind the following question will be asked: "Do patient strategies in the game "The Mind" yield better results?". This will help to give an insight in how cooperation games work in a continuous time frame. It will also contribute to cooperation games displayed by agent models. To find the answer, an agent model will be build in which different strategies of playing the game will be compared to each other. Every strategy will be compared to every strategy to find how strategies influence each other and which combinations would work the best with the given cards.

2 Methods

The agent model will be explained in this section. The main parts of the system are the game itself, the agents, the strategies and how those are combined in the program. The program was made in NetLogo.

2.1 The game "The Mind"

The experiment consists of agents with different strategies playing the game, programmed in Netlogo. Apart from the shuriken and life cards, the game is played differently on one more aspect. This aspect is the fact that agents will play until they reach level 10, while the normal game can go up to level 15. This was chosen to have a clear winning statement for the agents. The one thing that agents could do, which humans are not able to, is playing the cards at the same time. If this would happen in real life, either one player would back off or both players would back off after which the process would start again. To solve this problem multiple solutions were possible. Trying to simulate what humans do is a possibility, to forbid playing two cards in the same round is also a possibility or even to let the agents fail when two cards are played at the same time. When the program was eventually developed this did not seem a problem, since the actions of the agents are successive instead of simultaneously.

2.2 The agents

For the agents it is important how they are situated in the model. This could be in the form of a certain grid or by moving around randomly. For this study it was chosen that the agents move around randomly and when two agents meet each other, they will play the game. The agents move inside a set area of 16 by 16 cells. If two agents come across the same cell, they will play a game on that cell. The agents can move across the borders and they will appear at the border on the other side. If another agent comes across the playing agents the agent will move again to get away from the playing agents. This was chosen so the game is played with a fixed number of agents, in this case two agents, instead of random agents joining in.
Another important aspect for the agents is how the time of when to play the card will be measured. This can for example be done by looking at the time, ticks in this case, a percentage of how likely the agent is to play a card and other variations of this. Examples of this are looking at the time since the last card was played instead of the overall time, looking at percentages as either play or do not play instead of taking the chance of a percentage (e.g.

60% results in play every time or 60% of the time).

It was decided to go for percentages. This was the case since it makes a clear decision if an agent plays a card or not at a certain percentage, simulating human behaviour from the game (either play or do not play).

2.3 Strategies

Every agent needs a strategy to play the game, as explained before the outcome of the strategy will be in percentages of how big the chances are that the agent plays the card. Every strategy is formulated as in equation 2.1. Note that p is capped to a range from 0% to 100%. That is, outcomes over 100% are treated as if they were 100%, and outcomes below 0% are treated as if they were 0%.

$$p = 100\% - (ch - ct) + 1 * tick \tag{2.1}$$

$$p = 100\% - (ch - ct) * 1.2 + 1 * tick \tag{2.2}$$

$$p = 100\% - (ch - ct) * 0.8 + 1 * tick \tag{2.3}$$

$$p = 100\% - (ch - ct) + 1.2 * tick \tag{2.4}$$

The equations show the probability percentage p, the card in the agent's hand ch, the card on the table ct and the $tick$, which is the time measured since the last action. For example, playing a card or starting a new game/level. It is important to notice that that the card on the table is the last played card in the game. If the level is over the cards will be removed from the table and from the players and the game will end. Take for example the card 50, when there is no card played yet. The initial chance would be 0% for strategy 1, strategy 2 and strategy 4, while the chance for strategy 3 would be 17,6%. When 20 ticks have passed, strategy 3 has a 60% chance of playing while the other strategies have a chance below 45,4%.

These four strategies were chosen to get a better understanding on how a certain strategy can influence the winning percentage. These four strategies show three different sides from the normal strategy, 2.2 shows a more immediate patient approach, 2.3 shows a more immediate aggressive approach and 2.4 shows a more aggressive approach over time. These strategies show different human styles of playing the game, which can show what would be the best strategy to play and the best strategy with the other strategies.

2.4 The program

The program is a model made in Netlogo. The main function exists of a setup function, two subfunctions and a cooldown. The two subfunctions are: move-agents and play-game.

2.5 Setup

In the setup everything is made ready for the program to be run. In this case it first clears everything of the screen. Then it proceeds to give two lists, *cards-played* and *cards-dealt*, to the patches. After that it sets up the agents. This is done by moving the agent to a random patch, giving them a color and a list of the cards. It also sets a few variables: *got-card* to false, *agent-win* and *agent-loss* to zero, *cooldown* to one, *level* to one and the *level-avg* to zero. The last thing the setup function does, is reset the amount of ticks.

2.5.1 Move-agents

Move-agents is the simpler module of the two, consisting of a check if an agent is playing the game and a random move. Checking if the agent is playing the game is done by monitoring the patch it is on. The random move is done by letting the agents turn a random degree right and left (with a max of 90 degrees) and moving forward one step.

2.5.2 Play-game

Play-game is the main function of the program, by which everything is controlled. *Play-game* starts with checking if the agent is on a patch with another agent. If this is the case, the patch will turn white and the agents start a new game. After that, the function will call for all agents on a white patch if they have an empty cards list, if *got-card* is set to false and if the *cooldown* is lower than one.

In this case it will look if the *cards-dealt* list is

empty. If this is also true the agent will receive as much cards as the number of the variable level the agent has. It will put the card that is first dealt inside the *cards-dealt* list. If there are already cards dealt the agents will still receive as much cards as the number of the variable *level*. A thing that changes is the fact that it will also check if there will be only unique cards by using the *cards-dealt* list, so that the agents playing the game cannot have the same cards. After this, the *percentage* and *time* will be set to zero, *got-card* will be set to true and the cards from the agents will be sorted.
Subsequently, or if the agents did not meet the requirements two functions are called, first the function *calculate* and then the function *play-card*.

The function *calculate* first checks if the list cards is not empty. If this is not the case, it will look if there is already a card played. If so the equations 2.1, 2.2, 2.3 and 2.4 will be used depending of the strategy of the agent. Because there is no card played yet *ct* will be set to zero. If there is already a card played, agents will use the equations 2.1, 2.2, 2.3 and 2.4 again, depending on their strategy. The last function of the calculate function is adding one to the time so that every round the percentage changes even if the cards do not change.

The *play-card* function starts with a check to see if the cards list is not empty. If that is not the case it will take a randomized number to see if the calculated odds of the percentage are higher than the randomized number. If that is the case the function *check-if-correct* will be called. If that is not the case the function will end and start at the beginning again.

The *check-if-correct* function starts with looking at the current *level* of the agent. If the *level* is equal or higher than 10 it will add a *game-win* and an *agent-win* and will set everything back to normal. This includes removing all the cards from the patch and the agents, setting the *level* back to one, the *time* to zero, the *cooldown* to one, the *got-card* to false and the patch color to black.
If *level* 10 is not reached yet the program will check if there is a card played already. If this is not the case it will put the card from the agent in the *cards-played* list and remove it from the agent cards list. It will also set the *time* back to zero. When there is already a card played the program will see if the card that is played is higher than the card that is on top of the *cards-played* list. If this is correct, the program asks if the cards of both agents are empty. If this is the case the current *level* is won, resulting in emptying the *cards-played* and *cards-dealt* lists, setting *got-card* to false, adding one to the current *level* and adding one to the *level-avg*. If the *level* is not complete yet the card is added as first card into the *cards-played* list, the *time* is set to zero and the card that is played is removed from the agent and added into the *cards-played* list.
When the card is played out of order, the card that is played is lower than the highest card from the *cards-played* list, another path will be taken. This starts with adding one to the *game-lose* and the *agent-loss*, followed by emptying the *cards-played* and *cards-dealt* lists. Because the game is over, the patch color will be set to black and the agents cards lists will be emptied out. It ends with setting the variables, *got-card* back to false, *cooldown* to one and the *level* to one.

2.6 Testing

To test how every strategy performs the Behaviourspace tool of Netlogo will be used. An example can be seen in figure A.1. As can be seen in the figure there would be ten repetitions, in which the variable game-win would be counted until the total games reached a hundred (or more in case of two games ending on the same tick while surpassing a hundred). These runs were done for every strategy across each other and the strategies playing with only themselves. When the strategies would play with each other, there would be ten agents of both strategies. When the strategies played with themselves there would be twenty agents of the strategy. This was chosen, so that no matter which strategy/strategies would play, there would always be twenty agents playing.

3 Results

The data was gathered by using the Behaviourspace tool of Netlogo, as can be seen in figure A.1. At the end of the run the screen would show something like figure B.1, which shows the fact that the

different strategies won a certain games over time and the average level of the different strategies. Figures 3.1 and 3.2 show a total of 4 outliers. It was decided to still use the data since the situation in which the runs took place were not different than the other runs. It was chosen to gather the results by playing 100 games and looking at the games won by each strategy. This process was repeated 100 times for each experiment. A point to take into consideration when looking at this, is that when two agents reach level 10, they have played 110 cards in total. This shows that a game is not just playing a card once, but has more to it. Take for example a run with 50 won games. This holds that at least 5600 moves were made in the run (5500 for the winning games and 50*2 for the losing games). It was chosen to do the experiments in this way since it would show how each strategy performs with the same strategy or other strategies. This helps with answering the goal of this study.

The data can be divided into two groups, results of games using two different strategies and the results of games using the same strategy. These are summarised in figures 3.1, and 3.2. What is interesting about these figures is the fact that strategy 2 performs the best in every test. Another point of interest is to see that strategy 3 has a low performance overall, but especially when playing with the same strategy. The last point to notice is the fact that strategy 1 and strategy 4 perform similarly.

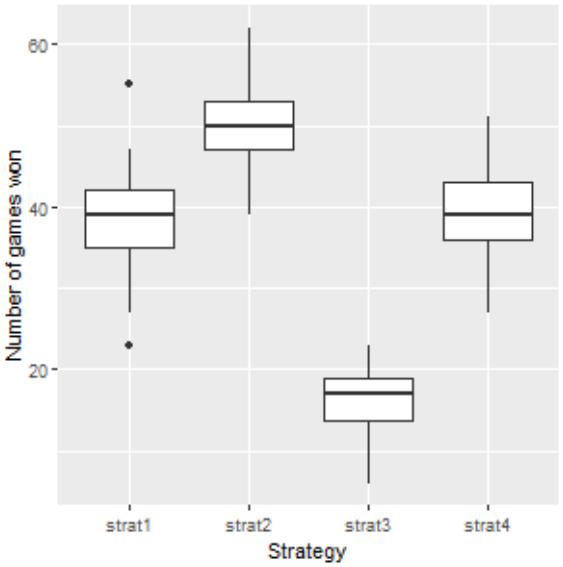

Figure 3.1: This figures shows a boxplot of the number of games won out of 100 games played in experiments with 20 agents using the same strategy.

4 Discussion

4.1 Conclusion

Looking at the obtained results some conclusions can be drawn. A first conclusion that can be drawn, is the fact that the agents who use the third strategy underperform both when cooperating with other strategies as when cooperating with itself. Looking at the low scores from cooperation with agents with other strategies can be explained by the fact that strategy 3 is an aggressive strategy, as can be seen in equation 2.3. Because of the way the strategy acts it can be the case that the agents using it play cards more quickly than the agent they are playing the game with. This results in higher chances of playing the wrong card.
An interesting observation is that strategy 2 does the exact opposite of strategy 3. With the strategy of decreasing the percentage of playing a card, strategy 2 has a higher percentage of winning. This has to do with the fact that the agents that use the strategy are more patient and only play the card when the card should be played. Showing that more patient strategies reward themselves better.
Something else that stands out is the similarity between strategies 1 and 4. Both seem to perform similar, both in the tests with the same strategies and the tests with the different strategies. Looking at figures 3.1 and 3.2 it can be seen that strategy 4 just outperforms strategy 1, however this is not a significant difference. This indicates that the small increase of percentage over time, does not influence the winning chance.

Taking this into account, the research question, “Do patient strategies in the game “The Mind” yield better results?”, can be answered. Of the investigated strategies, patient strategies seem to play better than aggressive strategies. Most

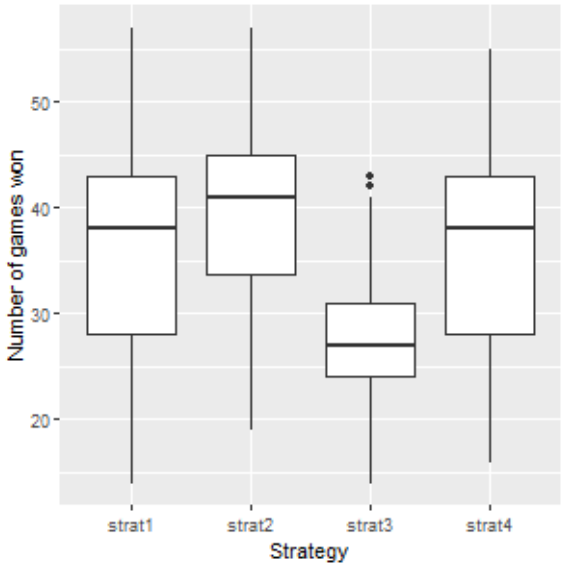

Figure 3.2: This figure shows a boxplot of the number games of games won out of 100 games played in experiments with 20 agents using different strategies. The 20 agents were divided in 10 agents of each strategy. This boxplot includes every strategy playing with every strategy except for playing with the same strategy.

likely due to do the fact that patient strategies wait until the right time to play the card instead of playing the card more aggressively and losing the game. An important point to notice is the fact that agents can only win the game by reaching level 10. Level 10 can only be reached if all the 110 cards are played correctly, which is more likely to happen if the cards are played with certainty instead of risk.
In real life it would be most successful to take a more patient approach while playing The Mind, since it makes it more likely to reach higher levels. It should be noted that this only provides a benefit if all players play patiently instead of just one player.

4.2 Limitations

There are some limitations to this research. For example, the fact that there are only four strategies investigated here, which limits the research to only look at these strategies. It could be that an even more passive strategy would do better.

Future research should look further into the methods used in this study. Since a program was used, it will always be hard to compare to human data. Not only regarding non-verbal human interaction, which influences the game a lot. But also looking at how humans would use a strategy, since there are no ticks in life and people cannot do constant calculations of the cards in their hand and the cards on the table.
A last opportunity for future research is based on the fact that strategies could also play against themselves in the experiments with different strategies. This can influence the scores a lot if certain strategies win or lose more in a run. This could be solved by changing the agents from twenty to two.

References

Daniel Friedman and Ryan Oprea. A continuous dilemma. *American Economic Review*, 102(1): 337–63, 2012.

Valentin Goranko, Antti Kuusisto, and Raine Rönnholm. Rational coordination with no communication or conventions. *Journal of Logic and Computation*, 30(6):1183–1211, 2020.

Ailin Leng, Lana Friesen, Kenan Kalayci, and Priscilla Man. A minimum effort coordination game experiment in continuous time. *Experimental Economics*, 21(3):549–572, 2018.

Vincenzo Scalzo. Pareto efficient nash equilibria in discontinuous games. *Economics Letters*, 107(3): 364–365, 2010.

Wolfgang Warsch. *The Mind*. Nürnberger-Spielkarten-Verlag GmbH, 2018. Card game.

A Appendix

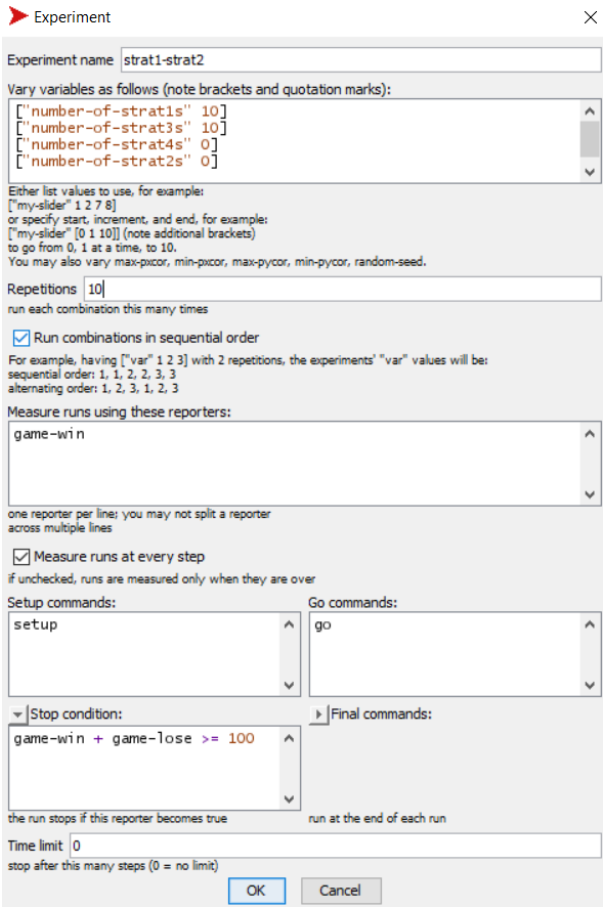

Figure A.1: An example from the Behaviourspace tool of Netlogo for strategy 1 and strategy 2

B Appendices

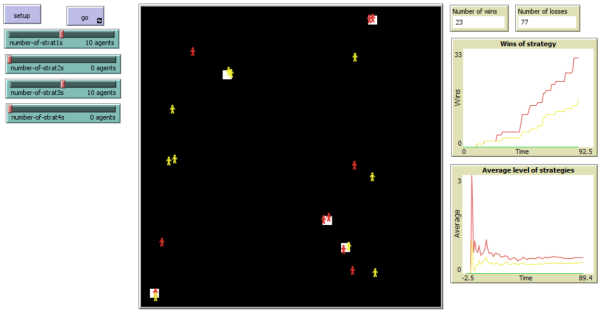

Figure B.1: An example from a run, which has reach the end goal of 100 games. The wins by different strategies and the averages of the strategies can be seen over time